BAB III

ANALISIS DAN PERANCANGAN

3. 1 Analisis

Analisis bertujuan untuk mengidentifikasi permasalahan-permasalahan yang terdapat pada sistem serta menentukan kebutuhan-kebutuhan dari sistem yang dibangun. Analisis tersebut meliputi analisis masalah, analisis kebutuhan data, analisis kebutuhan non fungsional, dan analisis sistem.

3. 1. 1. Analisis Masalah

Kampus *Online* Unikom merupakan jejaring sosial yang dikhususkan bagi civitas Unikom. Belum tersosialisasinya jejaring sosial Kampus *Online* Unikom menyebabkan jumlah pengguna masih sangat jauh jika dibandingkan dengan jumlah total civitas Unikom. Selain itu belum terintegrasinya fasilitas-fasilitas akademik yang ada seperti fasilitas untuk mengecek nilai dan autodebet secara *online* sedikit banyak akan menyulitkan pengguna itu sendiri.

3. 1. 2. Analisis Sistem Operasi

Analisis sistem operasi menjelaskan perbandingan antara sistem operasi Android dengan sistem operasi *mobile* yang digunakan saat ini dalam mengakses Kampus *Online*, Autodebet *Online*, dan Nilai *Online*. Perbandingan sistem operasi dilakukan dari segi perangkat keras, perangkat lunak, dan mekanisme kerja atau alur setiap sistem operasi dalam memetakkan setiap proses yang terdapat pada Kampus *Online*, Autodebet *Online*, dan Nilai *Online*.

Adapun sistem operasi yang dibandingkan dalam analisis ini adalah iOS, Blackberry, Symbian, dan Android. Karena banyaknya macam-macam *handphone* yang digunakan oleh setiap sistem operasi maka dalam analisis sistem operasi ini diambil *handphone-handphone* tipe *low end* dari masing-masing sistem operasi. Berikut ini tipe *handphone* yang digunakan sebagai pembanding seperti terlihat pada Tabel III.1.

Tabel III.1 *Handphone* Pembanding

Sistem Operasi	Tipe *Handphone*	Harga
Android	Samsung Galaxy Mini	1,5 Juta
iOS	iPhone 3G	2,7 Juta (2nd)
Symbian	Nokia C6	2,4 Juta
Blackberry	Storm 9500	1,4 Juta (2nd)

Handphone tersebut merupakan *handphone* dengan harga terendah dari masing-masing sistem operasi yang dianggap mampu untuk memenuhi kebutuhan dari aplikasi *mobile* Kampus *Online* yang nantinya akan dibangun. Adapun perbandingan setiap sistem operasi dari segi perangkat keras, perangkat lunak, dan mekanisme kerja dengan *handphone* seperti pada Tabel III. 1 adalah sebagai berikut:

1. Perbandingan Perangkat Keras Sistem Operasi

Perbandingan perangkat keras memaparkan perbandingan perangkat keras yang digunakan oleh masing-masing sistem operasi, yaitu iOS, Blackberry, Symbian, dan Android. Perangkat keras yang dibandingkan merupakan perangkat keras dibutuhkan oleh pengguna dalam mengakses Kampus *Online*, Nilai *Online*, dan Autodebet *Online*, yaitu *processor*, WiFi, *memory card*, dan *alert*. Adapun perbandingan sistem operasi iOS, Blackberry, Symbian, dan Android dari segi perangkat keras yang digunakan dapat dilihat pada Tabel III.2.

Tabel III.2 Perbandingan Perangkat Keras Sistem Operasi

Sistem Operasi	*Processor*	Wifi	*Memori Card*	*Alert*
Android	600 MHz	Ya	up to 32GB	MP3 - Vibration
iOS	412 MHz	Ya	up to 16GB	MP3 - Vibration
Symbian	434 MHz	Ya	up to 16GB	MP3 - Vibration
Blackberry	528 MHz	Ya	up to 16GB	MP3 - Vibration

Dari perbandingan perangkat keras pada Tabel III.2 maka dapat disimpulkan bahwa dari segi perangkat keras Android lebih unggul dalam hal kecepatan *processor* dan kapasitas memori maksimal yang bisa ditampung. Semakin cepat *processor* tentu saja aplikasi *mobile* Kampus *Online* yang dibangun nantinya akan berjalan dengan maksimal. Selain itu dukungan kapasitas memori yang besar untuk menyimpan data dari aplikasi *mobile* Kampus *Online* akan membuat performasi aplikasi menjadi meningkat.

2. Perbandingan Fitur

Perbandingan fitur memaparkan perbandingan sistem operasi, yaitu iOS, Blackberry, Symbian, dan Android dari segi fitur-fitur yang dimiliki oleh sistem operasi *mobile* dalam mengakses Kampus *Online*, Nilai *Online*, dan Autodebet *Online*, yaitu dari fitur *browser*, paket data, dan sistem notifikasi yang digunakan. Adapun perbandingan sistem operasi iOS, Blackberry, Symbian, dan Android dari segi perangkat fitur yang dimiliki dapat dilihat pada Tabel III.3.

Tabel III.3 Perbandingan Fitur Sistem Operasi

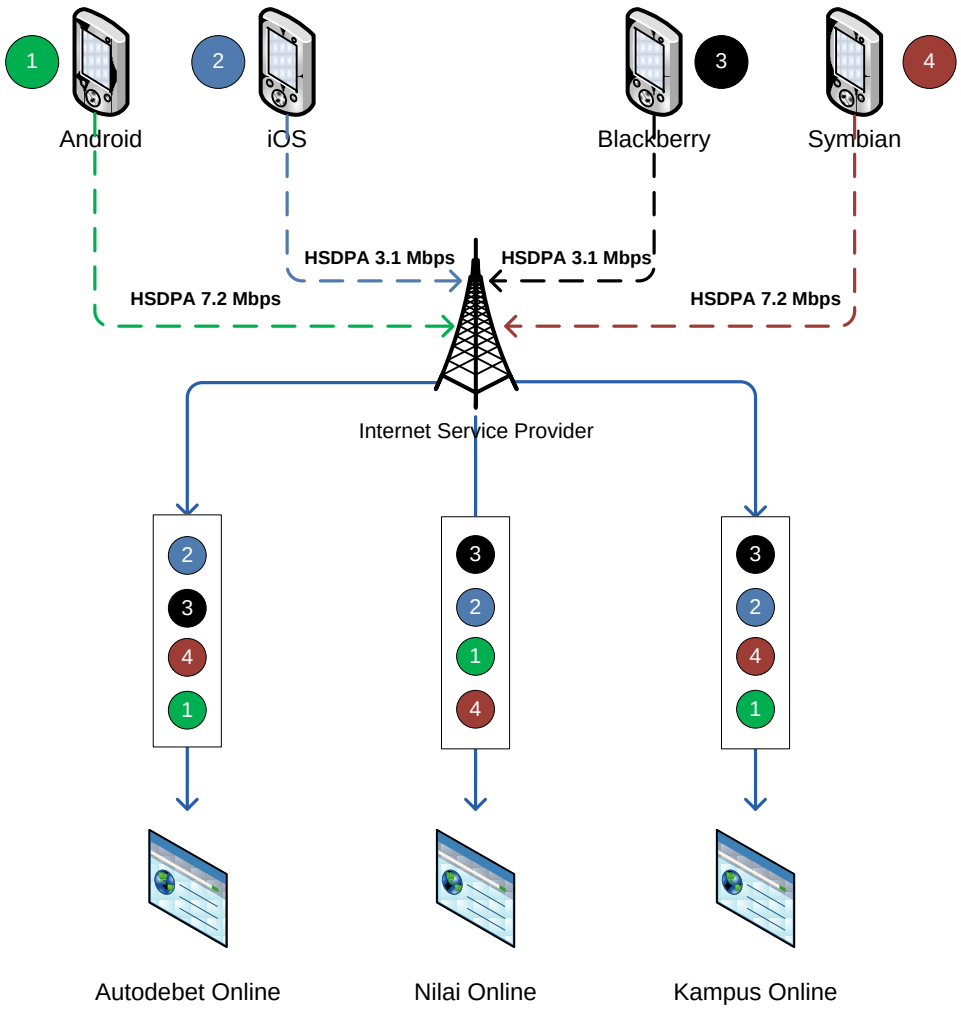
Sistem Operasi	*Browser*	Paket Data	Sistem Notifikasi
Android 2.X	Safari	HSDPA 7.2 Mbps	Ya
iOS 3.X	Chrome Browser	HSDPA 3.6 Mbps	Tidak
Symbian 3.X	WebKit Browser	HSDPA 7.2 Mbps	Tidak
Blackberry 5.X	WebKit Browser	HSDPA 3.6 Mbps	Tidak

Dari perbandingan fitur sistem operasi pada Tabel III.3 maka dapat disimpulkan bahwa Android lebih unggul dalam hal kecepatan transfer data yang sangat dibutuhkan untuk mengakses mengakses fasilitas *online* seperti Kampus *Online*, Nilai *Online*, dan Autodebet *Online* dengan cepat. Selain itu Android juga memiliki sistem notifikasi sebagai pusat pemberitahuan bagi pengguna. Nantinya sistem notifikasi ini akan digunakan oleh aplikasi *mobile* Kampus *Online* untuk membertahukan kepada pengguna notifikasi-notifikasi terbaru yang terdapat pada jejaring sosial Kampus *Online* sehingga pengguna bisa menikmati fitur notifikasi seperti pada Kampus Online versi *web*.

3. Perbandingan Mekanisme Sistem Operasi

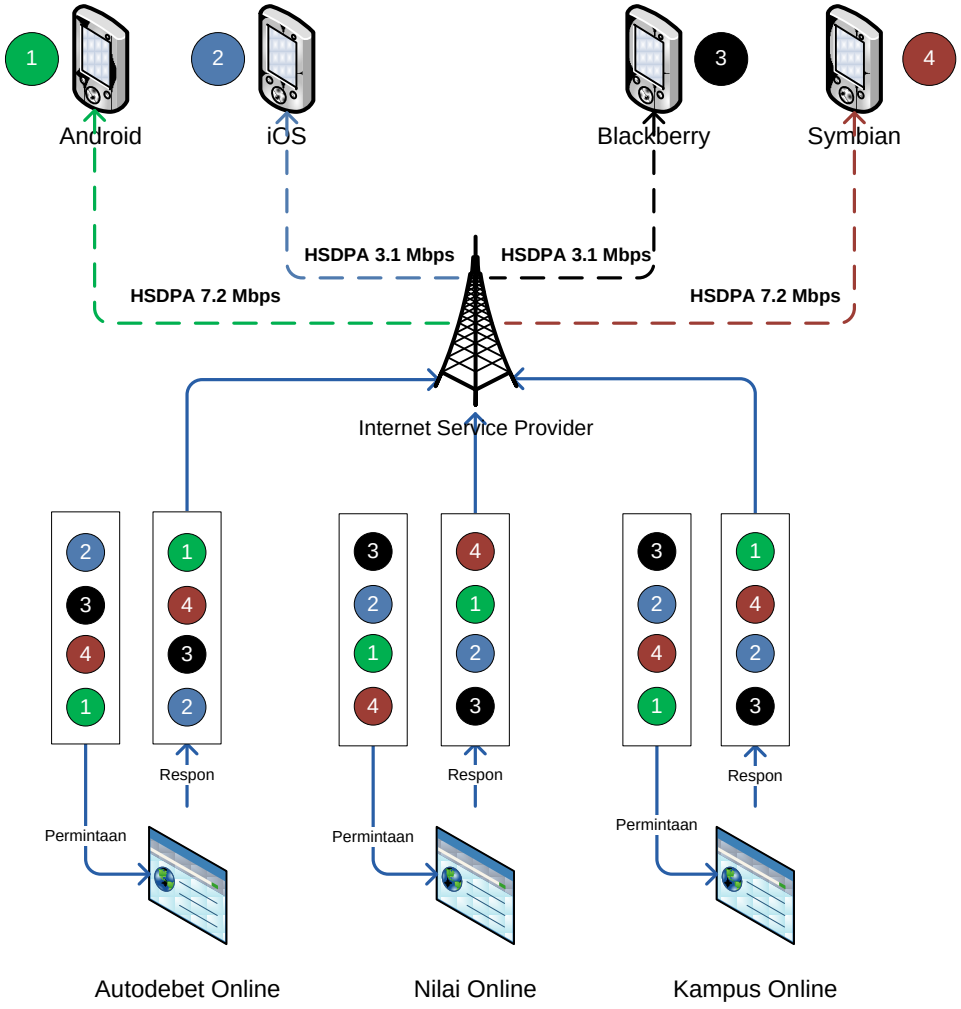
Perbandingan mekanisme sistem operasi menggambarkan perbandingan sistem operasi, yaitu iOS, Blackberry, Symbian, dan Android dalam mengakses Kampus *Online*, Nilai *Online*, dan Autodebet *Online*. Mekanisme tersebut menggambarkan bagaimana setiap sistem operasi melakukan permintaan data terhadap *server* yang ada saat ini secara bersama-sama.

Server yang ada saat ini, yaitu *server* Kampus *Online*, Nilai *Online*, dan Autodebet *Online* akan menerima permintaan data dengan sistem *First Request First Serve*, yang berarti bahwa *server* akan melayani setiap permintaan data dari sistem operasi yang pertama kali melakukan permintaan. Urutan permintaan dari setiap sistem operasi terhadap *server* ditentukan oleh kecepatan paket data dari masing-masing sistem operasi. Adapun perbandingan sistem operasi iOS, Blackberry, Symbian, dan Android dalam mekanisme pengaksesan Kampus *Online*, Nilai *Online*, dan Autodebet *Online* dapat dilihat pada Gambar III.1.



Gambar III.1 Mekanisme Pengaksesan *Server*

Server Kampus *Online*, Nilai *Online*, dan Autodebet *Online* akan memberi respon terhadap permintaan data terhadap sistem operasi yang pertama kali melakukan permintaan data. Hanya saja tidak selamanya sistem operasi yang paling cepat meminta akan cepat juga dalam mendapatkan data yang diminta. Kecepatan pengambilan data dari *server* tergantung pada kecepatan paket data yang dimiliki oleh setiap masing-masing sistem operasi. Berikut ini gambaran *server* Kampus *Online*, Nilai *Online*, dan Autodebet *Online* dalam memberikan respon terhadap permintaan sistem operasi seperti terlihat pada Gambar III.2.



Gambar III.2 Mekanisme Respon *Server*

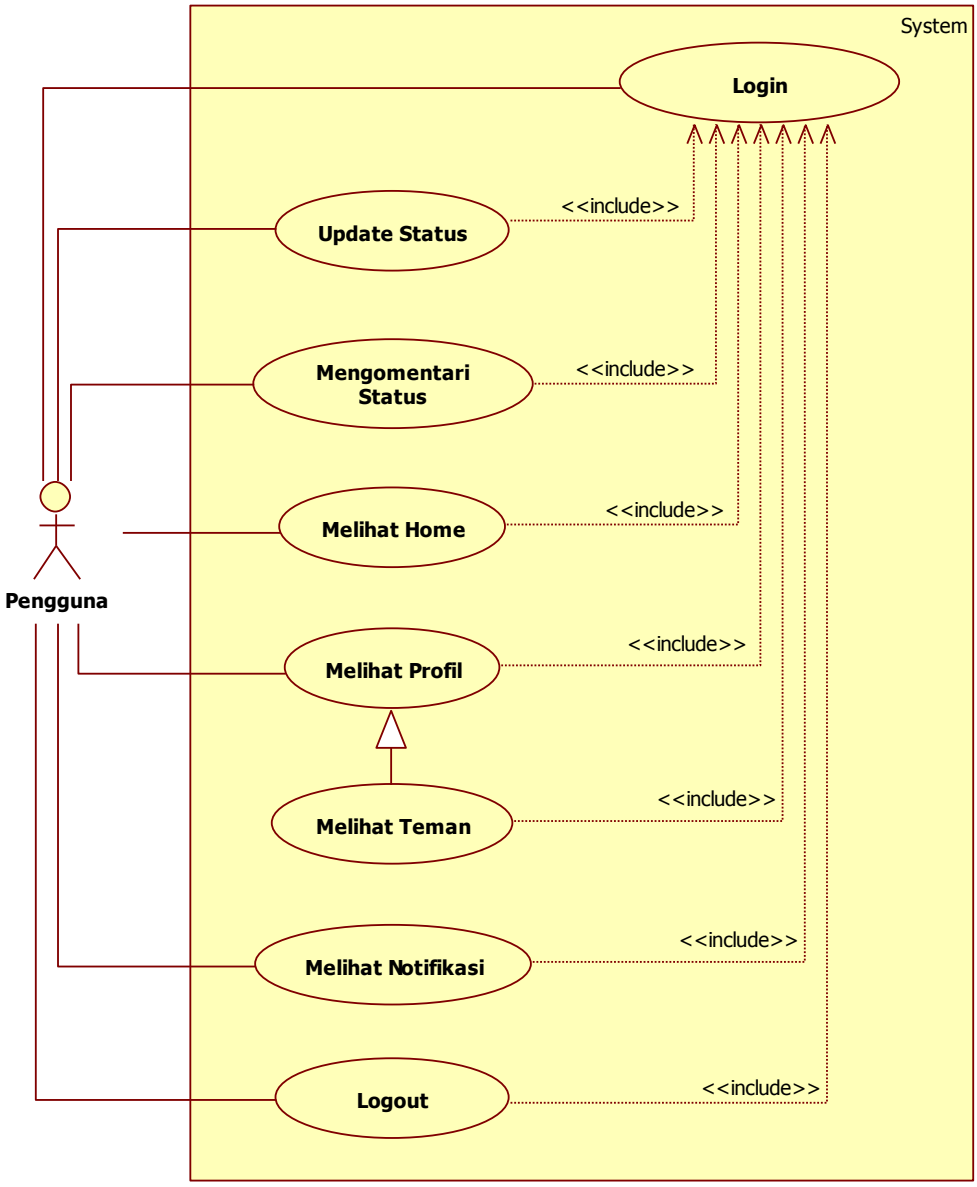
Dari hasil perbandingan sistem operasi iOS, Blackberry, Symbian, dan Android maka dapat disimpulkan bahwa Android memiliki perangkat keras dan fitur sistem operasi yang lebih unggul dibanding sistem operasi yang lainnya. Tidak hanya dari segi perangkat keras dan fitur tetapi dari segi harga pun Android memberikan harga yang lebih terjangkau. Untuk itu diharapkan dengan pemilihan Android sebagai sistem operasi dalam pengembangan aplikasi *mobile* Kampus *Online* dapat memenuhi kebutuhan pengguna dalam mengakses Kampus *Online*, Nilai *Online*, dan Autodebet *Online*.

3. 1. 3. Analisis Sistem yang Sedang Berjalan

Analisis sistem yang berjalan berisi tentang pemaparan prosedur-prosedur yang sedang berjalan saat ini. Analisis ini dimaksudkan agar perangkat lunak yang dibangun tidak keluar dari cakupan sistem yang ada. Adapun analisis dari sistem yang sedang berjalan saat ini terdiri dari analisis penggunaan jejaring sosial Kampus *Online* Unikom, analisis penggunaan Nilai *Online*, dan analisis penggunaan Autodebet *Online*.

3. 1. 3. 1. **Analisis Kampus Online Unikom**

Analisis Kampus *Online* Unikom memaparkan proses dari setiap prosedur yang berjalan saat ini pada jejaring sosial Kampus *Online* Unikom. Analisis penggunaan jejaring sosial Kampus *Online* Unikom terdiri dari analisis prosedur *login*, prosedur *update* status, prosedur mengomentari status, prosedur melihat daftar pertemanan, prosedur melihat profil pribadi, prosedur melihat halaman *home* (halaman utama), prosedur melihat data notifikasi, dan prosedur *logout.* Adapun proses dari setiap prosedur yang terdapat pada jejaring sosial Kampus *Online* digambarkan dengan menggunakan *usecase diagram* yang terlihat pada Gambar III.3.



Gambar III.3 *Usecase Diagram* Kampus *Online*

Setiap proses yang terdapat pada *use case diagram* Kampus *Online* (Gambar III.3) dapat dipaparkan lebih detail dengan menggunakan *activity diagram* yang terdiri dari *activity diagram login*, *activity diagram update status*, *activity diagram* mengomentari status, *activity diagram* melihat *home*, *activity diagram* melihat profil, *activity diagram* melihat teman, *activity diagram* melihat

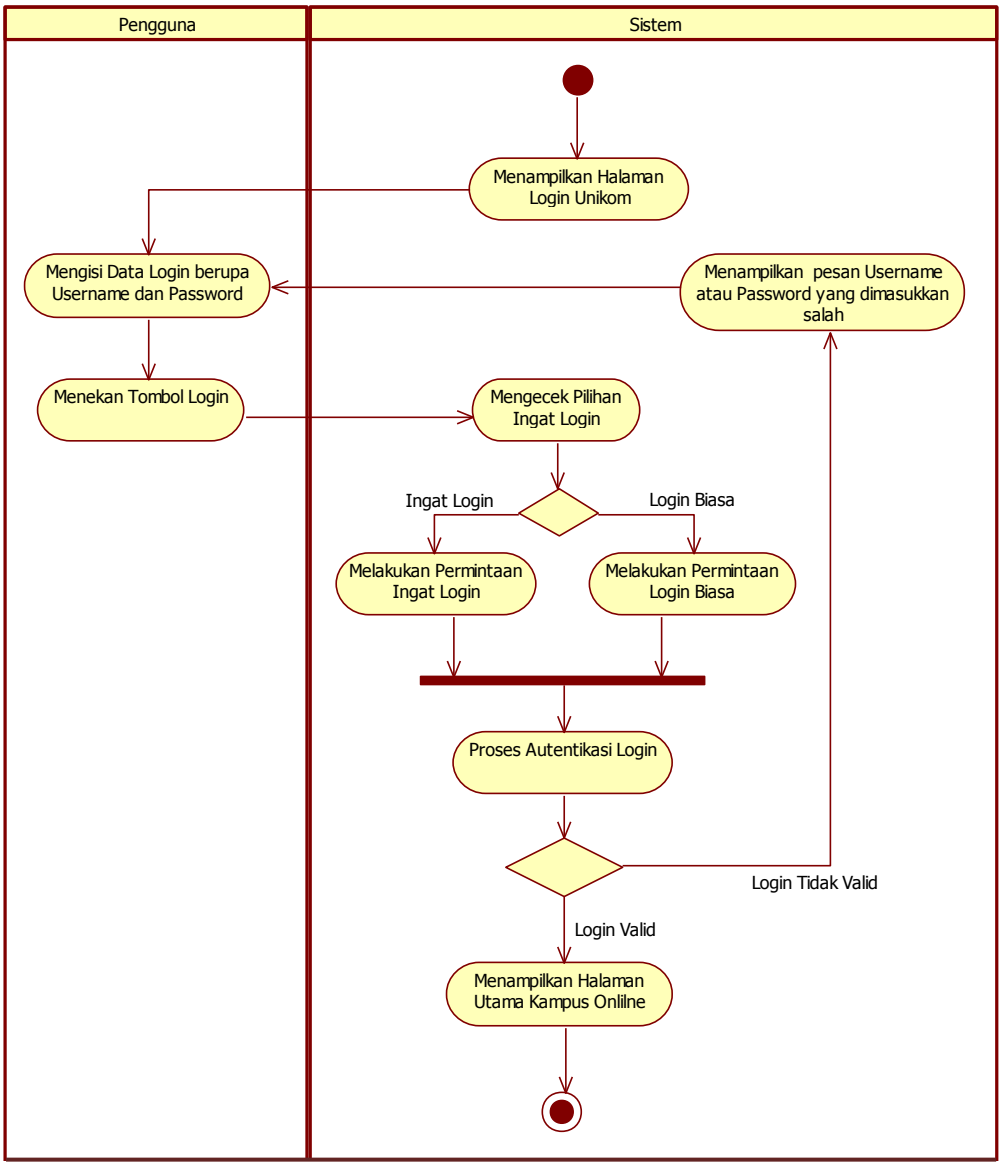
notifikasi, dan *activity diagram logout*: Adapun pemaparan dari setiap *activity diagram* yang terdapat pada *use case diagram* Kampus *Online* (Gambar III.3) adalah sebagai berikut:

1. Prosedur *Login*

Prosedur *login* menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna untuk mendapatkan hak akses masuk ke dalam jejaring sosial Kampus *Online* Unikom. Adapun prosedur *login* yang terdapat pada jejaring sosial Kampus *Online* Unikom adalah sebagai berikut:

1. Sistem menampilkan halaman *login*.
2. Pengguna mengisi data *login* berupa *username* dan *password*. Selain itu pengguna juga dapat memilih pilihan ingat *login*. Bila dipilih maka pengguna akan selalu *login* secara otomatis selama dua minggu pada perangkat yang sama, bila tidak maka waktu login hanya dua jam.
3. Setelah data *login* diisi kemudian pengguna menekan tombol *login.*
4. Sistem melakukan proses autentikasi *login* yaitu menyamakan data yang dimasukkan oleh pengguna dengan data yang terdapat pada basis data. Tidak ada proses validasi data *username* dan *password* walaupun data tersebut dikosongkan ketika pengguna menekan tombol *login*.
5. Jika data *login* benar (sesuai dengan data yang terdapat pada basis data) maka sistem akan menampilkan halaman *loading* sebelum akhirnya sistem akan menampilkan halaman *home* (halaman utama) jejaring sosial Kampus *Online*.

Berikut prosedur *login* dalam bentuk *activity diagram* yang terlihat pada Gambar III.4.



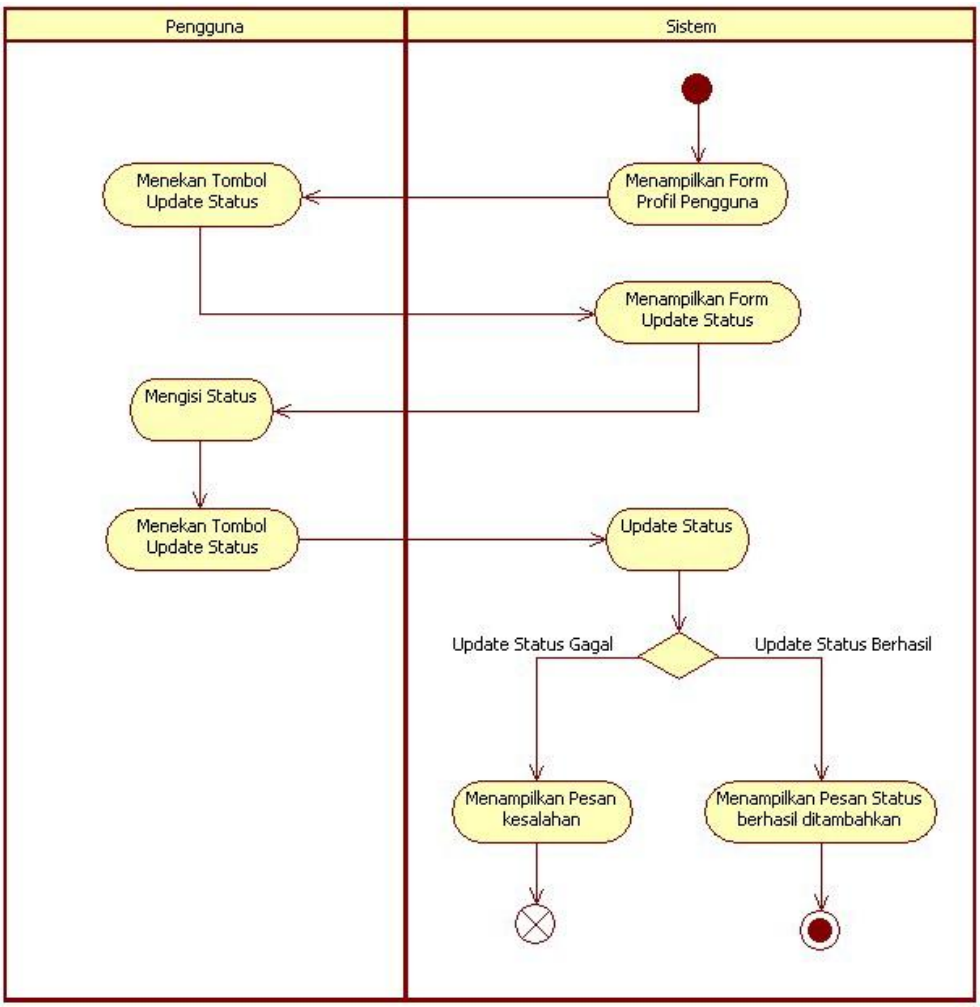
Gambar III.4 *Activity Diagram Login* Kampus *Online*

2. Prosedur *Update* Status

Prosedur *update status* merupakan tahapan-tahapan yang dilakukan pada saat pengguna akan menambah status baru pada jejaring sosial Kampus *Online* Unikom. Adapun prosedur *update* status yang terdapat pada jejaring sosial Kampus *Online* Unikom adalah:

1. Sistem menampilkan halaman utama.
2. Pengguna mengisi data status pada kotak isian *update* status yang terdapat pada bagian atas halaman utama.
3. Pengguna dapat menambahkan *emoticon* dengan memilih tombol *smile* pada bagian kiri bawah kotak isian status. Selain itu pengguna juga dapat melakukan sapaan kepada teman dengan memilih tombol @ di bagian kiri bawah kotak isian status. Kemudian sistem akan menampilkan kotak isian yang digunakan oleh pengguna untuk mengetikkan nama teman yang akan disapa. Jika terdapat kesahan pengetikkan *emoticon* dan sapaan terhadap teman maka sistem akan menganggap kesalahan pengetikkan tersebut sebagai sebagai karakter biasa pada status.
4. *Setelah status selesai diisi kemudian pengguna menekan tombol update status.*
5. Sistem akan menyimpan data status yang telah diisi kemudian menampilkan status tersebut di baris utama status pada halaman *home* (halaman utama).

Berikut prosedur *update status* dalam bentuk *activity diagram* yang terlihat pada Gambar III.5.



Gambar III.5 *Activity Diagram Update* Status Kampus *Online*

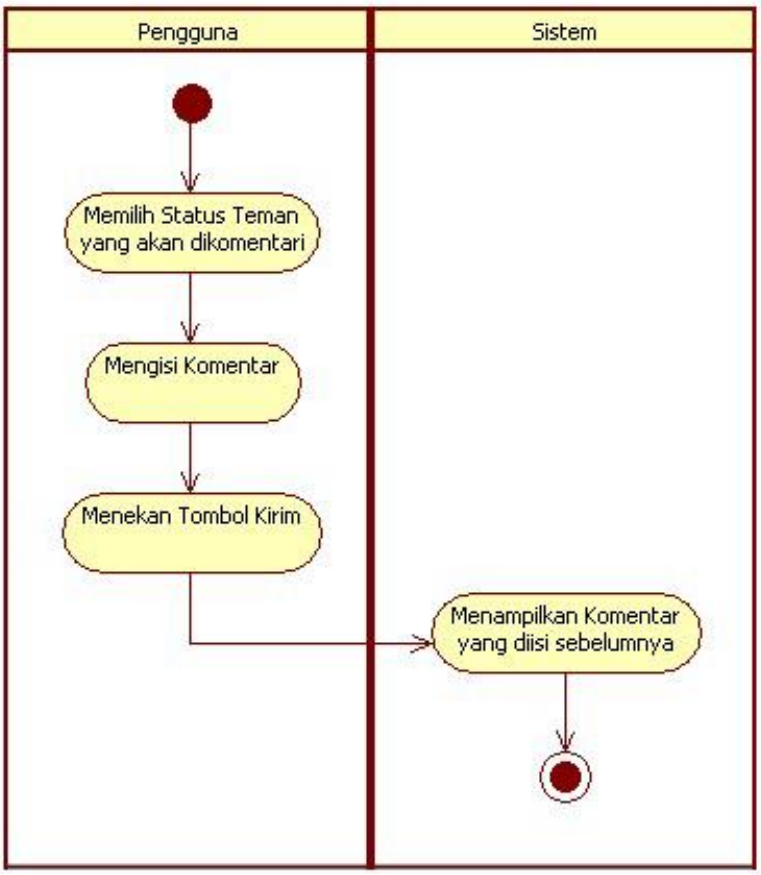
3. Prosedur Mengomentari Status

Prosedur mengomentari status adalah langkah-langkah yang dilakukan pengguna ketika ingin memberi komentar pada status baik itu status pribadi maupun status teman. Adapun prosedur mengomentari status yang terdapat pada jejaring sosial Kampus *Online* Unikom adalah:

1. Pengguna memilih status yang akan dikomentari.
2. Pengguna mengetik komentar di kotak isian komentar pada bagian bawah status yang akan dikomentari.

3. Pengguna dapat menambahkan *emoticon* dengan memilih tombol *smile* pada bagian kiri bawah kotak isian komentar. Selain itu pengguna juga dapat melakukan sapaan kepada teman dengan memilih tombol @ di bagian kiri bawah kotak isian status. Kemudian sistem akan menampilkan kotak isian yang digunakan oleh pengguna untuk mengetikkan nama teman yang akan disapa. Jika terdapat kesahan pengetikkan *emoticon* dan sapaan terhadap teman maka sistem akan menganggap kesalahan pengetikkan tersebut sebagai sebagai karakter biasa pada status.
4. *Setelah komentar selesai diisi kemudian pengguna menekan tombol kirim.*
5. Sistem akan menyimpan data komentar kemudian menampilkan komentar tersebut di baris terbawah pada status yang telah dikomentari.

Berikut prosedur mengomentari status dalam bentuk *activity diagram* yang terlihat pada Gambar III.6.



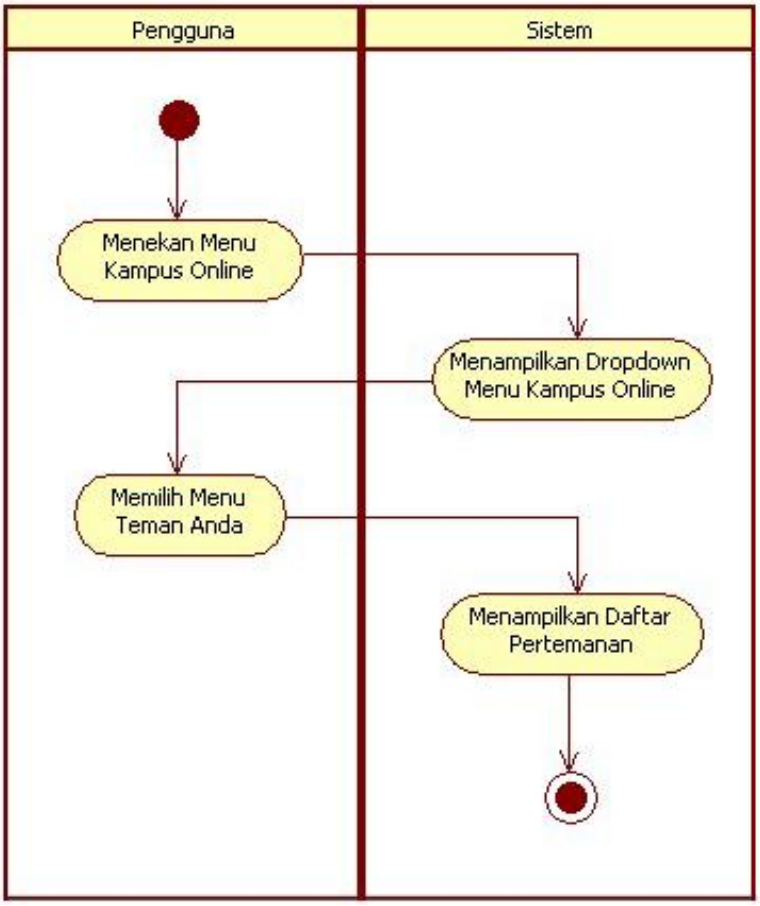
Gambar III.6 *Activity Diagram* Mengomentari Status Kampus *Online*

4. Prosedur Melihat Daftar Pertemanan

Prosedur melihat daftar pertemanan merupakan langkah-langkah untuk melihat daftar pertemanan dari pengguna. Adapun prosedur melihat daftar pertemanan yang terdapat pada jejaring sosial Kampus *Online* Unikom adalah:

1. Pengguna menekan menu Kampus Unikom yang terdapat dibagian sudut kiri atas.
2. Sistem akan menampilkan menu *dropdown* Kampus Unikom.
3. Selanjutnya pengguna memilih menu Teman Anda
4. Sistem akan menampilkan daftar pertemanan.

Berikut prosedur melihat daftar pertemanan dalam bentuk *activity diagram* yang terlihat pada Gambar III.7.



Gambar III.7 *Activity Diagram* Melihat Daftar Pertemanan Kampus *Online*

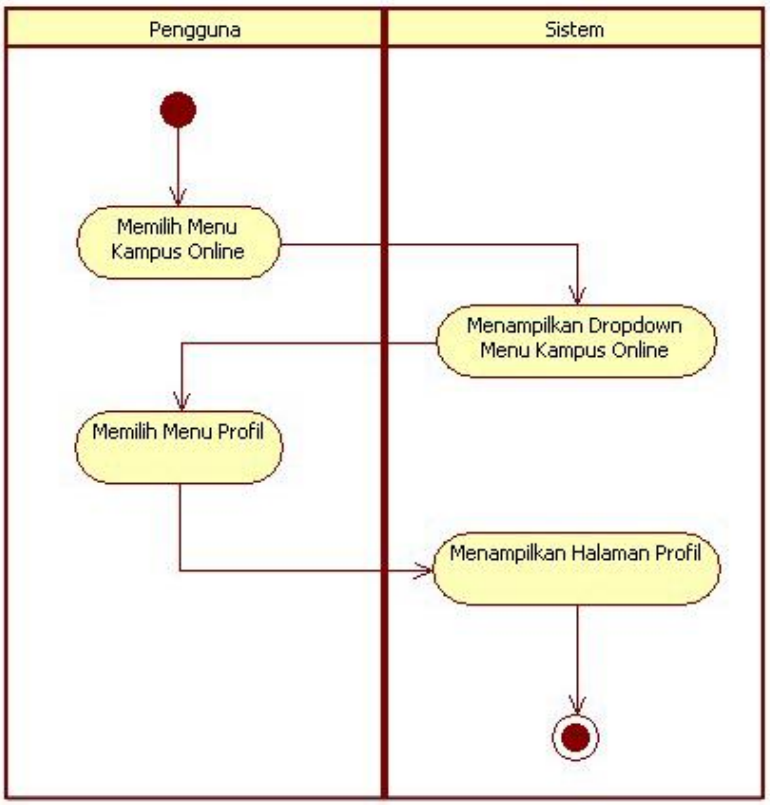
5. Prosedur Melihat Profil Pribadi

Prosedur melihat profil pribadi merupakan langkah-langkah untuk melihat halaman profil pengguna. Adapun prosedur melihat profil pribadi yang terdapat pada jejaring sosial Kampus *Online* Unikom adalah:

1. Pengguna menekan menu Kampus Unikom yang terdapat dibagian sudut kiri atas.
2. Sistem akan menampilkan menu *dropdown* Kampus Unikom.

3. Selanjutnya pengguna memilih menu Profil Anda.
4. Sistem akan menampilkan halaman profil pribadi pengguna.

Berikut prosedur melihat profil pribadi dalam bentuk *activity diagram* yang terlihat pada Gambar III.8.



Gambar III.8 *Activity Diagram* Melihat Profil Pribadi Kampus *Online*

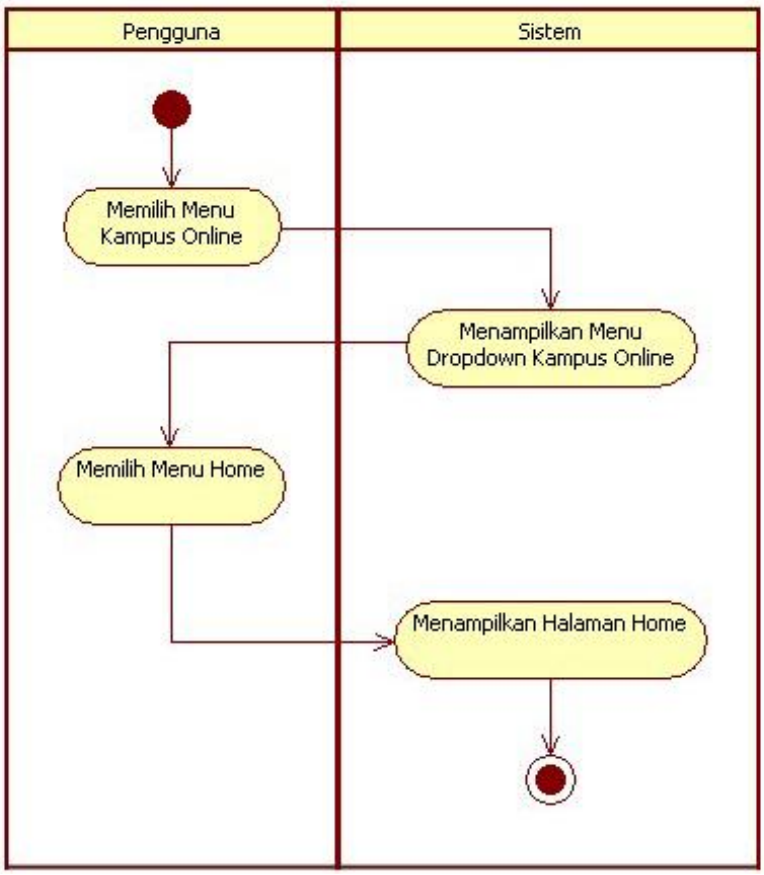
6. Prosedur Melihat Halaman *Home*

Prosedur melihat halaman *home* merupakan langkah-langkah untuk melihat halaman utama Kampus *Online* Unikom. Adapun prosedur melihat halaman *home* yang terdapat pada jejaring sosial Kampus *Online* Unikom adalah:

1. Pengguna menekan menu Kampus Unikom yang terdapat dibagian sudut kiri atas.
2. Sistem akan menampilkan menu *dropdown* Kampus Unikom.

3. Selanjutnya pengguna memilih menu *Home*.
4. Sistem akan menampilkan halaman home (halaman utama).

Berikut prosedur melihat halaman *home* dalam bentuk *activity diagram* yang terlihat pada Gambar III.9.



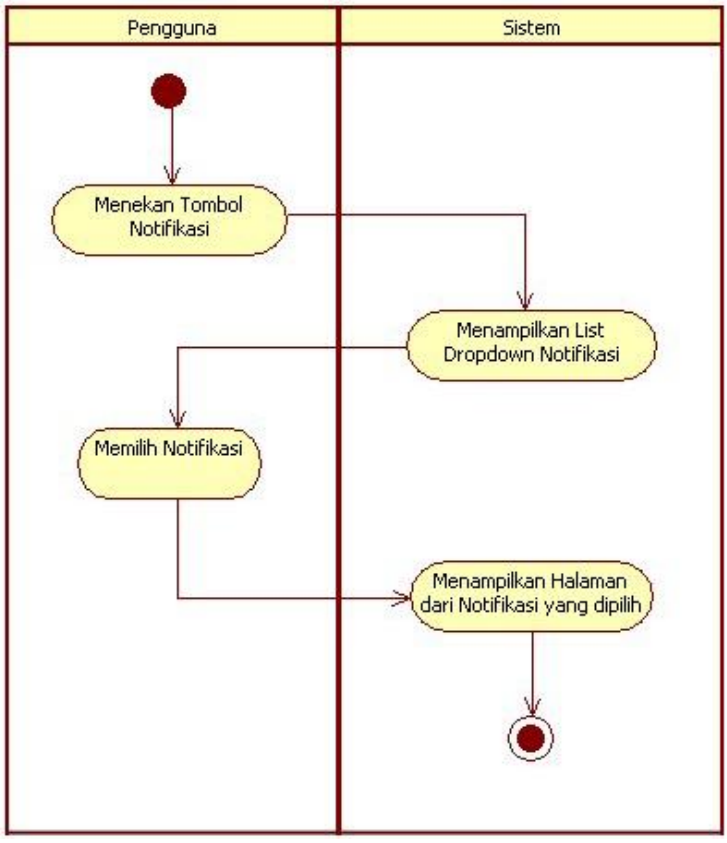
Gambar III.9 *Activity Diagram* Halaman *Home* Kampus *Online*

7. Prosedur Melihat Data Notifikasi

Prosedur melihat data notifikasi merupakan langkah-langkah untuk melihat data notifikasi dari aktivitas yang berkaitan dengan pengguna. Adapun prosedur melihat data notifikasi yang terdapat pada jejaring sosial Kampus *Online* Unikom adalah:

1. Pengguna menekan tombol notifikasi pada *menu bar* Kampus Unikom yang terdapat dibagian kanan atas.
2. Sistem akan menampilkan *dropdown list* notifikasi.
3. Selanjutnya pengguna memilih *list item* yang terdapat pada *dropdown* notifikasi.
4. Sistem akan menampilkan halaman dari notifikasi yang dipilih pengguna.

Berikut prosedur melihat data notifikasi dalam bentuk *activity diagram* yang terlihat pada Gambar III.10.



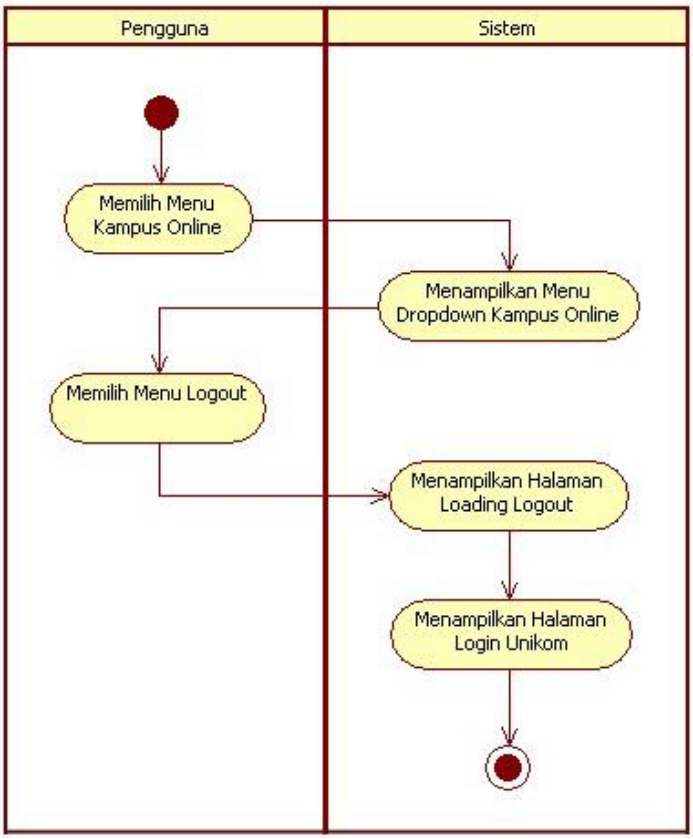
Gambar III.10 *Activity Diagram* Melihat Data Notifikasi Kampus *Online*

8. Prosedur *Logout*

Prosedur *logout* menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna untuk keluar dari jejaring sosial Kampus *Online* Unikom. Adapun prosedur *logout* yang terdapat pada jejaring sosial Kampus *Online* Unikom adalah:

1. Pengguna menekan menu Kampus Unikom yang terdapat dibagian sudut kiri atas.
2. Sistem akan menampilkan menu *dropdown* Kampus Unikom.
3. Selanjutnya pengguna memilih menu *Logout*.
4. Sistem akan menampilkan halaman *loading* sebelum akhirnya menampilkan halaman *login* jejaring sosial Kampus *Online*.

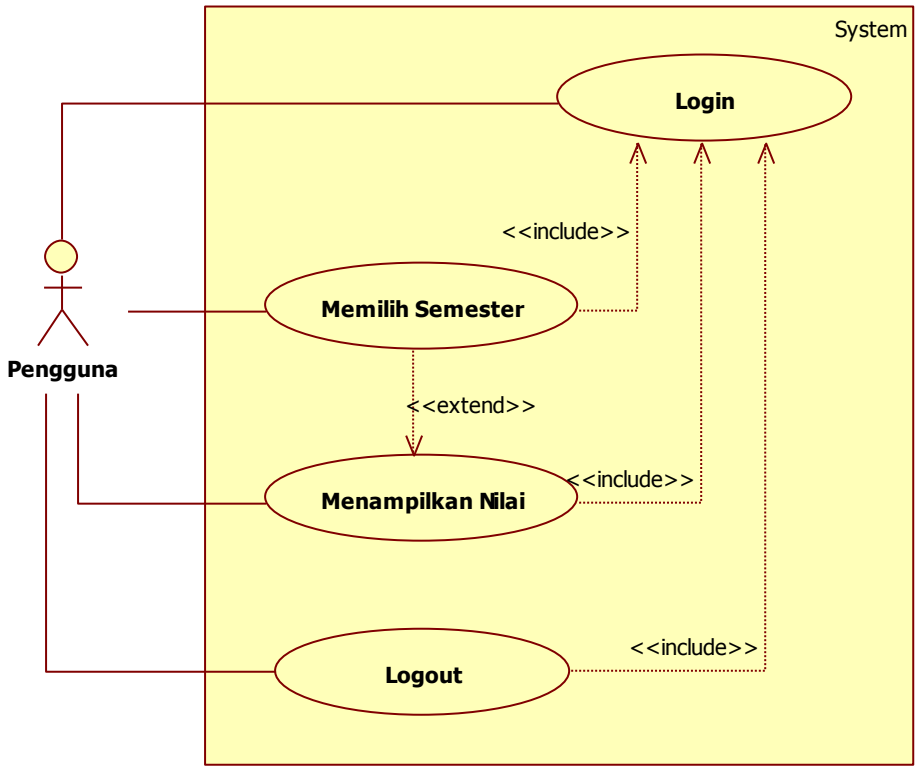
Berikut prosedur *logout* dalam bentuk *activity diagram* yang terlihat pada Gambar III.11.



Gambar III.11 *Activity Diagram Logout* Kampus *Online*

3. 1. 3. 2. **Analisis Nilai *Online***

Analisis nilai *online* memaparkan proses dari setiap prosedur penggunaan fasilitias akademik Nilai *Online*. Analisis penggunaan Nilai Unikom terdiri dari analisis prosedur *login*, analisis prosedur melihat daftar nilai, analisis memilih semester, dan analisis prosedur *logout*. Adapun proses dari setiap prosedur yang terdapat pada Nilai *Online* digambarkan dengan menggunakan *usecase diagram* yang terlihat pada Gambar III.12.



Gambar III.12 *Usecase Diagram* Nilai *Online*

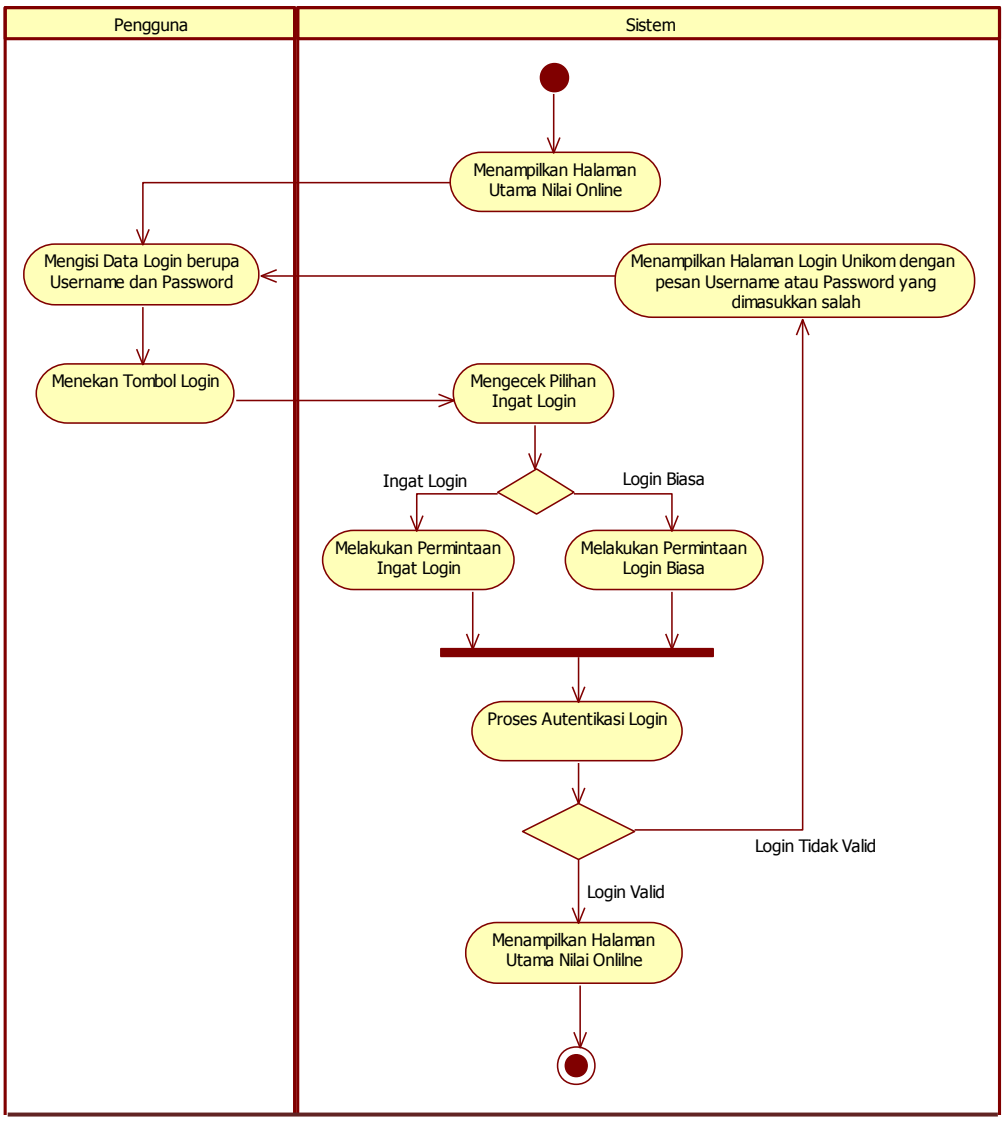
Setiap proses yang terdapat pada *use case diagram* Nilai *Online* (Gambar III.12) dapat dipaparkan lebih detail dengan menggunakan *activity diagram* yang terdiri dari *activity diagram login*, *activity diagram* memilih semester, *activity diagram* menampilkan nilai, dan *activity diagram logout*. Adapun pemaparan dari setiap *activity diagram* yang terdapat pada *use case diagram* Nilai *Online* (Gambar III.12) adalah sebagai berikut:

1. Prosedur *Login*

Prosedur *login* menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna untuk mendapatkan hak akses masuk ke dalam Nilai *Online*. Adapun prosedur *login* yang terdapat pada Nilai *Online* adalah sebagai berikut:

1. Sistem menampilkan halaman *login*.
2. Pengguna mengisi data *login* berupa *username*, *password*, dan pilihan ingat *login*. Bila ingat *login* dipilih maka pengguna akan selalu *login* secara otomatis selama dua minggu pada komputer yang sama, bila tidak maka waktu login hanya dua jam.
3. Setelah data *login* diisi kemudian pengguna menekan tombol *login.*
4. Sistem mengecek apakah ingat login dipilih atau tidak. Hal tersebut berpengaruh terhadap waktu login yang telah dijelaskan sebelumnya pada poin kedua.
5. Sistem melakukan proses autentikasi *login* yaitu menyamakan data yang dimasukkan oleh pengguna dengan data yang terdapat pada basis data. Tidak ada proses validasi data *username* dan *password* walaupun data tersebut dikosongkan ketika pengguna menekan tombol *login*.
6. Jika data *login* maka sistem akan menampilkan halaman utama Nilai *Online* dan jika *login* salah maka sistem akan menampilkan halamam *login* Unikom dengan pesan *username* atau *password* yang dimasukkan salah.

Berikut prosedur *login* Nilai *Online* dalam bentuk *activity diagram* yang terlihat pada Gambar III.13.



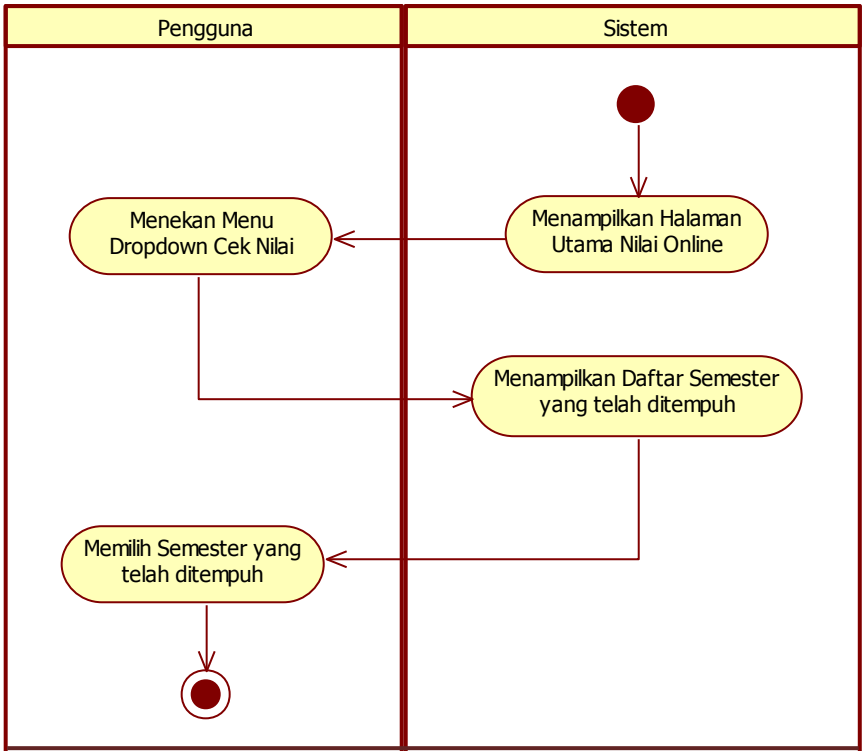
Gambar III.13 *Activity Diagram Login* Nilai *Online*

2. Prosedur Memilih Semester

Prosedur memilih semester menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna pada saat memilih semester yang telah ditempuh pada fasilitas Nilai *Online*. Adapun prosedur memilih semester yang terdapat pada Nilai *Online* adalah sebagai berikut:

1. Sistem menampilkan halaman utama Nilai Online.
2. Pengguna menekan menu *dropdown* cek nilai yang terletak pada bagian atas halaman Nilai *Online*.
3. Sistem akan menampilkan daftar semester yang telah ditempuh oleh mahasiswa yang bersangkutan.
4. Pengguna memilih semester yang telah ditempuh pada menu *dropdown* cek nilai.

Berikut prosedur memilih semester pada fasilitas Nilai *Online* dalam bentuk *activity diagram* yang terlihat pada Gambar III.14.



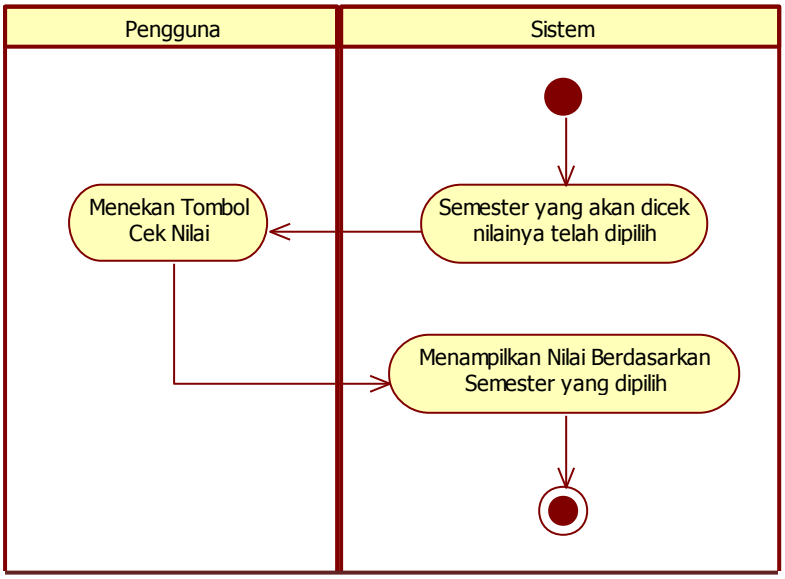
Gambar III.14 *Activity Diagram* Memilih Semester Nilai *Online*

3. Prosedur Menampilkan Nilai

Prosedur menampilkan nilai menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna pada saat menampilkan nilai pada fasilitas Nilai *Online*. Adapun prosedur menampilkan nilai yang terdapat pada Nilai *Online* adalah sebagai berikut:

1. Pengguna telah memilih semester yang nilainya akan dicek.
2. Setelah semester dipilih lalu pengguna menekan tombol cek nilai.
3. Sistem akan menampilkan nilai berdasarkan semester yang dipilih.

Berikut prosedur menampilkan nilai pada fasilitas Nilai *Online* dalam bentuk *activity diagram* yang terlihat pada Gambar III.15.



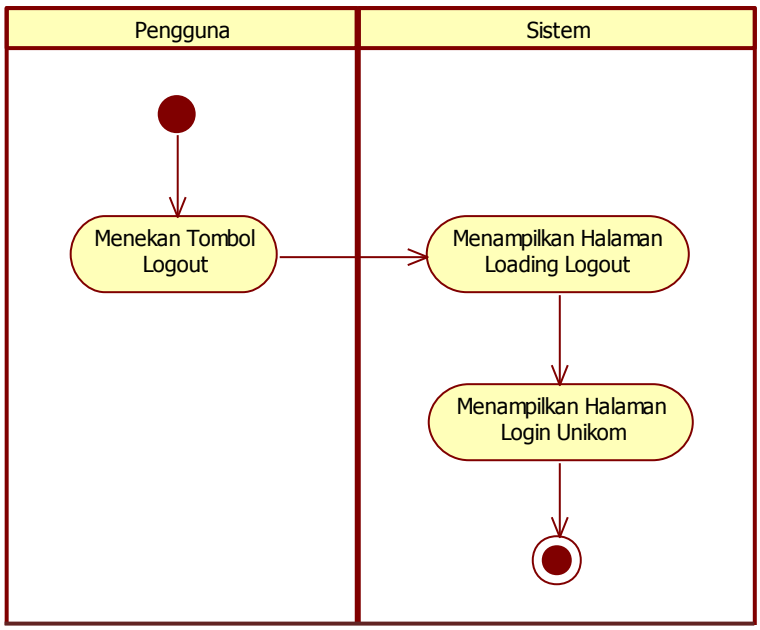
Gambar III.15 *Activity Diagram* Menampilkan Daftar Nilai *Online*

4. Prosedur *Logout*

Prosedur *logout* menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna untuk keluar dari fasilitas Nilai *Online*. Adapun prosedur *logout* yang terdapat pada fasilitas Nilai *Online* adalah sebagai berikut:

1. Pengguna menekan *link logout* pada bagian atas halaman Nilai *Online*.
2. Sistem akan menampilkan halaman *loading* sebelum akhirnya menampilkan halaman *login* Unikom.

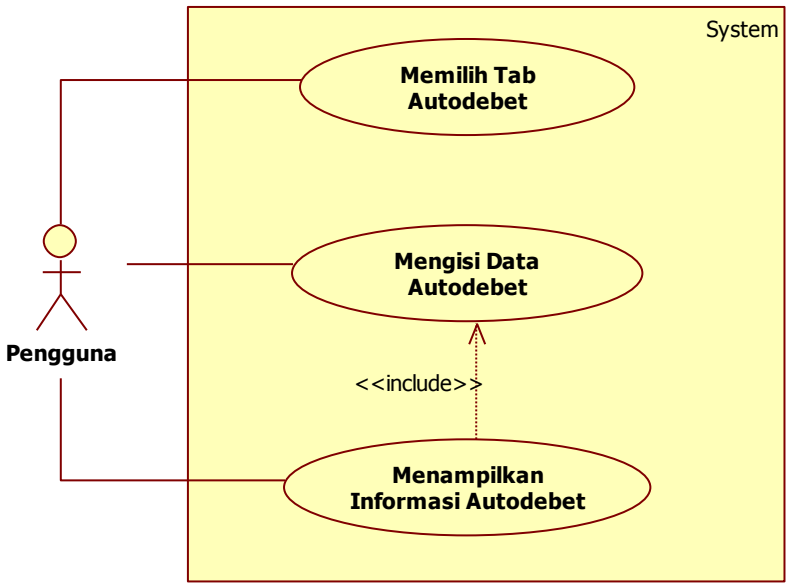
Berikut prosedur *logout* dalam bentuk *activity diagram* yang terlihat pada Gambar III.16.



Gambar III.16 *Activity Diagram Logout* Kampus *Online*

3. 1. 3. 3. **Analisis Autodebet *Online***

Analisis autodebet *online* memaparkan proses dari setiap prosedur penggunaan fasilitias akademik Autodebet *Online*. Analisis penggunaan Autodebet Unikom terdiri dari analisis prosedur memilih tab autodebet, analisis prosedur mengisi data autodebet, dan analisis menampilkan informasi autodebet. Adapun proses dari setiap prosedur yang terdapat pada Autodebet *Online* digambarkan dengan menggunakan *usecase diagram* yang terlihat pada Gambar III.17.



Gambar III.17 *Usecase Diagram* Autodebet *Online*

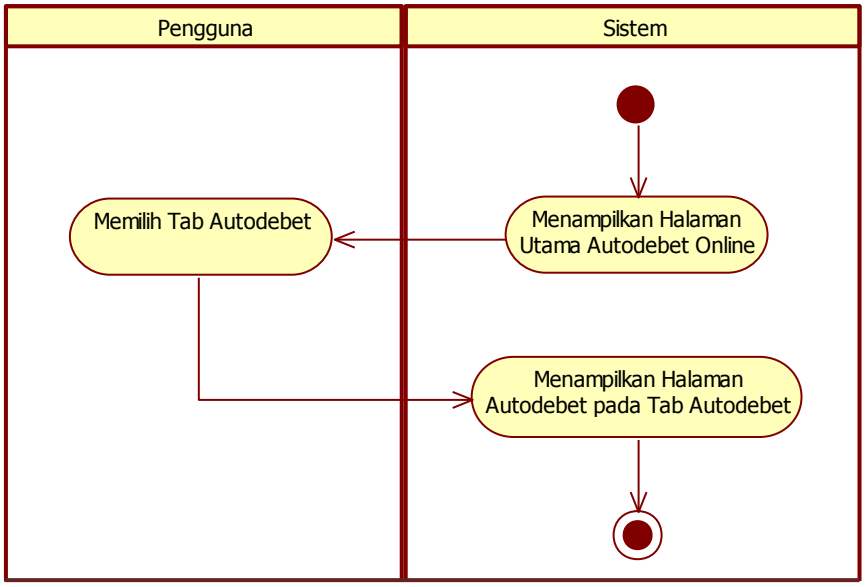
Setiap proses yang terdapat pada *use case diagram* Autodebet *Online* (Gambar III.17) dapat dipaparkan lebih detail dengan menggunakan *activity diagram* yang terdiri dari *activity diagram* memilih tab autodebet, *activity diagram* mengisi data autodebet, dan *activity diagram* menampilkan informasi autodebet. Adapun pemaparan dari setiap *activity diagram* yang terdapat pada *use case diagram* Autodebet *Online* (Gambar III.17) adalah sebagai berikut:

1. Prosedur Memilih Tab Autodebet

Prosedur memilih tab autodebet menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna pada saat memilih tab autodebet pada situs Autodebet *Online*. Adapun prosedur memilih tab autodebet yang terdapat pada fasilitas Autodebet *Online* adalah sebagai berikut:

1. Sistem menampilkan halaman utama Autodebet *Online*.
2. Pengguna memilih tab autodebet pada bagian atas halaman Autodebet *Online*.
3. Sistem akan menampilkan halaman Autodebet Online pada tab autodebet.

Berikut prosedur memilih tab autodebet pada fasilitas Autodebet *Online* dalam bentuk *activity diagram* yang terlihat pada Gambar III.18.



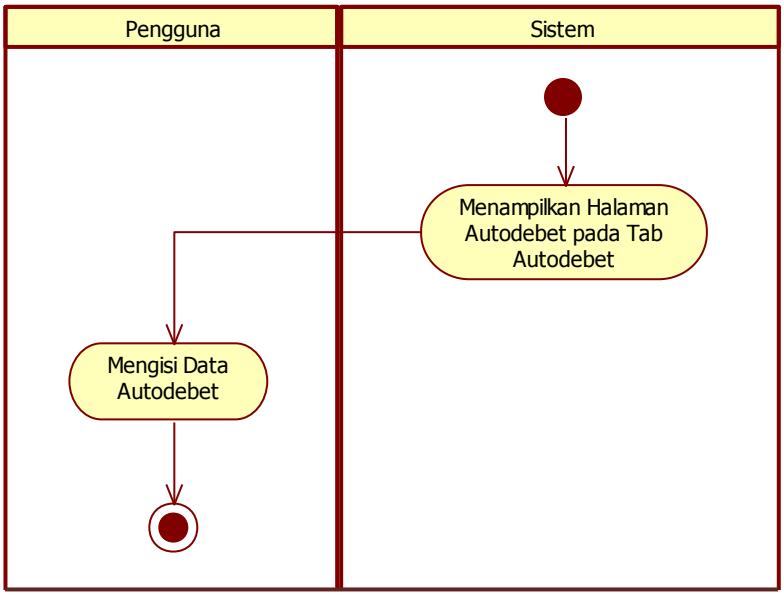
Gambar III.18 *Activity Diagram* Memilih Tab Autodebet *Online*

2. Prosedur Mengisi Data Autodebet

Prosedur mengisi data autodebet menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna pada saat mengisi data autodebet pada fasilitas Autodebet *Online*. Adapun prosedur mengisi data autodebet yang terdapat pada fasilitas Autodebet *Online* adalah sebagai berikut:

1. Sistem menampilkan halaman Autodebet *Online* pada tab Autodebet.
2. Pengguna mengisi data autodebet yang akan ditampilkan.

Berikut prosedur mengisi data autodebet pada fasilitas Autodebet *Online* dalam bentuk *activity diagram* yang terlihat pada Gambar III.19.



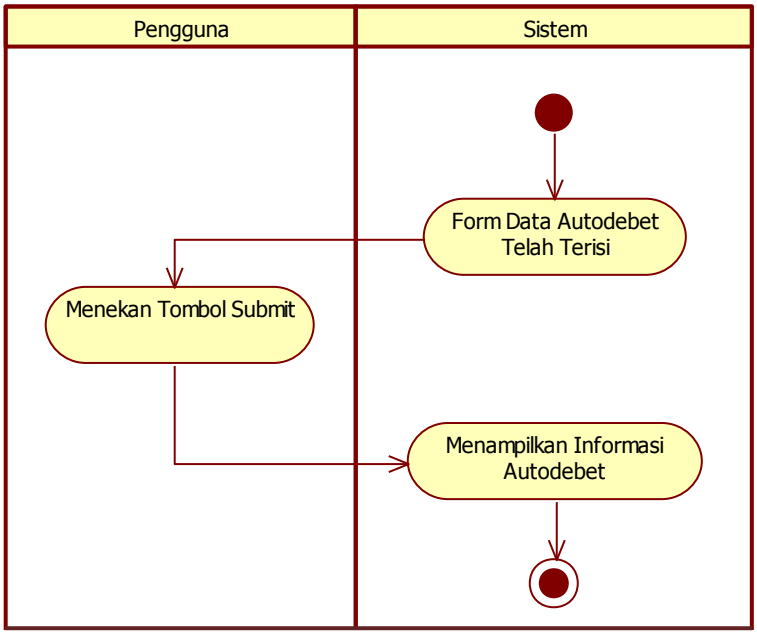
Gambar III.19 *Activity Diagram* Mengisi Data Autodebet *Online*

3. Prosedur Menampilkan Informasi Autodebet

Prosedur menampilkan informasi autodebet menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna untuk menampilkan informasi autodebet pada fasilitas Autodebet *Online*. Adapun prosedur menampilkan informasi autodebet yang terdapat pada fasilitas Autodebet *Online* adalah sebagai berikut:

1. Data autodebet pada tab autodebet telah terisi sesuai dengan informasi autodebet yang akan ditampilkan.
2. Pengguna menekan tombol submit.
3. Sistem menampilkan informasi autodebet sesuai data autodebet yang dimasukkan sebelumnya

Berikut prosedur menampilkan informasi autodebet pada fasilitas Autodebet *Online* dalam bentuk *activity diagram* yang terlihat pada Gambar III.20.



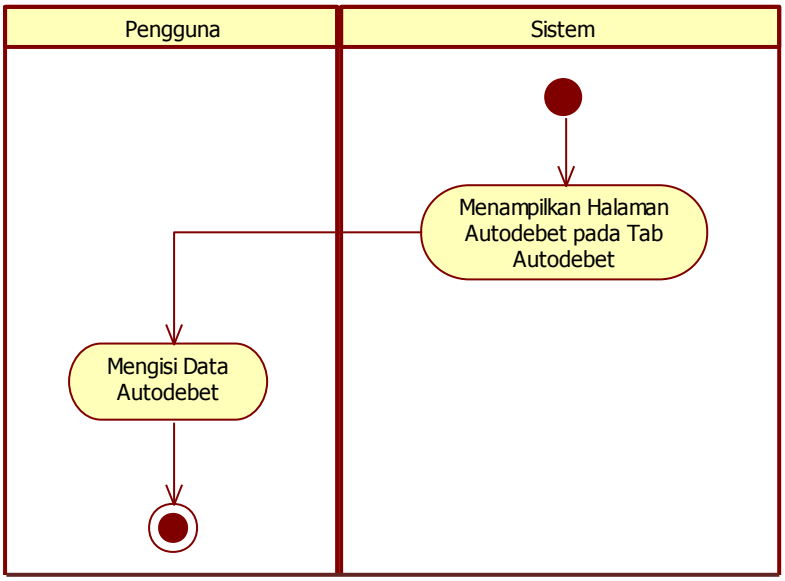
Gambar III.20 *Activity Diagram* Menampilkan Informasi Autodebet *Online*

4. Prosedur Mengisi Data Autodebet

Prosedur mengisi data autodebet menjelaskan tentang tahapan-tahapan yang dilakukan oleh pengguna pada saat mengisi data autodebet pada fasilitas Autodebet *Online*. Adapun prosedur mengisi data autodebet yang terdapat pada fasilitas Autodebet *Online* adalah sebagai berikut:

1. Sistem menampilkan halaman Autodebet *Online* pada tab Autodebet.
2. Pengguna mengisi data autodebet yang akan ditampilkan.

Berikut prosedur mengisi data autodebet pada fasilitas Autodebet *Online* dalam bentuk *activity diagram* yang terlihat pada Gambar III.21.



Gambar III.21 *Activity Diagram* Mengisi Data Autodebet *Online*

3. 1. 4. Analisis Kebutuhan Data

Analisis kebutuhan data menggambarkan berbagai data yang digunakan pada aplikasi *mobile* Kampus *Online* Unikom. Adapun data yang digunakan pada aplikasi *mobile* Kampus *Online* Unikom terlihat pada Tabel III.4.

Tabel III.4 Kebutuhan Data Aplikasi *Mobile* Kampus *Online* Unikom

No	Nama Data	Fungsi
1	Data *Session*	Berisi informasi status *session* yang sedang berjalan.
2	Data Pengguna	Berisi informasi data pengguna.
3	Data Status	Berisi informasi status-status terbaru, *mention*, dan sapaan kepada teman.
4	Data Notifikasi	Berisi informasi pemberitahuan bagi pengguna meliputi pemberitahuan komentar, *rating* bagus, *rating* jelek, sapaan, mention, dan teman.
5	Data Komentar	Berisi informasi data komentar dari status.
6	Data *Rating*	Berisi informasi data *rating* dari suatu status yang terdiri dari *rating* bagus dan *rating* jelek.
7	Data Pertemanan	Berisi informasi pertemanan pengguna yang terdiri dari *following* (pengguna mengikuti status pengguna lain), *followers* (pengguna lain mengikuti status pengguna yang dimaksud), dan *friends* (pengguna saling mengikuti status masing-masing).
8	Autodebet	Berisi informasi mengenai data hasil autodebet biaya kuliah mahasiswa yang dibayarkan setiap semesternya.
9	Nilai	Berisi informasi data nilai akhir mahasiswa dari semester yang telah ditempuh

3. 1. 5. Analisis Basis Data

Analisis basis data menggambarkan basis data yang ada saat ini. Basis data yang ada saat ini terdiri dari beberapa basis data yang terpisah satu sama lain. Hal tersebut terjadi dikarenakan setiap pengembangan fasilitas baru dibuat basis data baru yang terpisah dengan basis data yang ada sebelumnya. Adapun analisis yang

dilakukan pada basis data yang ada saat ini adalah analisis pada basis data Kampus *Online*, analisis pada basis data Nilai *Online*, dan analisis pada basis data Autodebet *Online*.

3. 1. 5. 1. **Analisis Basis Data Kampus *Online***

Analisis basis data Kampus *Online* memaparkan tabel-tabel pada basis data yang saat ini digunakan dalam fasilitas Kampus *Online*. Adapun tabel-tabel pada basis data Kampus *Online* saat ini dapat dilihat pada Tabel III.5.

Tabel III.5 Analisis Basis Data Kampus *Online*

No	Nama Tabel	Deskripsi
1	Session	Berisi waktu dan data pengguna yang login
2	Mahasiswa	Berisi data mahasiswa yang telah melakukan aktivasi ke dalam Kampus *Online* baik yang telah menjadi alumni maupun yang masih berstatus mahasiswa
3	Karyawan	Berisi data karyawan yang telah melakukan aktivasi ke dalam Kampus *Online*
4	Dosen	Berisi data dosen yang telah melakukan aktivasi ke dalam Kampus *Online*
5	Pengguna	Berisi data pengguna Kampus *Online* yang terdiri dari data mahasiswa, dosen, karyawan, dan alumni
6	Status	Berisi data status-status dari pengguna
7	Notifikasi	Berisi data pemberitahuan bagi pengguna meliputi pemberitahuan komentar, *rating* bagus, *rating* jelek, sapaan, mention, dan teman.
8	Komentar	Berisi data komentar dari status.
9	*Rating*	Berisi data *rating* dari suatu status yang terdiri dari *rating* bagus dan *rating* jelek.
10	*Friends*	Berisi data pertemanan pengguna yang terdiri dari *following* (pengguna mengikuti status pengguna lain), *followers* (pengguna lain mengikuti status pengguna yang dimaksud), dan *friends* (pengguna saling mengikuti status masing-masing).

3. 1. 5. 2. **Analisis Basis Data Nilai *Online***

Analisis basis data Nilai *Online* memaparkan tabel-tabel pada basis data yang saat ini digunakan dalam fasilitas Nilai *Online*. Adapun tabel-tabel pada basis data Nilai *Online* saat ini dapat diliahat pada Tabel III.6.

Tabel III.6 Analisis Basis Data Nilai *Online*

No	Nama Data	Fungsi
1	Tahun Akademik	Berisi data tahun akademik yang telah ditempuh oleh mahasiswa.
2	Fakultas	Berisi data fakultas yang terdapat di Unikom
3	Jurusan	Berisi data jurusan yang terdapat di Unikom
4	Mahasiswa	Berisi data mahasiswa yang masih aktif
5	KRS	Berisi data KRS yang dimiliki oleh mahasiswa
6	Detail KRS	Berisi data detail dari KRS
7	Dosen	Berisi data dosen yang mengajar di Unikom baik dosen tetap maupun dosen luar biasa.
8	Mata Kuliah	Berisi data mata kuliah yang ada di Unikom

3. 1. 5. 3. **Analisis Basis Data Autodebet *Online***

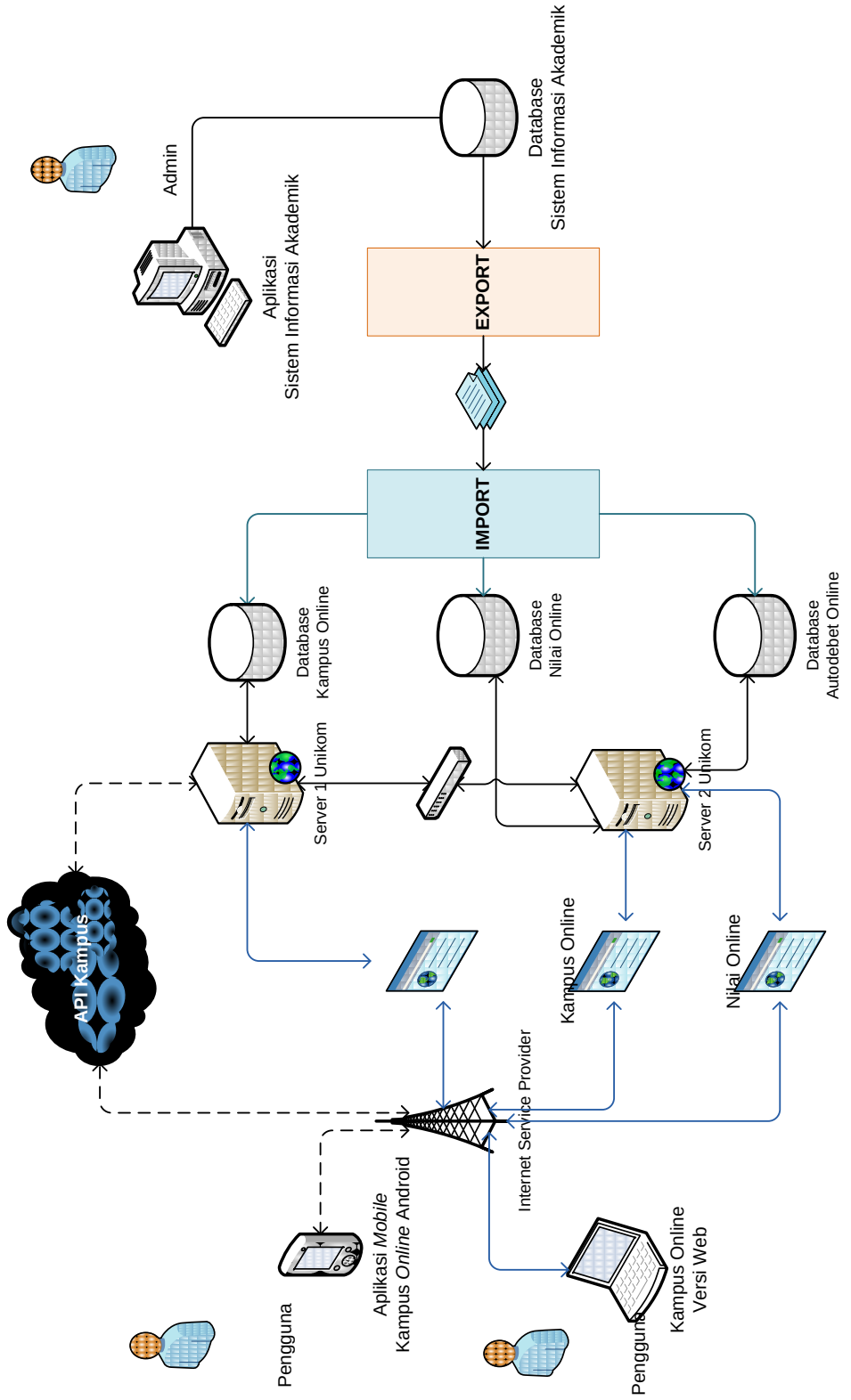
Analisis basis data Autodebet *Online* memaparkan tabel-tabel pada basis data yang saat ini digunakan dalam fasilitas Autodebet *Online*. Adapun tabel-tabel pada basis data Autodebet *Online* saat ini dapat diliahat pada Tabel III.7.

Tabel III.7 Analisis Basis Data Autodebet *Online*

No	Nama Data	Fungsi
1	Tahun Akademik	Berisi data tahun akademik yang telah ditempuh oleh mahasiswa.
2	Fakultas	Berisi data fakultas yang terdapat di Unikom
3	Jurusan	Berisi data jurusan yang terdapat di Unikom
4	Mahasiswa	Berisi data mahasiswa yang masih aktif
5	Autodebet	Berisi data autodebet yang telah dibayar

3. 1. 6. Analisis Jaringan

Analisis jaringan merupakan denah jaringan yang menggambarkan hubungan antara aplikasi Kampus *Online* berbasis Android dengan *server* Unikom melalui API Kampus sebagai *web service*. Saat ini Unikom memiliki dua server yang berbeda untuk menyimpan basis data Kampus *Online*, Nilai *Online*, dan Autodebet *Online*. Seperti yang telah dijelaskan pada analisis basis data bahwa basis data yang ada saat ini masih terpisah-pisah. Basis data Kampus *Online* terletak berada pada satu server yang terpisah dengan basis data Nilai *Online* dan Autodebet *Online*. Adapun gambaran detail arsitektur jaringan yang ada saat ini terlihat pada Gambar III.22.



Gambar III.22 Analisis Arsitektur Jaringan Unikom

3. 1. 7. Analisis API Kampus *Online*

Application Programming Interface (API) Kampus *Online* merupakan suatu sarana berbagi data antara suatu aplikasi (yang telah didaftarkan di Kampus *Online* Unikom) dengan jejaring sosial Kampus *Online* Unikom, Nilai *Online*, dan Autodebet *Online* baik untuk mengambil suatu data tertentu maupun melakukan penambahan data lainnya pada basis data Kampus *Online*, Nilai *Online*, dan Autodebet *Online* yang ada saat ini. Adapun analisis Kampus API terdiri dari data autentikasi Kampus API, kebutuhan dasar penggunaan Kampus API, HTTP *request* Kampus API, metode *request* untuk *login*, metode *request* setelah *login*, metode *post*, dan kode-kode *error* yang ada pada Kampus API.

3. 1. 7. 1. **Data Autentikasi Kampus API**

Untuk memanfaatkan fasilitas Kampus API, pengembang aplikasi harus mendaftarkan aplikasi yang akan dibangun terlebih dahulu kepada Unikom *Center*. Setelah terdaftar maka pengembang akan mendapatkan dua variabel penting yang akan selalu digunakan ketika mengambil data ataupun melakukan penambahan data pada basis data Kampus *Online*, Nilai *Online*, dan Autodebet *Online* yang ada saat ini melalui Kampus API. Adapun kedua variabel tersebut adalah API_KEY dan API_SECRET.

1. API_KEY

API_KEY merupakan kunci unik suatu aplikasi yang terdaftar sebagai pengguna layanan Kampus API. Kunci ini merupakan kunci unik seperti halnya *username* tapi digeneralisasi oleh Kampus *Online* Unikom secara otomatis. API_KEY memiliki panjang 32 karakter dan hanya bisa digunakan berpasangan

dengan API_SECRET. API_KEY akan selalu digunakan pada *request* HTTP ketika mengakses Kampus API. Hal ini dimaksudkan agar *server* Kampus *Online* Unikom dapat mengetahui *request* yang dimaksud benar-benar berasal dari aplikasi yang sudah terdaftar tersebut.

2. API_SECRET

API_SECRET merupakan kunci rahasia dari API_KEY, berfungsi seperti *salt* pada enkripsi. Semua *request* modul atau perintah API harus dienkripsi dengan *salt* dan API_KEY yang dimaksud sehingga *request* yang serupa tidak dapat dilakukan oleh pihak lainnya yang tidak memiliki API_KEY dan API_SECRET. API_SECRET juga diperlukan untuk melakukan enkripsi pada data-data riskan seperti *password*. Sehingga data yang ditransfer tidak mungkin untuk di-*decrypt* oleh pihak lain di tengah jalan dan *server* Kampus juga hanya dapat melakukan pencocokan *password* dengan *salt* yang dimaksud.

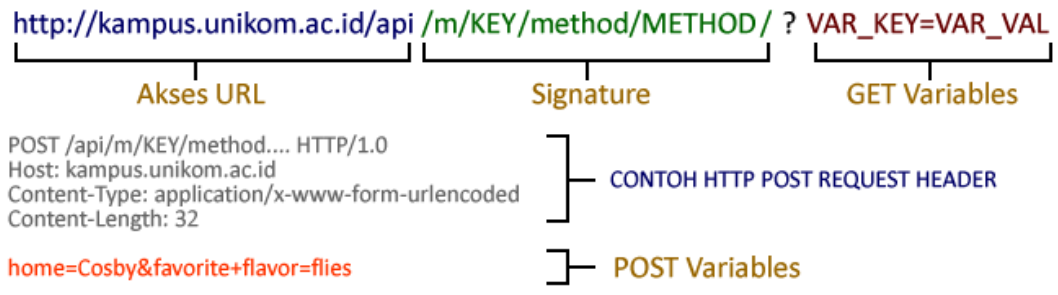
3. 1. 7. 2. **Kebutuhan Penggunaan Kampus API**

Aplikasi yang sedang dikembangkan harus memiliki fungsi enkripsi dasar dengan format MD5. Satu alamat *request* yang dikirimkan hanya dapat digunakan satu kali saja dengan tujuan agar pihak lain yang tidak memiliki API_KEY dan API_SECRET tidak dapat melakukan aktifitas serupa dengan menyalin URL yang pernah di *request* dari aplikasi yang dibangun. Setiap *request* setelah melakukan *login* akan mendapatkan variabel status yang berisi *code*, *message,* dan *session*, dimana *session* tersebut berfungsi sebagai *salt* pada atribut metode yang akan dilakukan pada *request* selanjutnya.

Selain MD5 aplikasi juga harus mendukung *urlencode* dimana *query-query* yang dikirimkan dapat berisi data-data yang valid sesuai dengan standar pengiriman URL. Bila aplikasi akan melakukan aktifitas seperti *update* status dan edit data, aplikasi harus mendukung *request* HTTP *POST* dimana ada beberapa variabel yang berisi kontent yang sangat panjang sehingga tidak akan cukup dimuat hanya dengan *request GET*.

3. 1. 7. 3. **HTTP *Request* Kampus API**

URL untuk *request* HTTP pada Kampus API terbagi dalam empat bagian, yaitu Akses URL, *Signature, GET Variables*, dan *POST Variables*. Setiap bagian tersebut memiliki fungsi masing-masing dan beberapa bagian harus dihitung terlebih dahulu dan tergantung dari *request* sebelumnya atau API_KEY dan API_SECRET dari aplikasi. Berikut gambaran HTTP request Kampus API yang terlihat pada Gambar III.23.



Gambar III.23 *Request* HTTP Kampus API

3. 1. 7. 3. 1. Akses URL

Akses URL merupakan bagian statis dari *request* HTTP pada Kampus API. Bagian ini harus berisi http://kampus.unikom.ac.id/api

3. 1. 7. 3. 2. Signature

Signature adalah bagian *request* yang paling utama dan sedikit sukar untuk diimplementasikan. Pada bagian *signature* ini dibagi ke dalam dua bagian, yaitu bagian KEY dan bagian METHOD.

1. KEY

Bagian ini memiliki format dasar /m/API_KEY, dimana API_KEY harus diisi oleh API_KEY aplikasi Anda yang telah didaftarkan sebelumnya. Contoh: /m/d5ec955a07c71f46c55cf6adaedc9213.

2. METHOD

Bagian ini merupakan token untuk *request* yang diminta dengan format dasar /metode/METHOD_TOKEN/, dimana METHOD_TOKEN adalah hasil perhitungan *hash* dengan MD5 dari kombinasi nama metode yang akan diakses dengan API_SECRET.

3. 1. 7. 3. 3. GET Variables

GET Variables merupakan *query* string pada URL yang mengirimkan data-data berupa nama variabel dan isi variabel. *GET Variables* ini harus dimasukkan setelah akses URL, *signature*, dan tanda tanya "?" dengan format dasar namavariabel1=isivariabel1&namavariabel2=isivariabel2. Satu variabel dipisahkan dengan tanda &, dan antara nama variabel dan isi variabel dipisahkan dengan tanda =, dimana isi variabel harus dalam keadaan telah di *urlencode*.

3. 1. 7. 3. 4. POST Variables

Seperti halnya *GET Variables, POST Variables* memiliki format dasar yang sama persis hanya saja peletakan variabel ini tidak berada pada URL tapi berada pada *request body*.

3. 1. 7. 4. **Penulisan Respon Data Kampus API**

Penulisan respon data dari Kampus API merupakan penjelasan bagaimana respon data dari metode-metode pada Kampus API ditulis dalam laporan tugas akhir ini. Respon tersebut merupakan data yang didapat dari basis data Kampus *Online*, Nilai *Online*, dan Autodebet *Online* yang ada saat ini dalam bentuk JSON. Adapun format penulisan dari respon data tersebut adalah sebagai berikut:

1	**nama_variabel_1**	Penjelasan **nama_variabel_1**
a	**nama _variabel_anak_1**	Penjelasan **nama _variabel_anak_1**
(a)	**nama_variabel_anak_a**	Penjelasan **nama_variabel_anak_a**

Keterangan : 1, a, (a) adalah urutan variabel yang diperoleh

3. 1. 7. 5. **Metode *Request Login***

Hal pertama yang harus dilakukan sebelum melakukan aktifitas dengan Kampus API adalah dengan melakukan *login*. Dengan melakukan *login* maka aplikasi *mobile* Kampus *Online* Android akan mendapatkan variabel *session* yang dapat digunakan untuk *signature* pada *request-request* selanjutnya.

Metode *request login* merupakan metode yang digunakan oleh aplikasi Kampus *Online* Android untuk mengambil data *login* pada basis data Kampus *Online* yang ada saat ini. Metode *request login* memiliki beberapa variabel yang

dikirimkan sebagai parameter pada saat melakukan *login* ke dalam Kampus *Online*. Berikut ini variable-variabel yang dibutuhkan oleh metode *login* pada saat melakukan *login* ke dalam Kampus *Online* seperti terlihat pada Tabel III.8.

Tabel III.8 Variabel Metode *Request Login*

Nama Variabel	Isi Variabel	Tipe Variabel
username	*Username* dari pengguna yang akan melakukan *login*.	GET
token	Kombinasi Username, Password dan API_KEY	GET
keeplogin	1 = Ingat *Login*	GET

Hasil dari metode *login* yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun struktur respon data hasil metode *login* adalah sebagai berikut:

1. status – Berisi data status *request* yang diminta
    a. code – Kode status dari *request*. Bila berisi 0, status *request* tersebut adalah OK atau Berhasil, selain 0 berarti error (Kode status akan dibahas pada bagian berikutnya)
    b. message – Merupakan pesan status dari *request* yang dimaksud. Biasanya berisi OK bila berhasil, atau berisi pesan error bila status *request* tersebut tidak berhasil.
    c. servertime – UNIX Timestamp server ketika melakukan kalkulasi *request* ini.

2. data – Berisi data hasil *login* bila *login* yang dilakukan berhasil.
    a. session – Merupakan variabel yang harus disimpan oleh aplikasi untuk melakukan kalkulasi metode dan tambahan variabel pada *request* selanjutnya.
    b. *username* – Username dari *user* yang telah melakukan *login* tersebut.
    c. displayname – Nama tampilan dari *user* yang telah melakukan *login* tersebut.
    d. tipe – Tipe *user* dari *user* yang telah melakukan *login* tersebut, misalnya mahasiswa, dosen, karyawan, alumni dan pimpinan.
    e. support_facebook – Akan berisi 1 bila *user* ini dapat melakukan update status di facebook dengan menggunakan variabel update_facebook=1 pada metode postStatus.
    f. support_twitter – Akan berisi 1 bila *user* ini dapat melakukan update status di twitter juga dengan menggunakan variabel update_twitter=1 pada metode postStatus.
    g. profile_picture – Akan berisi URL image foto profile dari *user* yang melakukan *login*
    h. keterangan – Keterangan *user* yang sedang *login*, misalnya: Dosen Jurusan Teknik Informatika.

3. 1. 7. 6. **Metode *Request* Setelah *Login***

Metode *request* setelah *login* berisi metode-metode yang memungkinkan aplikasi *mobile* Kampus *Online* Android untuk mengambil data pada basis data Kampus *Online*, Nilai *Online*, dan Autodebet *Online* yang ada saat ini. Metode *request* Kampus API setelah *login* sedikit berbeda dengan operasi *request* pada saat melakukan *login*. Perbedaanya yaitu untuk melakukan *request* setelah *login* aplikasi *mobile* Kampus *Online* Android membutuhkan data *session* yang didapatkan dari respon ketika melakukan *login*. Adapun metode-metode yang terdapat pada Kampus API setelah melakukan *login* adalah checkSession, getUserData, getUsersData, getStatus, getNotifications, getComments, getLikes, getFriends, getNilai, dan getAutodebet.

1. checkSession

Metode checkSession merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mendapatkan data *session* yang sedang berjalan pada basis data Kampus *Online* yang ada saat ini. Metode checkSession memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data *session*. Berikut ini variabel yang dibutuhkan oleh metode checkSession dalam mengambil data *session* pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.9.

Tabel III.9 Variabel Metode getUserData

Nama Variabel	Isi Variabel	Tipe Variabel
username	*Username* dari pengguna yang telah melakukan proses *login*.	GET
session	Session yang didapat pada metode *login*	GET

Hasil dari metode checkSession yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan dipetakan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun struktur respon data hasil metode checkSession tersebut adalah sebagai berikut:

1. session - ID dari session yang sedang berjalan
2. session_expire – Waktu kadaluarsa session dalam format UNIX
3. username – Username yang sedang *login*
4. displayname – Nama Tampilan *user* yang sedang *login*
5. tipe – Tipe User
6. support_facebook – Mendukung update status via facebook atau tidak (0/1)
7. support_twitter – Mendukung update status via twitter atau tidak (0/1)
8. profile_picture – Alamat file image profile picture
9. keterangan - Keterangan dari *user* yang sedang *login*

2. getUserData

Metode getUserData merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mendapatkan data pengguna yang diinginkan pada basis data Kampus *Online* yang ada saat ini. Metode ini digunakan hanya untuk meminta data pengguna sebanyak satu pengguna. Apabila aplikasi yang dibangun membutuhkan data untuk beberapa pengguna maka gunakan metode **getUsersData**. Metode getUserData memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data pengguna. Berikut ini variabel yang dibutuhkan oleh metode getUserData dalam

mengambil data pengguna pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.10.

Tabel III.10 Variabel Metode getUserData

Nama Variabel	Isi Variabel	Tipe Variabel
username	*Username* dari pengguna yang telah melakukan proses *login*.	GET
session	Session yang didapat pada metode *login*	GET

Hasil dari metode getUserData yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun struktur respon data hasil metode checkSession adalah sebagai berikut:

1. username – Username dari *user* yang diminta
2. tipe – Tipe *user* (mahasiswa, alumni, dosen, karyawan,pimpinan)
3. displayname – Nama Tampilan
4. noid – Nomor Identitas *user* (misalnya NIM)
5. profile_picture – Alamat image profile picture dari *user* yang diminta (Format JPEG 60x60px)
6. keterangan – Keterangan singkat *user* (misal: Dosen Teknik Informatika)

7. *friend*_status – Status Pertemanan dengan *user* yang melakukan *login*
    a. code – Kode pertemanan (0=bukan teman, 1=User yang *login* mengikuti statusnya, 2=User ini mengikuti status *user* yang sedang *login*, 3=User tersebut adalah *user* yang sedang *login*)
    b. description – Penjelasan tentang kode pertemanan
8. status – Status getUserData
    a. code – Kode status (0 = OK, 5=variabel *username* harus diisi, 6=*username* tidak ditemukan)
    b. message – Pesan status

3. getUsersData

Metode getUsersData merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mendapatkan data dari beberapa pengguna yang diinginkan pada basis data Kampus *Online* yang ada saat ini. Metode ini digunakan untuk mengambil data dari beberapa pengguna sekaligus. Apabila aplikasi yang dibangun membutuhkan data untuk satu pengguna saja maka gunakan metode **getUserData**. Metode getUsersData memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data pengguna. Berikut ini variabel yang dibutuhkan oleh metode getUsersData dalam mengambil data pengguna pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.11.

Tabel III.11 Variabel Metode getUsersData

Nama Variabel	Isi Variabel	Tipe Variabel
usernames	Beberapa Username dari *user* yang akan di ambil datanya dengan koma sebagai pemisah. Contoh: **usernames=10502131,amarullz,nursidik**	GET
session	Session yang didapat pada metode *login*	GET

Hasil dari metode getUsersData yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun struktur respon data hasil metode getUsersData adalah sebagai berikut:

1. item[array] – Berisi *item array* dengan *index* berurutan, dan berisi informasi *user-user* yang diminta.
    a. *username* – Username dari *user* yang diminta
    b. tipe – Tipe *user* (mahasiswa, alumni, dosen, karyawan,pimpinan)
    c. displayname – Nama Tampilan
    d. noid – Nomor Identitas *user* (misalnya NIM)
    e. profile_picture – Alamat image profile picture dari *user* yang diminta (Format JPEG 60x60px)
    f. keterangan – Keterangan singkan *user* (misal: Dosen Teknik Informatika)

g. *friend*_status – Status Pertemanan dengan *user* yang melakukan *login*

   1. code – Kode pertemanan (0=bukan teman, 1=User yang *login* mengikuti statusnya, 2=User ini mengikuti status *user* yang sedang *login*, 3=User tersebut adalah *user* yang sedang *login*)
   2. description – Penjelasan tentang kode pertemanan

h. status – Status getUserData

   1. code – Kode status (0 = OK, 5=variabel *username* harus diisi, 6=*username* tidak ditemukan)
   2. message – Pesan status

4. getStatus

Metode getStatus merupakan metode yang digunakan untuk mendapatkan data status dari teman, yaitu *following* dan *friends* yang diambil dari basis data Kampus *Online* yang ada saat ini. Dengan metode ini aplikasi dapat meminta status-status terbaru, status-status pada *profile* atau *wall user* lain, dan status dengan id tertentu.

Metode getStatus memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data status. Berikut ini variabel yang dibutuhkan oleh metode getStatus dalam mengambil data status pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.12.

Tabel III.12 Variabel Metode getStatus

Nama Variabel	Isi Variabel	Default	Tipe Variabel
usernames	Beberapa Username dari *user* yang akan di ambil datanya dengan koma sebagai pemisah. Contoh: **usernames=10502131,amarullz,nursidik**	-	GET
session	Session yang didapat pada metode *login*	-	GET
limit	Jumlah data maksimal yang diminta. Bila tidak di set, maka isi variabel ini akan bernilai 10, bila variabel **id** di set, maka limit akan diacuhkan dan hanya akan menampilkan satu data status saja	10	GET
start	Posisi awal data status yang diminta dengan indeks awal 0. Bila tidak di set, variabel ini akan bernilai 0, bila variabel **id** di set maka *start* akan diacuhkan. Variabel ini berguna untuk menampilkan data selanjutnya dengan memasukan isi respon **next_***start* pada getStatus sebelumnya.	0	GET
id	ID status yang ingin ditampilkan. Bila di set, jumlah data yang akan didapatkan hanya sebanyak 1 status saja yang memiliki ID sesuai dengan isi variabel ini, bila tidak di set, maka akan menampilkan listing status. ID Status dapat diperoleh dari beberapa metode lain, diantaranya dengan menggunakan metode **getNotifications**.	0	GET
userwall	Username dari status yang ingin ditampilkan. Dengan melakukan set variabel ini, status yang tampil hanya yang berhubungan dengan *user* yang dimaksud.	Kosong	GET

Hasil dari metode getStatus yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun struktur respon data hasil metode getStatus adalah sebagai berikut:

1. next_start – Berisi posisi untuk status-status pada halaman selanjutnya, isi respon ini dapat digunakan pada getStatus selanjutnya dengan menambahkan variabel start. Bila tidak terdapat status selanjutnya, respon data ini akan berisi text EOF (End Of File).
2. statuses_count – Berisi informasi jumlah status yang didapatkan.
3. statuses – Berisi data-data status yang diminta berupa item array sebagai berikut:
4. item[array] – Berisi item array dengan index berurutan, dan berisi data-data status yang diminta.
    a. id – ID dari status
    b. poster – Data *user* yang mengirim status dengan format seperti yang didapat dengan menggunakan getUserData.
    c. target – Data *user* tujuan dengan format seperti yang didapat dengan menggunakan getUserData. Data ini hanya akan tampil bila poster mengirim status pada profil orang lain (atau memberikan "sapaan").
    d. unixtime – Timestamp di server dengan format UNIX – Waktu pengiriman status.
    e. posted – Waktu pengiriman status dengan format readable (misal 1 Jam yang lalu).
    f. text – Isi dari status

g. stat – Statistik dari status.
    1. comments – Jumlah komentar pada status ini
    2. bagus – Jumlah *rating* bagus pada status ini
    3. jelek – Jumlah *rating* jelek pada status ini
    4. rated – Status *rating user* yang sedang *login*. Bila *user* yang sedang *login* telah melakukan *rating*, data ini akan berisi 1, bila belum akan berisi 0.

h. mentions – Data *user-user* yang di mention/singgung pada status ini dengan format yang sama seperti pada getUsersData. Data ini hanya akan tampil bila text status berisi mention/singgungan pada *user* lainnya.

i. can_delete – Bernilai 1 bila status yang dimaksud dapat dihapus dengan metode deleteStatus, bila bernilai 0, maka status yang dimaksud tidak dapat dihapus.

5. getNotifications

Metode getNotifications merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mendapatkan data notifikasi pengguna yang sedang melakukan *login* pada basis data Kampus *Online* yang ada saat ini. Dengan metode ini aplikasi dapat meminta data-data notifikasi pengguna baik itu semua notifikasi, notifikasi yang belum dibaca saja, jumlah notifikasi yang belum dibaca saja, ataupun mencari *update* notifikasi terbaru. Selain mengambil data notifikasi, metode ini juga dapat melakukan perubahan status notifikasi dari *unreaded* menjadi *readed* secara otomatis setelah data tersebut diminta.

Metode getNotifications memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data notifikasi. Berikut ini variabel yang dibutuhkan oleh metode getNotifications dalam mengambil data notifikasi pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.13.

Tabel III.13 Variabel Metode getNotifications

Nama Variabel	Isi Variabel	Default	Tipe Variabel
usernames	Beberapa Username dari *user* yang akan di ambil datanya dengan koma sebagai pemisah. Contoh: **usernames=10502131,amarullz,nursidik**	-	GET
session	Session yang didapat pada metode *login*	-	GET
limit	Jumlah data maksimal yang diminta. Bila tidak di set, maka isi variabel ini akan bernilai 50.	50	GET
unreadonly	Set 1 bila hanya ingin membaca notifikasi yang belum dibaca saja, set 0 bila ingin membaca semua notifikasi.	0	GET
countonly	Set 1 bila hanya ingin mengambil jumlah notifikasi yang belum dibaca saja. Set 0 bila ingin mengambil juga data notifikasinya. Bila di set 1 maka respon **notifications** tidak akan didapatkan.	0	GET
updateid	Set dengan **next_updateid** yang didapatkan pada **getNotifications** sebelumnya bila hanya akan mengambil data terbaru tanpa memasukan data yang sebelumnya telah didapatkan.	0	
autoreaded	Set 1 bila akan mengeset semua notifikasi yang belum di baca menjadi sudah dibaca secara otomatis. Set 0 bila tidak ingin melakukannya.	0	GET

Hasil dari metode getNotifications yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun struktur respon data hasil metode getNotifications adalah sebagai berikut:

1. unread_count – Berisi jumlah notifikasi yang belum dibaca
2. next_updateid – Berisi id untuk dimasukan pada variabel updateid ketika memanggil getNotifications selanjutnya untuk mendapatkan notifikasi terbaru saja (update).
3. Notifications – Berisi data-data notifikasi yang diminta berupa item array berikut :
4. item[array] – Berisi item array dengan index berurutan, dan berisi data-data notifikasi yang diminta.
    a. id – ID dari notifikasi, dapat digunakan pada metode setReadNotifications.
    b. unixtime – Timestamp di server dengan format UNIX – Waktu notifikasi.
    c. time – Waktu notifikasi dengan format readable (misal 1 Jam yang lalu).
    d. message – Pesan notifikasi
    e. url – URL item notifikasi di Kampus *Online* Unikom.
    f. type – Tipe notifikasi (comment, ratebagus, ratejelek, sapaan, mention, *friend*).

g. statusid – Bila tipe notifikasi bukan "*friend*", maka respon ini akan didapatkan. Akan berisi ID dari status yang dimaksud. Dapat digunakan pada variabel id ketika memanggil metode getStatus.
h. *friend* – Bila tipe notifikasi "*friend*", maka respon ini akan didapatkan. Respond ini akan berisi data *user* yang bersangkutan dengan format seperti yang didapat dengan menggunakan getUserData.

6. getComments

Metode getComments merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mendapatkan data komentar dari status pengguna pada basis data Kampus *Online* yang ada saat ini. Dengan metode ini aplikasi dapat meminta komentar-komentar terbaru pada status. Pengambilan komentar harus disertai variabel **statusid** yang harus diisi dengan id dari status yang ingin diambil komentarnya.

Metode getComments memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data komentar. Berikut ini variabel yang dibutuhkan oleh metode getComments dalam mengambil data komentar pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.14.

Tabel III.14 Variabel Metode getComments

Nama Variabel	Isi Variabel	Default	Tipe Variabel
usernames	Beberapa Username dari *user* yang akan di ambil datanya dengan koma sebagai pemisah. Contoh: **usernames=10502131,amarullz,nursidik**	-	GET
session	Session yang didapat pada metode *login*	-	GET
statusid	ID status yang komentarnya ingin ditampilkan. ID Status bisa didapatkan dari hasil metode **getStatus**.	Harus diisi	GET
limit	Jumlah data maksimal yang diminta. Bila tidak di set, maka isi variabel ini akan bernilai 10.	10	GET
start	Posisi awal data komentar yang diminta. Dengan indeks awal 0, bila tidak di set, variabel ini akan bernilai 0. Variabel ini berguna untuk menampilkan data selanjutnya, dengan memasukan isi respon **next_***start* pada **getComments** sebelumnya.	0	GET

Hasil dari metode getComments yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun struktur respon data hasil metode getComments adalah sebagai berikut:

1. next_*start* – Berisi posisi untuk komentar-komentar pada halaman selanjutnya, isi respon ini dapat digunakan pada getComments selanjutnya dengan menambahkan variabel *start*. Bila tidak terdapat komentar selanjutnya, respon data ini akan berisi text EOF (End Of File).
2. show_count – Berisi jumlah komentar yang ditampilkan.
3. count – Berisi jumlah seluruh komentar pada status yang dimaksud.
4. comments – Berisi data-data komentar yang diminta berisi item array dengan index berurutan, dan berisi data-data komentar yang diminta sebagai berikut:
    a. id – ID dari komentar
    b. poster – Data *user* yang mengirim komentar dengan format seperti yang didapat dengan menggunakan getUserData.
    c. unixtime – Timestamp di server dengan format UNIX – Waktu pengiriman komentar.
    d. posted – Waktu pengiriman komentar dengan format readable (misal 1 Jam yang lalu).
    e. text – Isi dari komentar
    f. mentions – Data *user-user* yang di mention/singgung pada komentar ini dengan format yang sama seperti pada getUsersData. Data ini hanya akan tampil bila text komentar berisi mention/singgungan pada *user* lainnya.
    g. can_delete – Bernilai 1 bila komentar yang dimaksud dapat dihapus dengan metode deleteComment, bila bernilai 0, maka komentar yang dimaksud tidak dapat dihapus.

7. getLikes

Metode getLikes merupakan metode yang digunakan digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mendapatkan data *rating* status (Bagus atau Jelek) pada basis data Kampus *Online* yang ada saat ini. Dengan metode ini aplikasi dapat meminta data-data pengguna yang telah melakukan *rating* pada status. Pengambilan *rating* harus disertai variabel statusid yang harus diisi dengan id dari status yang ingin diambil data *rating*-nya.

Metode getLikes memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data *rating*. Berikut ini variabel yang dibutuhkan oleh metode getLikes dalam mengambil data *rating* pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.15.

Tabel III.15 Variabel Metode getLikes

Nama Variabel	Isi Variabel	Default	Tipe Variabel
usernames	Beberapa Username dari *user* yang akan di ambil datanya dengan koma sebagai pemisah. Contoh: **usernames=10502131,amarullz,nursidik**	-	GET
session	Session yang didapat pada metode *login*	-	GET
statusid	ID status yang data *rating*-nya ingin ditampilkan. ID Status bisa didapatkan dari hasil metode **getStatus**.	Harus diisi	GET
limit	Jumlah data maksimal yang diminta. Bila tidak di set, maka isi variabel ini akan bernilai 10.	10	GET
start	Posisi awal data *rating* yang diminta. Dengan indeks awal 0, bila tidak di set, variabel ini akan bernilai 0. Variabel ini berguna untuk menampilkan data selanjutnya, dengan memasukan isi respon **next_***start* pada **getLikes** sebelumnya.	0	GET

Hasil dari metode getLikes yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun struktur respon data hasil metode getLikes adalah sebagai berikut:

1. next_*start* – Berisi posisi untuk data-data *user* yang melakukan *rating* pada halaman selanjutnya, isi respon ini dapat digunakan pada getLikes selanjutnya dengan menambahkan variabel *start*. Bila tidak terdapat data *rating* selanjutnya, respon data ini akan berisi text EOF (*End Of File*).
2. show_count – Berisi jumlah data *rating* yang ditampilkan.
3. count – Berisi jumlah seluruh data *rating* pada status yang dimaksud.
4. count_bagus – Berisi semua jumlah *rating* bagus pada status yang dimaksud.
5. count_jelek – Berisi semua jumlah *rating* jelek pada status yang dimaksud.
6. likes – Berisi data-data *rating* yang diminta berisi item array dengan index berurutan, dan berisi data-data *rating* yang diminta sebagai berikut:
    a. poster – Data *user* yang melakukan *rating* dengan format seperti yang didapat dengan menggunakan getUserData.
    b. type – Tipe *rating* yang dilakukan (bagus, jelek).

8. getFriends

Metode getFriends merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mendapatkan data daftar pertemanan dari pengguna yang diinginkan pada basis data Kampus *Online* yang ada saat ini.

Dalam melakukan pencarian data teman, metode getFriends juga dapat melakukan filterisasi atau pencarian teman dan mengetahui tipe pertemanan. Tipe pertemanan pada Kampus *Online* Unikom dibagi menjadi tiga tipe, yaitu *following*, *followers*, dan *friends*. Berikut penjelasan dari masing-masing tipe pertemanan yang dapat dilihat pada Tabel III.16.

Tabel III.16 Tipe Pertemanan

Kode	Nama	Keterangan
1	*following*	User yang dimaksud telah mengikuti status temannya.
2	*follower*	Temannya telah mengikuti status *user* yang dimaksud.
3	*friend*	Telah menjadi teman (*user* yang dimaksud dan temannya telah saling mengikuti status mereka masing-masing).

Metode getFriends memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data daftar pertemanan. Berikut ini variabel yang dibutuhkan oleh metode getFriends dalam mengambil data daftar pertemanan pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.17.

Tabel III.17 Variabel Metode getFriends

Nama Variabel	Isi Variabel	Default	Tipe Variabel
usernames	Beberapa Username dari *user* yang akan di ambil datanya dengan koma sebagai pemisah. Contoh: **usernames=10502131,amarullz,nursidik**	-	GET
session	Session yang didapat pada metode *login*	-	GET
friendtype	*all, friend, following, follower* Hanya tampilkan *user* yang memiliki tipe pertemanan sesuai dengan yang dimasukkan. Bila diisi *all*, maka semua temannya baik itu *follower*, *following*, atau *friend* akan ditampilkan.	all	GET
username	Username dari *user* yang akan ditampilkan data temannya. Bila tidak diisi, maka data yang ditampilkan akan berisi data teman-teman *user* yang sedang melakukan *login*.	Username yang sedang *login*	GET
search	Kata pencarian teman Bila diisi, maka data yang ditampilkan hanya data dari teman *user* yang dimaksud yang memiliki kriteria pencarian. Pencarian ini akan mencocokan variabel search ini dengan *username*, nim dan *displayname*.	Kosong	GET
limit	Jumlah data maksimal yang diminta. Bila tidak di set, maka isi variabel ini akan bernilai 10.	10	GET
start	Posisi awal data teman yang diminta. Dengan indeks awal 0, bila tidak di set, variabel ini akan bernilai 0. Variabel ini berguna untuk menampilkan data selanjutnya, dengan memasukan isi respon **next_***start* pada **getFriends** sebelumnya.	0	GET

Hasil dari metode getFriends yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun struktur respon data hasil metode getFriends adalah sebagai berikut:

1. next_*start* – Berisi posisi untuk data-data teman *user* yang dimaksud pada halaman selanjutnya, isi respon ini dapat digunakan pada getFriends selanjutnya dengan menambahkan variabel *start*. Bila tidak terdapat data *rating* selanjutnya, respon data ini akan berisi text EOF (End Of File).
2. show_count – Berisi jumlah data yang ditampilkan.
3. count – Berisi jumlah seluruh data yang dimaksud.
4. get_type – Berisi variabel friendtype yang dikirimkan dalam *request*.
5. friendof – Berisi variabel *username* yang dikirimkan dalam *request*, atau berisi *username user* yang sedang *login* bila variabel *username* tidak di set ketika melakukan *request*.
6. get_search – Berisi variabel search yang dikirimkan dalam *request*, respon ini tidak akan tampil bila variabel search tidak di set ketika melakukan *request*.
7. *friends* – Berisi data-data teman yang diminta yang berisi item array dengan index berurutan, dan berisi data-data teman yang diminta sebagai berikut:

a. *username* – Username teman.
b. *friend*_code – Kode pertemanan sesuai dengan tabel tipe pertemanan di atas.
c. *friend*_type – Keterangan dari *friend*_code.
d. *friend*_data – Data teman dengan format seperti yang didapat dengan menggunakan getUserData.

9. getNilai

Metode getNilai merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mendapatkan data nilai akhir dari semester yang telah ditempuh pada basis data Nilai *Online* yang ada saat ini. Dengan metode ini aplikasi dapat meminta data semester yang telah ditempuh, data nilai, dan indeks prestasi dari semester yang dipilih.

Metode getNilai memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data semester yang ditempuh dan data nilai dari setiap semester. Berikut ini variabel yang dibutuhkan oleh metode getNilai dalam mengambil data nilai pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.18.

Tabel III.18 Variabel Metode getNilai

Nama Variabel	Isi Variabel	Default	Tipe Variabel
usernames	Beberapa Username dari *user* yang akan di ambil datanya dengan koma sebagai pemisah. Contoh: **usernames=10502131,amarullz**	-	GET
session	Session yang didapat pada metode *login*	-	GET
listtahunakad	0 = Ambil data Nilai pada tahun akademik tertentu 1 = Ambil Listing Tahun Akademik	0	GET
tahunakad	Ambil data nilai pada tahun akademik yang dimaksud. Listing tahun akademik dapat di lihat dengan meminta metode ditambah variabel **listtahunakad=1**. Contoh: **tahunakad=20101** untuk tahun 2010 semester ganjil.	Semester yang berjalan	GET

Hasil dari metode getNilai yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Respon data dari metode getNilai dibagi menjadi dua tipe, yaitu untuk listtahunakad = 1 dan listtahunakad = 0. Adapun struktur respon data hasil metode getNilai untuk listtahunakad = 1 adalah sebagai berikut:

1. current – Tahun Akademik yang sedang berjalan
2. count – Jumlah tahun akademik yang ada di database nilai

3. tahunakad[array] – Berisi item array dengan index berurutan, yang berisi data tahun akademik yang ada di database nilai.
    a. tahunakad – Kunci tahun akademik untuk digunakan pada parameter tahunakad pada metode getNilai tanpa listtahunakad=1.
    b. tahun – Tahun Ajaran
    c. semester – Ganjil atau Genap
    d. iscurrent – Bernilai 1 bila merupakan tahun ajaran yang sedang berjalan, 0 bila bukan.

Sedangkan untuk listtahunakad = 0 memiliki struktur respon data JSON sebagai berikut:

1. nim – Nim mahasiswa / *user* yang sedang *login*
2. nama – Nama lengkap *user* yang terdaftar pada database akademik unikom.
3. kodejur – Kode Jurusan
4. kodefak – Kode Fakultas
5. prodi – Program Studi (S1, D3, dsb)
6. kelas – Kelas Mahasiswa
7. thnmasuk – Tahun Masuk mahasiswa
8. pmdk – Apakah mahasiswa tersebut PMDK atau bukan
9. dosenwali – Bila ada akan berisi nama dari dosen wali mahasiswa
10. fakultas – Nama Fakultas
11. jurusan – Nama Jurusan

12. tahunakad – Berisi data-data tahun akademik yang dipilih pada nilai yang ditampilkan
    a. tahunakad – Tahun akademik dengan format "20101"
    b. tahun – Tahun
    c. semester – Nama semester (Ganjil, Genap)
13. nilai[array] - Berisi item array dengan index berurutan, yang berisi data nilai pada tahun akademik yang diminta.
    a. uploaded – 1 : Sudah di upload, 0 : Belum di upload.
    b. kodemk – Kode Matakuliah
    c. namamk – Nama Matakuliah
    d. namadosen – Nama Dosen Pengajar pada matakuliah ini
    e. sks – SKS dari matakuliah
    f. nilai – Nilai yang didapat (A=4, B=3, C=2, D=1, LAIN=0)
    g. bobot – Bobot yang didapat (sks * nilai)
    h. index – Nilai yang didapat dalam index (A,B,C,D,E,T).
14. jumlah – Berisi data-data jumlah dari nilai-nilai yang didapatkan
    a. sks – Jumlah SKS yang ditempuh
    b. nilai – Jumlah nilai yang didapatkan
    c. bobot – Jumlah bobot yang didapatkan
    d. ip – IP yang didapatkan pada tahun akademik yang dimaksud.

10. getAutodebet

Metode getAutodebet merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mendapatkan data tahun akademik yang

telah ditempuh dan data autodebet mahasiswa yang telah dibayarkan setiap semesternya pada basis data Autodebet *Online* yang ada saat ini. Dengan metode ini aplikasi dapat meminta data-data autodebet berdasarkan semester dan angsuran yang dipilih.

Metode getAutodebet memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan pengambilan data tahun akademik yang telah ditempuh dan data autodebet mahasiswa yang telah dibayarkan setiap semesternya. Berikut ini variabel yang dibutuhkan oleh metode getAutodebet dalam mengambil data autodebet pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.19.

Tabel III.19 Variabel Metode getAutodebet

Nama Variabel	Isi Variabel	Default	Tipe Variabel
usernames	Beberapa Username dari *user* yang akan di ambil datanya dengan koma sebagai pemisah. Contoh: **usernames=10502131,amarullz,nursidik**	-	GET
session	Session yang didapat pada metode *login*	-	GET
listtahunakad	0 = Ambil data Nilai pada tahun akademik tertentu 1 = Ambil Listing Tahun Akademik	0	GET
tahunakad	Ambil data nilai pada tahun akademik yang dimaksud. Listing tahun akademik dapat di lihat dengan meminta metode ditambah variabel **listtahunakad=1**. Contoh: **tahunakad=20101** untuk tahun 2010 semester ganjil.	Semester yang berjalan	GET

Respon data dari metode getAutodebet berupa *array* dengan *header* data mahasiswa, yaitu nim dan nam serta data detail angsuran dan semester pembayaran yang telah dipilih. Adapun hasil respon data dari metode getAutodebet adalah sebagai berikut:

Hasil dari metode getAutodebet yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Respon data dari metode getAutodebet dibagi menjadi dua tipe, yaitu untuk listtahunakad = 1 dan listtahunakad = 0. Adapun struktur respon data hasil metode getAutodebet untuk listtahunakad = 1 adalah sebagai berikut:

1. current – Tahun Akademik yang sedang berjalan dengan format (YYYYYYYY) contoh (20102011 yang berarti 2011/2011)
2. count – Jumlah tahun akademik yang ada di database nilai
3. tahunakad[array] – Berisi item array dengan index berurutan, yang berisi data tahun akademik yang ada di database nilai.
    a. tahunakad – Kunci tahun akademik untuk digunakan pada parameter tahunakad pada method getAutodebet tanpa listtahunakad=1.
    b. tahun – Tahun Ajaran
    c. iscurrent – Bernilai 1 bila merupakan tahun ajaran yang sedang berjalan, 0 bila bukan.

Sedangkan untuk listtahunakad = 0 memiliki struktur respon data JSON sebagai berikut:

1. Bila Error akan terdapat data dengan *key* "error" dan pesan lengkapnya.
2. nim – Nim mahasiswa / *user* yang sedang login
3. nama – Nama lengkap *user* yang terdaftar pada *database* akademik Unikom.
4. tahunakad – Berisi data-data tahun akademik yang dipilih pada nilai yang ditampilkan
5. autodebet[array] - Berisi *item array* dengan index berurutan, yang berisi data autodebet pada tahun akademik yang diminta. Bila *error* akan terdapat data dengan key "error" dan pesan lengkapnya.
    a. angsuran – Angsuran ke berdasarkan nomor urut dari 1 sampai 6.
    b. nominal – Jumlah yang dibayarkan
    c. tgl – Tanggal Autodebet

3. 1. 7. 7. **Metode Post**

Metode *post* berisi metode-metode yang memungkinkan aplikasi *mobile* Kampus *Online* Android untuk melakukan penambahan atau perubahan data pada basis data Kampus *Online* yang ada saat ini. Sama seperti metode *request* setelah *login*, metode ini pun membutuhkan variabel *session* untuk mengirimkan data. Berikut ini metode-metode *post* yang terdapat dalam Kampus API adalah postStatus, postComments, dan postLike.

1. postStatus

Metode postStatus merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mengirimkan menambahkan status atau sapaan terhadap pengguna lain pada basis data Kampus *Online* yang ada saat ini.

Metode postStatus memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan penambahan data status atau sapaan kepada pengguna lain. Berikut ini variabel yang dibutuhkan oleh metode postStatus dalam melakukan penambahan data (status atau sapaan) pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.20.

Tabel III.20 Variabel Metode postStatus

Nama Variabel	Isi Variabel	Default	Tipe Variabel
update_facebook	1 atau 0 Set 1 bila akan mengupdate juga status di facebook, dengan syarat respon support_facebook pada saat *login* bernilai 1.	0	GET
update_twitter	1 atau 0 Set 1 bila akan mengupdate juga status di twitter, dengan syarat respon support_twitter pada saat *login* bernilai 1.	0	GET
target	Nama *user* yang akan disapa. Bila update status adalah sapaan (menulis di wall teman) maka target ini diisi dengan *username* tujuan. Target ini tidak dapat disatukan dengan update_facebook dan update_twitter.	Kosong	GET
text	Diisi dengan text update status Untuk emoticon tulis dengan <emo:nomor_emoticon>, untuk mention gunakan *@username*. Text ini wajib diisi, dan maksimal sepanjang 400 karakter	Kosong	GET

Hasil dari metode postStatus yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun respon data dari metode postStatus adalah berisi id status yang baru saja dikirimkan dengan format yang sama dengan respon item statuses pada metode getStatus.

2. postComment

Metode postComments merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mengirimkan komentar terhadap suatu status pada basis data Kampus *Online* yang ada saat ini.

Metode postComments memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan penambahan data komentar. Berikut ini variabel yang dibutuhkan oleh metode postComments dalam melakukan penambahan data komentar pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.21.

Tabel III.21 Variabel Metode postComments

Nama Variabel	Isi Variabel	Default	Tipe Variabel
statusid	ID Dari status atau sapaan *wajib	Kosong	GET
text	Teks komentar yang dikirimkan. *wajib	Kosong	GET, POST

Hasil dari metode postComments yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online*

Android. Adapun respon data dari metode postComments, yaitu berisi id status yang merupakan id dari status yang telah dikomentari.

3. postLike

Metode postLikes merupakan metode yang digunakan oleh aplikasi *mobile* Kampus *Online* Android untuk mengirimkan *rating* bagus, jelek, atau menghapus *rating* yang pernah dikirimkan pada basis data Kampus *Online* yang ada saat ini.

Metode postLikes memiliki beberapa variabel yang dikirimkan sebagai parameter pada saat melakukan penambahan data *rating*. Berikut ini variabel yang dibutuhkan oleh metode postLikes dalam melakukan penambahan data *rating* pada basis data Kampus *Online* yang ada saat ini seperti terlihat pada Tabel III.22.

Tabel III.22 Variabel Metode postLikes

Nama Variabel	Isi Variabel	Default	Tipe Variabel
statusid	ID Dari status atau sapaan *wajib	Kosong	GET
text	Teks komentar yang dikirimkan. *wajib	Kosong	GET, POST

Hasil dari metode postLikes yaitu berupa data dengan format JSON yang didapat dari basis data Kampus *Online* yang ada saat ini. Data dengan format JSON tersebut nantinya akan diuraikan (*parsing*) oleh aplikasi Kampus *Online* Android. Adapun respon data dari metode postLikes, yaitu berisi id status yang merupakan id dari status yang telah diberikan *rating*.

3. 1. 7. 8. **Kode Error**

Kode *error* merupakan kode dari pesan-pesan ketika penggunaan Kampus API yang tidak sesuai. Berikut adalah referensi kode *error* pada respon seluruh metode Kampus API yang terlihat pada Tabel III.23, Tabel III.24, dan Tabel III.25.

Tabel III.23 Kode *Error*

Kode	Keterangan	Dalam Metode
0	**OK** Bila kode status bernilai 0, berarti operasi metode yang dilakukan berhasil dan tidak mengandung error.	**ALL**
1	**Metode yang dimaksud tidak didukung** Error ini akan tampil bila metode yang dimasukkan tidak didukung oleh Kampus API. Hal ini juga dapat terjadi bila kalkulasi signature metode tidak valid, atau next_token telah kadaluarsa.	**ALL**
2	**Session yang berjalan telah dalam keadaan *Login*** Error ini akan tampil bila Anda memanggil metode *login* ketika *user* telah melakukan *login*.	***login***
3	**Username *Login* Tidak Terdaftar** Error ini akan tampil bila Anda melakukan metode *login*, dan isi variable *username* tidak ditemukan oleh server sebagai *user* yang telah terdaftar.	***login***
4	**Password yang dimasukan tidak valid** Error ini akan tampil bila melakukan metode *login*, dan isi variabel token tidak cocok dengan kalkulasi password, *username* dan API_SECRET pada *user* yang dimaksud.	***login***
5	**Variabel [*username*(s)] harus diisi** Error ini akan tampil bila Anda lupa memasukan variabel *username* atau usernames.	**getUsersData** **getUserData**
6	**Tidak Ada Notifikasi** Error ini akan tampil bila metode getNotifications tidak menampilkan data apapun.	**getNotifications**

Tabel III.24 Kode *Error* (Lanjutan)

Kode	Keterangan	Dalam Metode
7	**Metode pertama harus berupa *LOGIN*** Error ini akan tampil bila Anda mencoba untuk memanggil metode selain *login* ketika session yang dimaksud belum melakukan *login* atau telah kadaluarsa.	**ALL**
8	**Variabel [statusid] Harus diisi dengan ID status** Error ini akan tampil bila variabel statusid tidak diisi pada metode getComments dan getLikes.	**getComments** **getLikes** **postComment** **postLike**
9	**ID Status yang dimaksud tidak valid** Error ini akan tampil bila id yang dikirimkan lewat statusid tidak valid (Tidak ada status dengan id yang dimaksud).	**getComments** **getLikes** **postComment** **postRate**
10	**Username yang dimaksud tidak terdaftar** Error ini akan tampil bila variabel ***username*** yang dikirimkan tidak terdapat dalam database Kampus *Online* Unikom.	**getFriends**
11	**User Target belum Anda ikuti statusnya.** Anda tidak dapat melakukan sapaan pada *user* yang belum diikuti statusnya.	**postStatus**
12	**Update Facebook atau Twitter tidak dapat dilakukan pada sapaan.** Variabel **update_facebook=1** atau **update_twitter=1** tidak dapat dilakukan bila variabel **target** juga di set.	**postStatus**
13	**Akun Anda tidak mendukung Update Facebook.** Error ini akan tampil bila menset variabel **update_facebook=1** tapi *user* tidak mendukung **support_facebook=1** pada respon *login*.	**postStatus**
14	**Akun Anda tidak mendukung Update Twitter.** Error ini akan tampil bila menset variabel **update_twitter=1** tapi *user* tidak mendukung **support_twitter=1** pada respon *login*.	**postStatus**

Tabel III.25 Kode *Error* (Lanjutan)

Kode	Keterangan	Dalam Metode
15	**Variabel GET/POST dengan nama [text] harus diisi** Error ini akan tampil bila tidak menset variabel **text** pada **postStatus** atau **postComment**.	**postStatus** **postComment**
16	**Variabel [rate] harus diisi dengan bagus, jelek atau hapus.** Error ini akan tampil bila tidak menset variabel **rate** dengan bagus, jelek atau hapus pada **postRate**	**postRate**
17	**Aktifitas Ilegal – Terlalu Cepat** Aktifitas Tidak Dapat Dilakukan karena *user* yang dimaksud baru saja melakukan aktifitas (contoh:*rating*). Untuk dapat melakukan aktifitas yang dimaksud, *user* harus menunggu setidaknya 10 menit.	**postRate**
18	**Session Tidak Valid** Session yang dimaksud tidak valid ketika mengakses metode checkSession	**checkSession**
19	**Method tidak didukung untuk *user* dengan tipe yang dimaksud.** Method dipanggil oleh tipe *user* yang tidak cocok. Misalnya dosen atau karyawan mengakses metode getNilai	**getNilai** **getAutodebet**
20	**Parameter yang dikirim tidak valid** Error ini akan tampil bila parameter yang dikirimkan tidak sesuai dengan ketentuan yang diminta oleh metode yang dimaksud.	**ALL**
21	**Request yang diminta berhasil tapi memiliki error** Detail error akan dikirimkan ke dalam data dengan kode dan pesan yang detail (Per metode).	**ALL**

3. 1. 8. Analisi Kebutuhan Nonfungsional

Analisis kebutuhan nonfungsional menggambarkan kebutuhan luar sistem yang diperlukan untuk menjalankan aplikasi yang dibangun. Adapun kebutuhan nonfungsional untuk menjalankan aplikasi *mobile* Kampus *Online* Unikom meliputi kebutuhan perangkat keras, kebutuhan perangkat lunak, dan pengguna yang akan memakai aplikasi. Analisis kebutuhan nonfungsional bertujuan agar aplikasi yang dibangun dapat digunakan sesuai dengan pengguna Kampus *Online* Unikom itu sendiri.

3. 1. 8. 1. **Perangkat Keras**

Analisis kebutuhan nonfungsional dari segi perangkat keras dibagi menjadi dua bagian, yaitu perangkat keras fitur pendungkung aplikasi dan jenis perangkat keras atau handset dari Android. Adapun penjelasan mengenai detail analisis perangkat keras untuk menjalankan aplikasi *mobile* Kampus *Online* Unikom adalah sebagai berikut:

3. 1. 8. 2. **Perangkat Keras Fitur**

Suatu aplikasi dapat berjalan dengan baik apabila didukung oleh perangkat keras. Adapun perangkat keras minimum yang dibutuhkan oleh aplikasi *mobile* Kampus *Online* Unikom agar dapat berjalan dengan baik dapat dilihat pada Tabel III.26.

Tabel III.26 Perangkat Keras Minimum

No	Perangkat Keras	Spesifikasi
1	Jaringan	GSM / CDMA
2	Data	GPRS / CDMA2000 1x
3	Layar	Touchscreen / Tidak Touchscreen
4	Baterai	Standar
5	Speaker	Standar

Selain spesifikasi perangkat keras minimum yang telah digambarkan pada Tabel III.26, aplikasi *mobile* Kampus *Online* Unikom dapat didukung dengan perangkat lainnya sehingga aplikasi dapat bekerja dengan maksimal. Adapun perangkat keras tambahan untuk menjalankan aplikasi *mobile* Kampus *Online* Unikom tertera pada Tabel III.27.

Tabel III.27 Perangkat Keras Tambahan

No	Perangkat Keras	Spesifikasi
1	Processor	500 Mhz
2	WLAN	Wifi
3	Data	HSDPA 3.6 / CDMA EV-DO
4	Layar	Accelerometer Sensor

1. Jenis Perangkat Keras Android

Jenis perangkat keras android dibagi menjadi dua tipe, yaitu *handphone* dan *tablet*. Sampai saati ini sistem operasi untuk *handphone* mulai dari sistem operasi Android versi 1.6 Donut sampai dengan versi 2.3 Gingerbread. Sedangkan untuk *tablet* mulai dari sistem operasi Android versi 2.1 Eclair sampai dengan Android versi 3.1 Honeycomb.

Aplikasi *mobile* Kampus *Online* Unikom yang dibangun pada dasarnya akan *support* pada kedua jenis perangkat tersebut. Terkecuali untuk perangkat Android tipe tablet dengan sistem operasi Android versi 3.0 keatas tidak akan *support* secara penuh. *Layout* dari aplikasi akan tertarik (*streach*) sehingga tidak terlihat proporsional seperti halnya pada tablet dengan sistem operasi dibawah Android. Hal tersebut dikarenakan sistem operasi Android versi 3.0 Honeycomb hanya khusus terpasang pada perangkat *tablet* dengan resolusi layar yang sangat tinggi. Sehingga agar aplikasi dapat berjalan dengan baik dibutuhkan *layout* aplikasi khusus yang mendukung *tablet* dengan resolusi tinggi.

3. 1. 8. 3. **Perangkat Lunak**

Aplikasi *mobile* Kampus *Online* Unikom yang dibangun membutuhkan perangkat lunak pendukung sehingga aplikasi tersebut dapat berjalan dengan baik. Adapun perangkat lunak minimum yang dibutuhkan untuk menjalankan aplikasi yang dibangun adalah sistem operasi Android versi 2.1 Eclair.

Android Eclair dipilih dikarenakan sistem operasi android yang sangat cepat berkembang sehingga agar aplikasi *mobile* Kampus *Online* Unikom masih dapat terus digunakan di sistem operasi Android selanjutnya, yaitu versi 2.2

Froyo, versi 2.3 Gingerbrad, versi 3.0 Honeycomb, bahkan versi 4.0 Ice Cream maka harus dipilih sistem operasi Android yang memiliki kesamaan lingkungan operasi. Kesamaan versi Android bisa dilihat dari keluarga sistem operasi Android yang terdiri dari dua keluarga, yaitu sistem operasi versi satu dengan anggota keluarga versi 1.5 Cupcake dan versi 1.6 Donut serta sistem operasi Android versi dua dengan anggota keluarga versi 2.1 Eclair, 2.2 Froyo, versi 2.3 Gingerbrad. Sehingga dengan dibangunnya aplikasi *mobile* Kampus *Online* Unikom pada versi Android 2.1 Eclair diharapkan bisa berjalan dengan baik di Android versi selanjutnya.

3. 1. 8. 4. **Pengguna**

Suatu aplikasi akan berjalan optimal apabila ditunjang oleh perangkat pikir yang memiliki kemampuan dalam menjalankan aplikasi yang bersangkutan. Pengguna jejaring sosial Kampus *Online* Unikom terdiri dari civitas Unikom, yaitu mahasiswa, karyawan, dosen, dan alumni. Berikut ini pemaparan dari setiap pengguna yang tertera pada Tabel III.28.

Tabel III.28 Pengguna jejaring sosial Kampus *Online* Unikom

No	Tipe	Umur Minimal	Pendidikan Terakhir
1	Mahasiswa	17 tahun	SMA
2	Dosen	23 tahun	Sarjana Strata 1
3	Karyawan	20 tahun	SMA
4	Alumni	22 tahun	Diploma 3

3. 1. 9. Analisi Kebutuhan Fungsional

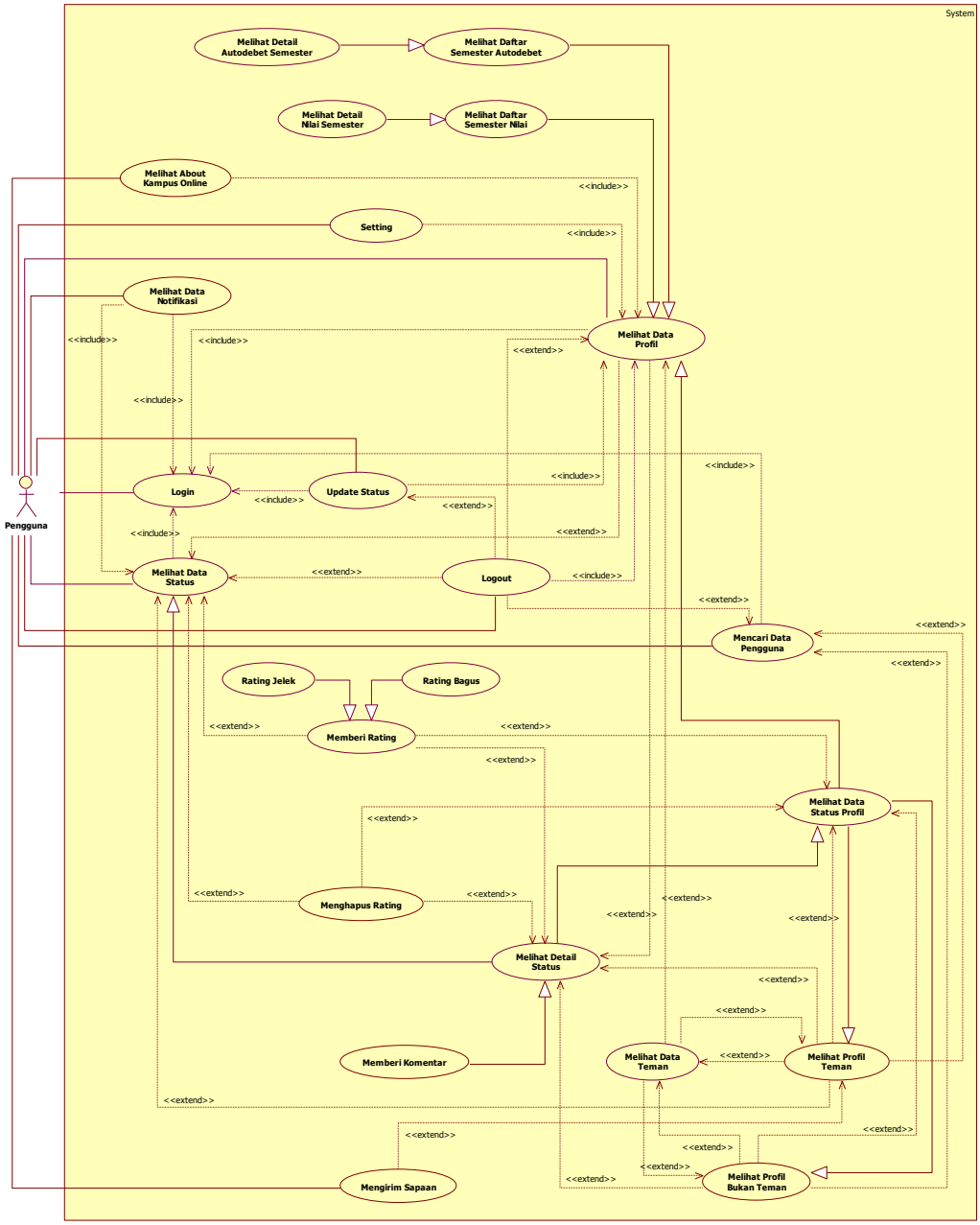
Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sistem yang akan dibangun sehingga sistem dapat berjalan dengan baik serta sesuai dengan kebutuhan proses bisnis.

Analisis dimodelkan dengan menggunakan UML (*Unified Modeling Language*). Tahap-tahap pemodelan dalam analisis menggunakan UML antara lain *use case diagaram*, *use case scenario*, *activity diagram*, *state diagram*, *squence diagram*, *dan class diagram*.

3. 1. 9. 1. ***Use Case Diagram***

Use Case Diagram merupakan konstruksi untuk mendeskripsikan hububungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada sistem. Sasaran pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional sistem dengan mendefinisikan skenario penggunaan sistem yang akan dibangun.

Use case diagram aplikasi Kampus *Online* berbasis Android merupakan gabungan *use case* hasil analisis sistem yang sedang berjalan dari *use case diagram* Kampus *Online* yang terdiri dari 8 *use case*, *use case diagram* Nilai *Online* yang terdiri dari 4 *use case,* dan *use case diagram* Autodebet *Online* yang terdiri dari 3 *use case*. Berikut ini *use case diagram* aplikasi Kampus *Online* Unikom yang dapat dilihat pada Gambar III.24.



Gambar III.24 *Use Case Diagram* Aplikasi *Mobile* Kampus *Online*

3. 1. 9. 2. ***Use Case Scenario***

Use Case Scenario merupakan deskripsi urutan langkah-langkah dalam proses bisnis baik yang dilakukan aktor terhadap sistem maupun yang dilakukan oleh sistem terhadap aktor. Berdasarkan *use case diagram* pada Gambar III.26 maka *use case scenario* untuk aplikasi Kampus *Online* Unikom terdiri dari skenario *login*, update status, mencari data pengguna, melihat data profil, melihat detail status, memberi komentar, memberi *rating*, melihat data status profil, melihat data teman, melihat daftar semester nilai, melihat detail nilai semester, melihat daftar semester autodebet, melihat detail autodebet semester, melihat informasi detail nilai, melihat keterangan mahasiswa, *setting*, *about*, dan *logout*. Berikut ini penjelasan dari masing-masing *use case scenario* yang terlihat pada Tabel III.29 sampai dengan Tabel III.54.

1. Use Case Scenario Login

Use case scenario login menjelaskan interaksi antara aktor pengguna dengan *use case login*. Dalam melakukan proses *login* terdapat aturan bisnis yang tercantum dalam Requirement A.1.

Requirement A.1
Data login terdiri dari *username* dan *password*. *Username* untuk mahasiswa merupakan NIM sedangkan untuk dosen dan karyawan tidak ada ketentuan karena tidak diambil dari NIP ataupun nomor absensi karyawan. Jika ingat *login* dipilih pada saat *login* maka pengguna akan tetap *login* pada aplikasi Kampus *Online* Android selama dua minggu. Jika ingat *login* tidak dipilih maka pengguna hanya akan *login* selama dua jam.

Adapun *use case scenario login* dapat dilihat pada Tabel III.29.

Tabel III.29 *Use Case Scenario Login*

Use case name	***Login***	
Related Requirements	Requirement A.1	
Goal In Context	Melakukan *login* ke dalam Kampus *Online*	
Preconditions	Form *login* Kampus *Online* ditampilkan	
Successful End Condition	Menampilkan *form home* Kampus *Online*	
Failed End Condition	Menampilkan pesan kesalahan *login*	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol *login*	
Included Case	-	
Base Case	-	
Main Flow	Step	Action
	1	Sistem menampilkan *form login* Kampus *Online*
	2	Pengguna mengisi data *login*
	3	Pengguna menekan tombol *login*
	4	Sistem melakukan validasi *field login*
	5	Sistem melakukan pengecekan ingat *login*
	6	Sistem melakukan autentikasi data *login*
Extension	Step	Branching Action
	3.1	*Username* dan *password* kosong
	3.2	*Username* kosong dan *password* terisi
	3.3	*Password* kosong
	3.4	*Username* dan *password* terisi
	5.1	Ingat *login*
	5.1	*Login* biasa
	6.1	*Username login* tidak terdatar
	6.2	*Password* yang dimasukan tidak valid

2. Use Case Scenario Update Status

Use case scenario update status menjelaskan interaksi antara aktor pengguna dengan *use case update status*. Dalam proses *update status* terdapat aturan bisnis yang tercantum dalam Requirement A.2.

Requirement A.2
Panjang karakter yang diterima untuk status adalah 400 karakter

Adapun *use case scenario update status* dapat dilihat pada Tabel III.30.

Tabel III.30 *Use Case Scenario Update Status*

Use case name	***Update Status***	
Related Requirements	Requirement A.2	
Goal In Context	Melakukan *update status* di Kampus *Online*	
Preconditions	Menampilkan *form* profil Kampus *Online*	
Successful End Condition	Menampilkan pesan status berhasil ditambahkan	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol *update status*	
Included Cases	***Login*, Melihat Data Profil**	
Base Case	-	
Main Flow	Step	Action
	1 **include::*Login***	Autentikasi *login*
	2	Sistem menampilkan *form* profil pengguna Kampus *Online*
	3	Pengguna menekan tombol *update status*
	4	Sistem menambilkan *form update status*
	5	Pengguna mengisi data status
	6	Pengguna menekan tombol *update status*
	7	Validasi *field* status
	8	Sistem melakukan penambahan data status
Extension	Step	Branching Action
	7.1	*Field* status kosong
	7.2	*Field* status terisi
	8.1	Penambahan data status berhasil
	8.2	Penambahan data status gagal
	8.3	Sistem akan *logout* jika *session* habis

3. *Use Case Scenario* Mencari Data Pengguna

Use case scenario mencari data pengguna menjelaskan interaksi antara aktor pengguna dengan *use case* mencari data pengguna. Dalam mencari data pengguna terdapat aturan bisnis yang tercantum dalam Requirement A.3.

Requirement A.3
Kata kunci untuk mencari data pengguna adalah *username* (untuk karyawan dan dosen) ataupun NIM (untuk mahasiswa)

Adapun *use case scenario* mencari data pengguna dapat dilihat pada Tabel III.31.

Tabel III.31 *Use Case Scenario* Mencari Data Pengguna

Use case name	**Mencari Data Pengguna**	
Related Requirements	Requirement A.3	
Goal In Context	Melakukan pencarian data pengguna Kampus *Online*	
Preconditions	Form pencarian pengguna Kampus *Online* ditampilkan	
Successful End Condition	Menampilkan data pengguna yang dicari	
Failed End Condition	Menampilkan pesan data pengguna tidak ditemukan	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol cari	
Included Case	***Login***	
Base Case	-	
	1 **include::*Login***	Autentikasi *login*
	2	Sistem menampilkan *form* pencarian data pengguna Kampus *Online*
	3	Pengguna mengisi *username* pengguna yang dicari
	4	Pengguna menekan tombol cari
	5	Validasi *field* cari
	6	Sistem melakukan pencarian data pengguna
Extension	Step	Branching Action
	5.1	*Field* cari kosong
	5.2	*Field* cari terisi
	6.1	Data pengguna ditemukan
	6.2	Data pengguna tidak ditemukan
	6.3	Sistem akan *logout* jika *session* habis

4. *Use Case Scenario* Melihat Data Profil

Use case scenario melihat data profil menjelaskan interaksi antara aktor pengguna dengan *use case* meliihat data profil. *Use case scenario* mencari data pengguna dapat dilihat pada Tabel III.32.

Tabel III.32 *Use Case Scenario* Melihat Data Profil

Use case name	**Melihat Data Profil**	
Related Requirements	-	
Goal In Context	Melihat profil pengguna Kampus *Online*	
Preconditions	*Form home* Kampus *Online* ditampilkan	
Successful End Condition	Menampilkan *form* profil	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih menu profil	
Included Case	***Login***	
Base Case	-	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form home* Kampus *Online*
	3	Pengguna menekan tombol menu
	4	Sistem menampilkan menu *home*
	5	Pengguna memilih menu profil
	6	Sistem melakukan pengambilan data profil
Extension	Step	Branching Action
	6.1	Pengambilan data berhasil
	6.2	Pengambilan data gagal
	6.3	Sistem akan *logout* jika *session* habis

5. *Use Case Scenario* Melihat Data Status

Use case scenario melihat data status menjelaskan interaksi antara aktor pengguna dengan *use case* melihat data status. *Use case scenario* melihat data status dapat dilihat pada Tabel III.33.

Tabel III.33 *Use Case Scenario* Melihat Data Status

Use case name	**Melihat Data Status**	
Related Requirements	Requirement A.3	
Goal In Context	Melihat data status pada halaman *home* Kampus *Online*	
Preconditions	Pengguna melakukan proses *login*	
Successful End Condition	Menampilkan data status	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna masuk ke *form home* atau pengguna memilih menu *refresh* di *form home*	
Included Case	***Login***	
Base Case	-	
Main Flow	Step	Action
	1 **include::*Login***	Autentikasi *login*
	2	Sistem menampilkan *form home*
	3	Pengguna meminta data status
	4	Sistem melakukan pegambilan data status
Extension	Step	Branching Action
	3.1	Pengguna masuk ke *form home* secara otomatis setelah melakukan proses *login*
	3.2	Pengguna masuk ke *form home* kemudian menekan tombol *refresh*
	4.1	Pengambilan data status berhasil
	4.2	Pengambilan data status gagal
	4.3	Sistem akan *logout* jika *session* habis

6. *Use Case Scenario* Melihat Data Notifikasi

Use case scenario melihat data notifikasi menjelaskan interaksi antara aktor pengguna dengan *use case* meliihat data notifikasi. *Use case scenario* melihat data notifikasi dapat dilihat pada Tabel III.34.

Tabel III.34 *Use Case Scenario* Melihat Data Notifikasi

Use case name	**Melihat Data Notifikasi**	
Related Requirements	-	
Goal In Context	Melihat data notifikasi	
Preconditions	*Form home* Kampus *Online* ditampilkan	
Successful End Condition	Menampilkan *sliding drawer* notifikasi	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol notifikasi	
Included Cases	***Login*, Melihat Data Status**	
Base Case	-	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2 **include::Melihat Data Status**	Menampilkan *form home* Kampus *Online*
	3	Pengguna menekan tombol notifikasi
	4	Sistem menampilkan *sliding drawer* notifikasi

7. *Use Case Scenario* Melihat Detail Status

Use case scenario melihat detail status menjelaskan interaksi antara aktor pengguna dengan *use case* melihat detail status. *Use case scenario* melihat detail status dapat dilihat pada Tabel III.35.

Tabel III.35 *Use Case Scenario* Melihat Detail Status

Use case name	**Melihat Detail Status**	
Related Requirements	-	
Goal In Context	Melihat detail dari status tertentu	
Preconditions	Menampilkan *form* status Kampus *Online*	
Successful End Condition	Menampilkan detail status yang dipilih	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih status	
Included Case	-	
Base Cases	**Melihat Data Status, Melihat Data Status Profil**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* status
	3	Pengguna memilih status
	4	Sistem melakukan pengambilan data status
Extension	Step	Branching Action
	2.1	Sistem menampilkan *form* status dari halaman *home* Kampus *Online*
	2.2	Sistem menampilkan *form* status dari halaman status profil
	4.1	Sistem akan *logout* jika *session* habis

8. *Use Case Scenario* Memberi Komentar

Use case scenario memberi komentar menjelaskan interaksi antara aktor pengguna dengan *use case* memberi komentar. Dalam proses memberi komentar terdapat aturan bisnis yang tercantum dalam Requirement A.4.

Requirement A.4
Panjang karakter yang diterima untuk komentar adalah 400 karakter

Adapun *use case scenario* memberi komentar dapat dilihat pada Tabel III.36.

Tabel III.36 *Use Case Scenario* Memberi Komentar

Use case name	**Memberi Komentar**	
Related Requirements	Requirement A.4	
Goal In Context	Menambahkan komentar dari status tertentu	
Preconditions	Menampilkan *form* status Kampus *Online*	
Successful End Condition	Menampilkan pesan komentar berhasil ditambahkan	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol kirim komentar	
Included Case	-	
Base Case	**Melihat Detail Status**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* detail status Kampus *Online*
	3	Pengguna mengisi komentar
	4	Pengguna menekan tombol kirim komentar
	5	Sistem menambahkan data komentar
Extension	Step	Branching Action
	5.1	Penambahan data komentar berhasil
	5.2	Penambahan data komentar gagal
	5.1	Sistem akan *logout* jika *session* habis

9. Use Case Scenario Memberi Rating

Use case scenario memberi *rating* menjelaskan interaksi antara aktor pengguna dengan *use case* memberi *rating*. *Use case scenario* memberi *rating* dapat dilihat pada Tabel III.37 dan Tabel III.38.

Tabel III.37 *Use Case Scenario* Memberi *Rating*

Use case name	**Memberi *Rating***	
Related Requirements	-	
Goal In Context	Menambahkan *rating* dari status tertentu	
Preconditions	Menampilkan status Kampus *Online*	
Successful End Condition	1. Menampilkan *rating* yang telah ditambahkan 2. Menampilkan pesan *rating* berhasil ditambahkan	
Failed End Condition	Menampilkan pesan kesalahan	
Primary Actor	Pengguna	
Trigger	Pengguna menekan pilihan menu *rating*	
Included Case	-	
Base Case	**Melihat Detail Status**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan status
	3	Pengguna memberi *rating*
	4	Sistem menambahkan *rating* pada status
Extension	Step	Branching Action
	2.1.a.i	Sistem menampilkan status dari halaman *home* Kampus *Online*
	2.1.a.ii	Sistem menampilkan status dari halaman status profil
	2.1.b	Pengguna memilih status yang akan diberi *rating*
	2.1.c	Sistem menampilkan *form* detail status yang dipilih
	2.1.d	Pengguna menekan tombol menu
	2.1.e	Sistem menampilkan menu status

Tabel III.38 *Use Case Scenario* Memberi *Rating* (Lanjutan)

Extension	Step	Branching Action
	2.1.f	Pengguna memilih menu *rating*
	2.1.g	Sistem menampilkan pilihan *rating*
	2.2	Sistem menampilkan status dari halaman detail status
	2.2.a	Pengguna menekan tombol menu
	2.2.b	Sistem menampilkan menu status
	2.2.c	Pengguna memilih menu *rating*
	2.2.d	Sistem menampilkan pilihan *rating*
	4.1	Penambahan *rating* berhasil
	4.2	Penambahan *rating* gagal
	4.3	Sistem akan *logout* jika *session* habis

10. *Use Case Scenario* Menghapus Rating

Use case scenario menghapus *rating* menjelaskan interaksi antara aktor pengguna dengan *use case* menghapus *rating*. Dalam proses menghapus *rating* terdapat aturan bisnis yang tercantum dalam Requirement A.5.

Requirement A.5
Dalam status yang sama, pengguna harus menunggu 10 menit untuk menghapus rating yang baru saja ditambahkan.

Adapun *use case scenario* menghapus *rating* dapat dilihat pada Tabel III.39 dan Tabel III.40.

Tabel III.39 *Use Case Scenario* Memberi *Rating*

Use case name	**Menghapus *Rating***	
Related Requirements	Requirement A.5	
Goal In Context	Menambahkan *rating* dari status tertentu	
Preconditions	Menampilkan status Kampus *Online*	
Successful End Condition	Menampilkan pesan *rating* berhasil dihapus	
Failed End Condition	Menampilkan pesan kesalahan	
Primary Actor	Pengguna	
Trigger	Pengguna menekan pilihan menu *rating*	
Included Case	-	
Base Case	**Melihat Detail Status**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan status
	3	Pengguna menghapus *rating*
	4	Sistem menghapus *rating* pada status
Extension	Step	Branching Action
	2.1.a.i	Sistem menampilkan status dari halaman *home* Kampus *Online*
	2.1.a.ii	Sistem menampilkan status dari halaman status profil
	2.1.b	Pengguna memilih status yang akan diberi *rating*
	2.1.c	Sistem menampilkan *form* detail status yang dipilih
	2.1.d	Pengguna menekan tombol menu
	2.1.e	Sistem menampilkan menu status
	2.1.f	Pengguna memilih menu *rating*
	2.1.g	Sistem menampilkan hapus *rating*
	2.2	Sistem menampilkan status dari halaman detail status
	2.2.a	Pengguna menekan tombol menu
	2.2.b	Sistem menampilkan menu status
	2.2.c	Pengguna memilih menu *rating*
	2.2.d	Sistem menampilkan hapus *rating*

Tabel III.40 *Use Case Scenario* Memberi *Rating* (Lanjutan)

Extension	Step	Branching Action
	4.1	Penambahan *rating* berhasil
	4.2	Penambahan *rating* gagal
	4.3	Sistem akan *logout* jika *session* habis

11. *Use Case Scenario* Melihat Data Status Profil

Use case scenario melihat data status profil menjelaskan interaksi antara aktor pengguna dengan *use case* melihat data status profil. *Use case scenario* melihat data status profil dapat dilihat pada Tabel III.41.

Tabel III.41 *Use Case Scenario* Melihat Data Status Profil

Use case name	**Melihat Data Status Profil**	
Related Requirements	-	
Goal In Context	Melihat data status profil Kampus *Online*	
Preconditions	Menampilkan *form* status profil Kampus *Online*	
Successful End Condition	Menampilkan *form* status profil	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol status	
Included Case	-	
Base Cases	**Melihat Data Profil, Melihat Profil Teman, Melihat Profil Bukan Teman**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* profil pengguna Kampus *Online*
	3	Pengguna menekan tombol status
	4	Sistem mengambil data status profil
Extension	Step	Branching Action
	4.1	Sistem akan *logout* jika *session* habis

12. *Use Case Scenario* Melihat Data Teman

Use case scenario melihat data teman menjelaskan interaksi antara aktor pengguna dengan *use case* melihat data teman. *Use case scenario* melihat data teman dapat dilihat pada Tabel III.42.

Tabel III.42 *Use Case Scenario* Melihat Data Teman

Use case name	**Melihat Data Teman**	
Related Requirements	-	
Goal In Context	Melihat data profil teman	
Preconditions	Menampilkan *form* profil Kampus *Online*	
Successful End Condition	Menampilkan *form* teman	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol teman	
Included Case	-	
Base Case	**Melihat Data Profil**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* profil Kampus *Online*
	3	Pengguna menekan tombol teman
	4	Sistem mengambil data teman
Extension	Step	Branching Action
	4.1	Sistem akan *logout* jika *session* habis

13. *Use Case Scenario* Melihat Profil Teman

Use case scenario melihat profil teman menjelaskan interaksi antara aktor pengguna dengan *use case* meliihat profil teman. *Use case scenario* melihat profil teman dapat dilihat pada Tabel III.43.

Tabel III.43 *Use Case Scenario* Melihat Profil Teman

Use case name	**Melihat Profil Teman**	
Related Requirements	-	
Goal In Context	Melihat profil teman Kampus *Online*	
Preconditions	Menampilkan daftar teman	
Successful End Condition	Menampilkan *form* profil teman	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih teman	
Included Case	-	
Base Case	-	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan daftar teman
	3	Pengguna memilih teman
	4	Sistem melakukan pengambilan data profil teman
Extension	Step	Branching Action
	2.1	Sistem menampilkan daftar teman dari halaman teman Kampus *Online*
	2.2	Sistem menampilkan daftar teman dari halaman cari Kampus *Online*
	4.1	Pengambilan data teman berhasil
	4.2	Pengambilan data teman gagal
	4.3	Sistem akan *logout* jika *session* habis

14. *Use Case Scenario* Melihat Profil Bukan Teman

Use case scenario melihat profil bukan teman menjelaskan interaksi antara aktor pengguna dengan *use case* meliihat profil bukan teman. *Use case scenario* melihat profil bukan teman dapat dilihat pada Tabel III.44.

Tabel III.44 *Use Case Scenario* Melihat Profil Bukan Teman

Use case name	**Melihat Profil Bukan Teman**	
Related Requirements	-	
Goal In Context	Melihat profil bukan teman Kampus *Online*	
Preconditions	Menampilkan daftar pengguna Kampus *Online*	
Successful End Condition	Menampilkan *form* profil bukan teman	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih pengguna bukan teman	
Included Case	***Login***	
Base Case	-	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan daftar pengguna
	3	Pengguna memilih pengguna bukan teman
	4	Sistem melakukan pengambilan data profil bukan teman
Extension	Step	Branching Action
	2.1	Sistem menampilkan pengguna bukan teman di *form* teman Kampus *Online*
	2.2	Sistem menampilkan pengguna bukan teman di *form* cari Kampus *Online*
	41	Pengambilan data bukan teman berhasil
	4.2	Pengambilan data bukan teman gagal
	4.3	Sistem akan *logout* jika *session* habis

15. Use Case Scenario Mengirim Sapaan

Use case scenario mengirim sapaan menjelaskan interaksi antara aktor pengguna dengan *use case* mengirim sapaan. Dalam proses *update status* terdapat aturan bisnis yang tercantum dalam Requirement A.6.

Requirement A.6
Panjang karakter yang diterima untuk sapaan adalah 400 karakter

Adapun *use case scenario* mengirim sapaan dapat dilihat pada Tabel III.45.

Tabel III.45 *Use Case Scenario* Mengirim Sapaan

Use case name	**Mengirim Sapaan**	
Related Requirements	Requirement A.6	
Goal In Context	Mengirim sapaan kepada teman	
Preconditions	Menampilkan *form* profil Kampus *Online*	
Successful End Condition	Menampilkan pesan sapaan berhasil dikirimkan	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol kirim	
Included Case	-	
Base Case	-	
Main Flow	Step	Action
	1 **include::*Login***	Autentikasi *login*
	2	Sistem menampilkan *form* profil pengguna Kampus *Online*
	3	Pengguna menekan tombol kirim sapaan
	4	Sistem menampilkan *form* sapaan Kampus *Online*
	5	Pengguna mengisi data sapaan
	6	Pengguna menekan tombol kirim
	7	Validasi field sapaan
	8	Sistem melakukan penambahan data sapaan
Extension	Step	Branching Action
	7.1	*Field* sapaan kosong
	7.2	*Field* sapaan terisi
	8.1	Penambahan data sapaan berhasil
	8.2	Penambahan data sapaan gagal
	8.3	Sistem akan *logout* jika *session* habis

16. *Use Case Scenario* Melihat Daftar Semester Nilai

Use case scenario melihat daftar semester nilai menjelaskan interaksi antara aktor pengguna dengan *use case* melihat daftar semester nilai. *Use case scenario* melihat daftar semester nilai dapat dilihat pada Tabel III.46.

Tabel III.46 *Use Case Scenario* Melihat Daftar Semester Nilai

Use case name	**Melihat Daftar Semester Nilai**	
Related Requirements	-	
Goal In Context	Melihat daftar semester nilai yang telah ditempuh	
Preconditions	Menampilkan *form* profil Kampus *Online*	
Successful End Condition	Menampilkan *form* daftar semester nilai	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol nilai	
Included Case	-	
Base Case	**Melihat Data Profil**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* profil Kampus *Online*
	3	Pengguna menekan tombol nilai
	4	Sistem mengambil daftar semester nilai
Extension	Step	Branching Action
	4.1	Sistem akan *logout* jika *session* habis

17. *Use Case Scenario* Melihat Detail Nilai Semester

Use case scenario melihat detail nilai semester menjelaskan interaksi antara aktor pengguna dengan *use case* melihat detail nilai semester. *Use case scenario* melihat detail nilai semester dapat dilihat pada Tabel III.47.

Tabel III.47 *Use Case Scenario* Melihat Detail Nilai Semester

Use case name	**Melihat Detail Nilai Semester**	
Related Requirements	-	
Goal In Context	Melihat detail nilai dari semester yang telah dipilih	
Preconditions	Menampilkan *form* daftar semester yang telah ditempuh	
Successful End Condition	Menampilkan detail nilai akhir dari semester yang dipilih	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih semester	
Included Case	-	
Base Case	**Melihat Daftar Semester Nilai**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* daftar semester yang telah ditempuh
	3	Pengguna memilih semester
	4	Sistem mengambil detail nilai semester
Extension	Step	Branching Action
	4.1	Sistem akan *logout* jika *session* habis

18. *Use Case Scenario* Melihat Informasi Detail Nilai

Use case scenario melihat informasi detail nilai menjelaskan interaksi antara aktor pengguna dengan *use case* melihat informasi detail nilai. *Use case scenario* melihat informasi detail nilai dapat dilihat pada Tabel III.48.

Tabel III.48 *Use Case Scenario* Melihat Informasi Detail Nilai

Use case name	**Melihat Informasi Detail Nilai**	
Related Requirements	-	
Goal In Context	Melihat informasi detail nilai yang telah dipilih	
Preconditions	Menampilkan detail nilai akhir dari semester yang dipilih	
Successful End Condition	Menampilkan informasi detail nilai yang telah dipilih	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih nilai	
Included Case	-	
Base Case	**Melihat Detail Nilai Semester**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* detail nilai akhir semester yang telah ditempuh
	3	Pengguna memilih nilai
	4	Sistem menampilkan informasi detail nilai yang dipilih

19. *Use Case Scenario* Melihat Keterangan Mahasiswa

Use case scenario melihat keterangan mahasiswa menjelaskan interaksi antara aktor pengguna dengan *use case* melihat keterangan mahasiswa. *Use case scenario* melihat keterangan mahasiswa dapat dilihat pada Tabel III.49.

Tabel III.49 *Use Case Scenario* Melihat Keterangan Mahasiswa

Use case name	**Melihat Keterangan Mahasiswa**	
Related Requirements	-	
Goal In Context	Melihat informasi detail nilai yang telah dipilih	
Preconditions	Menampilkan detail nilai akhir dari semester yang dipilih	
Successful End Condition	Menampilkan informasi keterangan mahasiswa dari pengguna Kampus *Online*	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol detail	
Included Case	-	
Base Case	**Melihat Detail Nilai Semester**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* detail nilai akhir semester yang telah ditempuh
	3	Pengguna menekan tombol detail
	4	Sistem menampilkan informasi keterangan mahasiswa pengguna Kampus *Online*

20. *Use Case Scenario* Melihat Daftar Semester Autodebet

Use case scenario melihat daftar semester autodebet menjelaskan interaksi antara aktor pengguna dengan *use case* melihat daftar semester autodebet. *Use case scenario* melihat daftar semester autodebet dapat dilihat pada Tabel III.50.

Tabel III.50 *Use Case Scenario* Melihat Daftar Semester Autodebet

Use case name	**Melihat Daftar Semester Autodebet**	
Related Requirements	-	
Goal In Context	Melihat daftar semester autodebet yang telah ditempuh	
Preconditions	Menampilkan *form* profil Kampus *Online*	
Successful End Condition	Menampilkan *form* daftar semester autodebet	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol autodebet	
Included Case	-	
Base Case	**Melihat Data Profil**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* profil Kampus *Online*
	3	Pengguna menekan tombol autodebet
	4	Sistem mengambil daftar semester autodebet
Extension	Step	Branching Action
	4.1	Sistem akan *logout* jika *session* habis

21. *Use Case Scenario* Melihat Detail Autodebet Semester

Use case scenario melihat detail autodebet semester menjelaskan interaksi antara aktor pengguna dengan *use case* melihat detail autodebet semester. *Use case scenario* melihat detail autodebet semester dapat dilihat pada Tabel III.51.

Tabel III.51 *Use Case Scenario* Melihat Detail Autodebet Semester

Use case name	**Melihat Detail Autodebet Semester**	
Related Requirements	-	
Goal In Context	Melihat detail autodebet dari semester yang telah dipilih	
Preconditions	Menampilkan *form* daftar autodebet yang telah ditempuh	
Successful End Condition	Menampilkan detail autodebet dari semester yang dipilih	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih semester	
Included Case	-	
Base Case	**Melihat Daftar Semester Autodebet**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2	Sistem menampilkan *form* daftar semester yang telah ditempuh
	3	Pengguna memilih semester
	4	Sistem mengambil detail autodebet semester
Extension	Step	Branching Action
	4.1	Sistem akan *logout* jika *session* habis

22. *Use Case Scenario Setting*

Use case scenario setting menjelaskan interaksi antara aktor pengguna dengan *use case setting*. *Use case scenario setting* dapat dilihat pada Tabel III.52.

Tabel III.52 *Use Case Scenario Setting*

Use case name	***Setting***	
Related Requirements	-	
Goal In Context	Melihat halaman pengaturan aplikasi Kampus *Online*	
Preconditions	Menampilkan *form* profil Kampus *Online*	
Successful End Condition	Menampilkan *form setting* Kampus *Online*	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih menu *setting*	
Included Cases	**Melihat Data Profil**	
Base Case	-	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2 **include::Melihat Data Profil**	Menampilkan halaman data profil
	3	Sistem menampilkan *form* profil Kampus *Online*
	4	Pengguna menekan tombol menu
	5	Sistem menampilkan menu profil
	6	Pengguna memilih menu *setting*
	7	Sistem menampilkan *form setting* Kampus *Online*

23. *Use Case Scenario About*

Use case scenario about menjelaskan interaksi antara aktor pengguna dengan *use case about*. *Use case scenario about* dapat dilihat pada Tabel III.53.

Tabel III.53 ***Use Case Scenario About***

Use case name	***About***	
Related Requirements	-	
Goal In Context	Melihat penjelasan tentang aplikasi Kampus *Online*	
Preconditions	Menampilkan *form* profil Kampus *Online*	
Successful End Condition	Menampilkan *form about* Kampus *Online*	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih menu *about*	
Included Cases	**Melihat Data Profil**	
Main Flow	Step	Action
	1 **include::Login**	Autentikasi *login*
	2 **include::Melihat Data Profil**	Menampilkan halaman data profil
	3	Sistem menampilkan *form* profil Kampus *Online*
	4	Pengguna menekan tombol menu
	5	Sistem menampilkan menu profil
	6	Pengguna memilih menu *about*
	7	Sistem menampilkan *form about* Kampus *Online*

24. *Use Case Scenario Logout*

Use case scenario logout menjelaskan interaksi antara aktor pengguna dengan *use case logout*. *Use case scenario logout* dapat dilihat pada Tabel III.54.

Tabel III.54 *Use Case Scenario Logout*

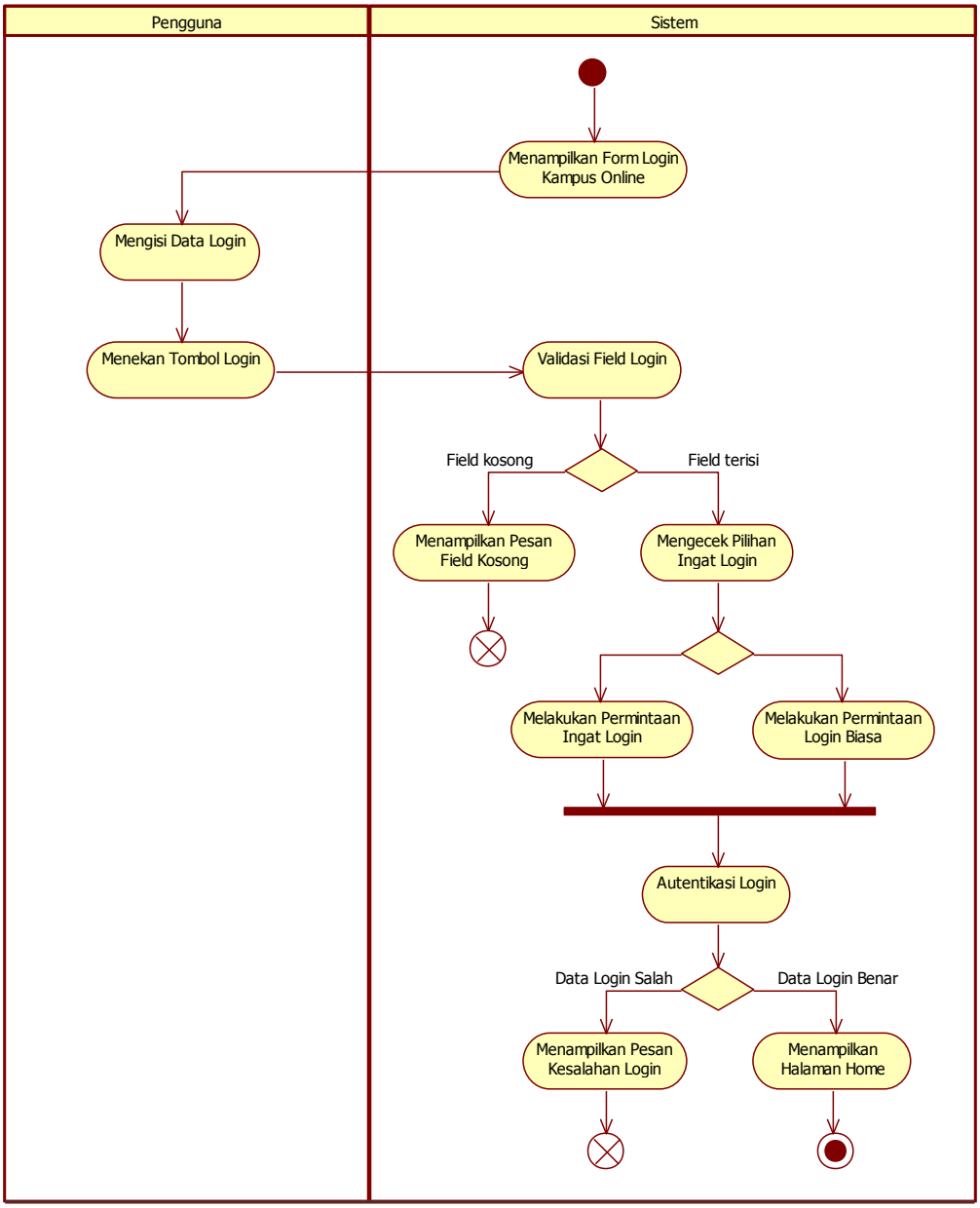
Use case name	***Logout***	
Related Requirements		
Goal In Context	Keluar dari aplikasi Kampus *Online*	
Preconditions	Menampilkan *form* profil Kampus *Online*	
Successful End Condition	Menampilkan *form login* Kampus *Online*	
Failed End Condition	-	
Primary Actor	Pengguna	
Trigger	Pengguna memilih menu *logout*	
Included Cases	**Melihat Data Profil**	
Base Case	-	
Main Flow	Step	Action
	1	Sistem menampilkan *form* profil Kampus *Online*
	2	Pengguna menekan tombol menu
	3	Sistem menampilkan menu profil
	4	Pengguna memilih menu *logout*
	5	Sistem menampilkan pesan konfirmasi *logout*
	6	Pengguna memasukkan pilihan konfirmasi *logout*
	7	Sistem melakukan pengecekkan pilihan konfirmasi *logout*
	8	Sistem melakukan *logout* aplikasi
Extension	Step	Branching Action
	7.1	Pengguna mengkonfirmasi *logout*
	7.2	Pengguna membatalkan *logout* aplikasi

3. 1. 9. 3. ***Activity Diagram***

Activity diagram merupakan diagram yang memodelkan aliran kerja atau *workflow* dari urutan aktifitas dalam suatu proses yang mengacu pada *use case diagram* yang ada. *Activity diagram* pada aplikasi Kampus *Online* Unikom terdiri dari *activity diagram login*, update status, mencari data pengguna lain, melihat data profil, melihat detail status, memberi komentar, memberi *rating*, melihat data status profil, melihat data teman, melihat daftar semester nilai, melihat detail nilai semester, melihat daftar semester autodebet, melihat detail autodebet semester, melihat informasi detail nilai, melihat keterangan mahasiswa, *setting*, *about*, dan *logout*. Berikut ini penjelasan dari masing-masing *activity diagram* yang terlihat pada Gambar III.25 sampai dengan Gambar III.48.

1. *Activity Diagram Login*

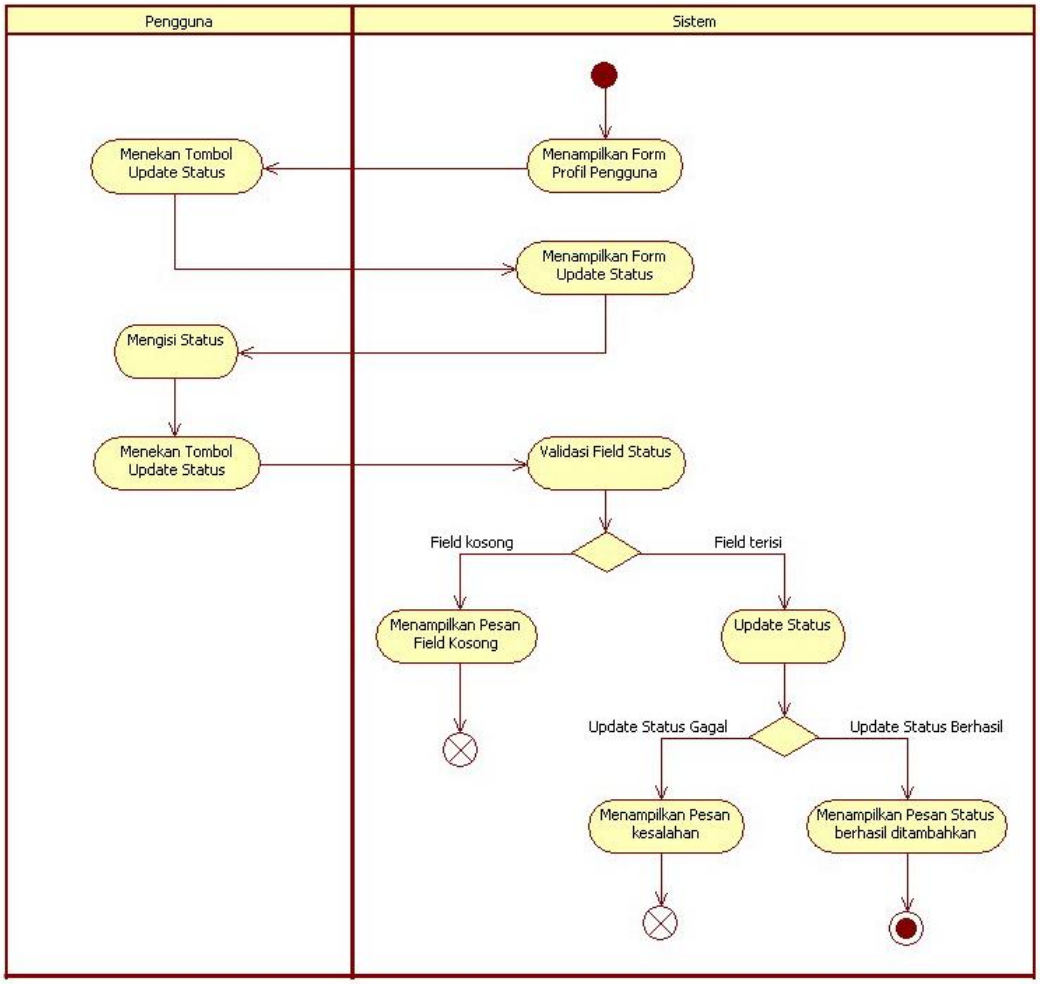
Activity diagram login menjelaskan aliran kerja aktor pengguna pada saat *login* ke dalam aplikasi *mobile* Kampus *Online*. *Activity diagram login* dapat dilihat pada Gambar III.25.



Gambar III.25 *Activity Diagram Login*

2. *Activity Diagram Update Status*

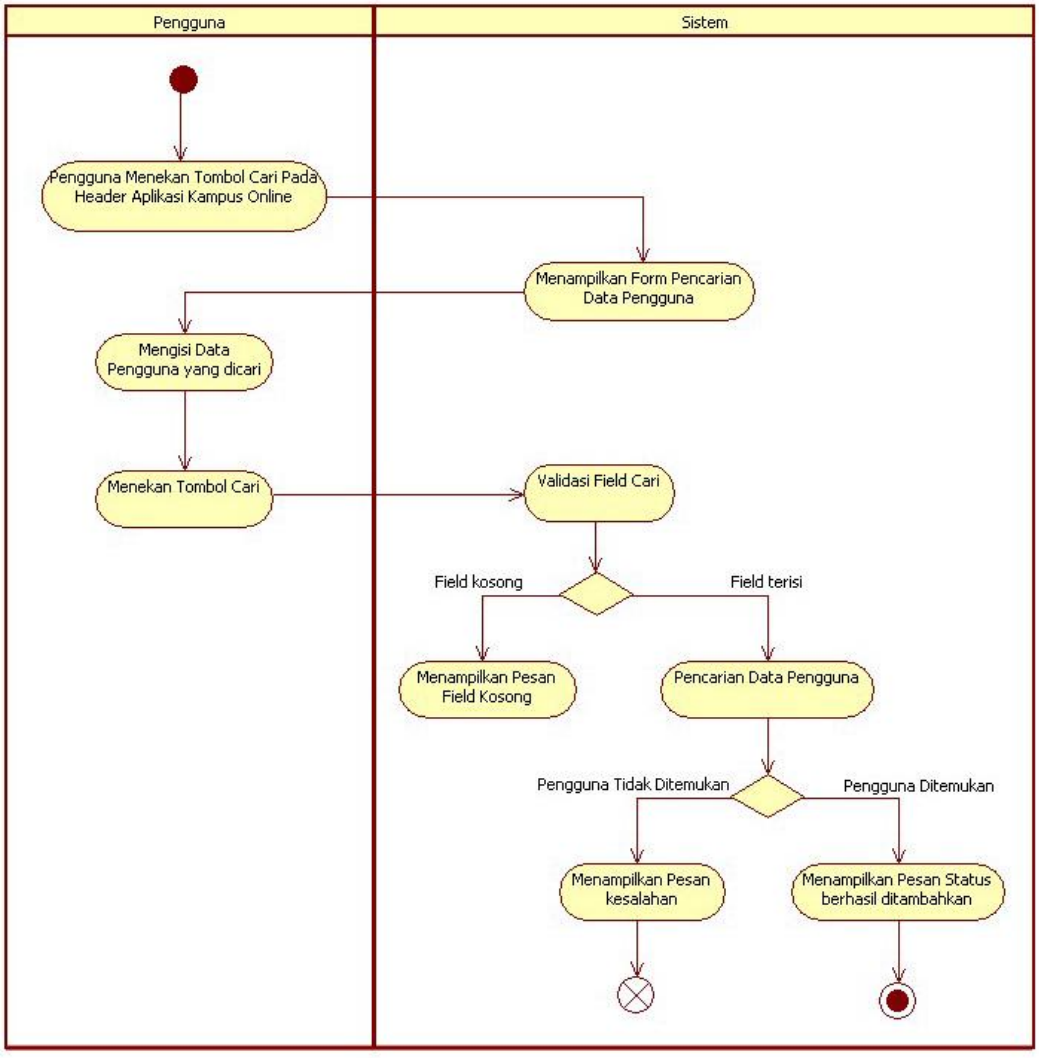
Activity diagram update status menjelaskan aliran kerja aktor pengguna untuk melakukan *update status* pada aplikasi *mobile* Kampus *Online*. *Activity diagram update status* dapat dilihat pada Gambar III.26.



Gambar III.26 ***Activity Diagram Update Status***

3. *Activity Diagram* Mencari Data Pengguna

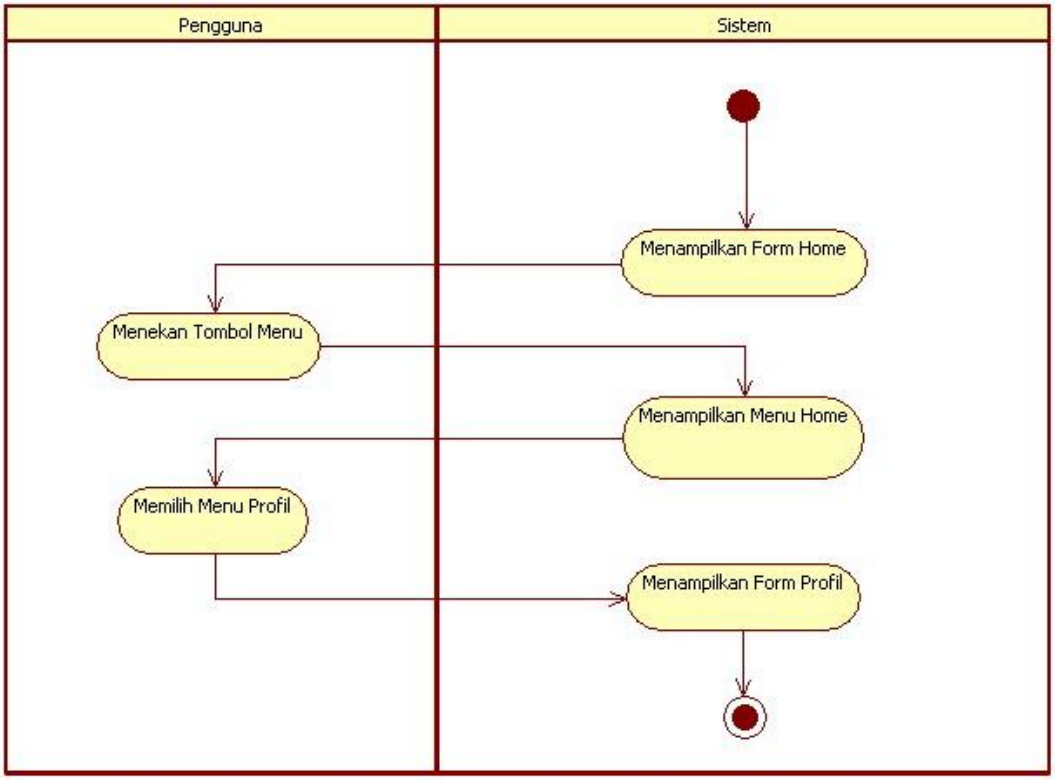
Activity diagram mencari data pengguna menjelaskan aliran kerja aktor pengguna pada saat mencari data pengguna lain di aplikasi *mobile* Kampus *Online*. *Activity diagram* mencari data pengguna lain dapat dilihat pada Gambar III.27.



Gambar III.27 *Activity Diagram* Mencari Data Pengguna Lain

4. *Activity Diagram* Melihat Data Profil

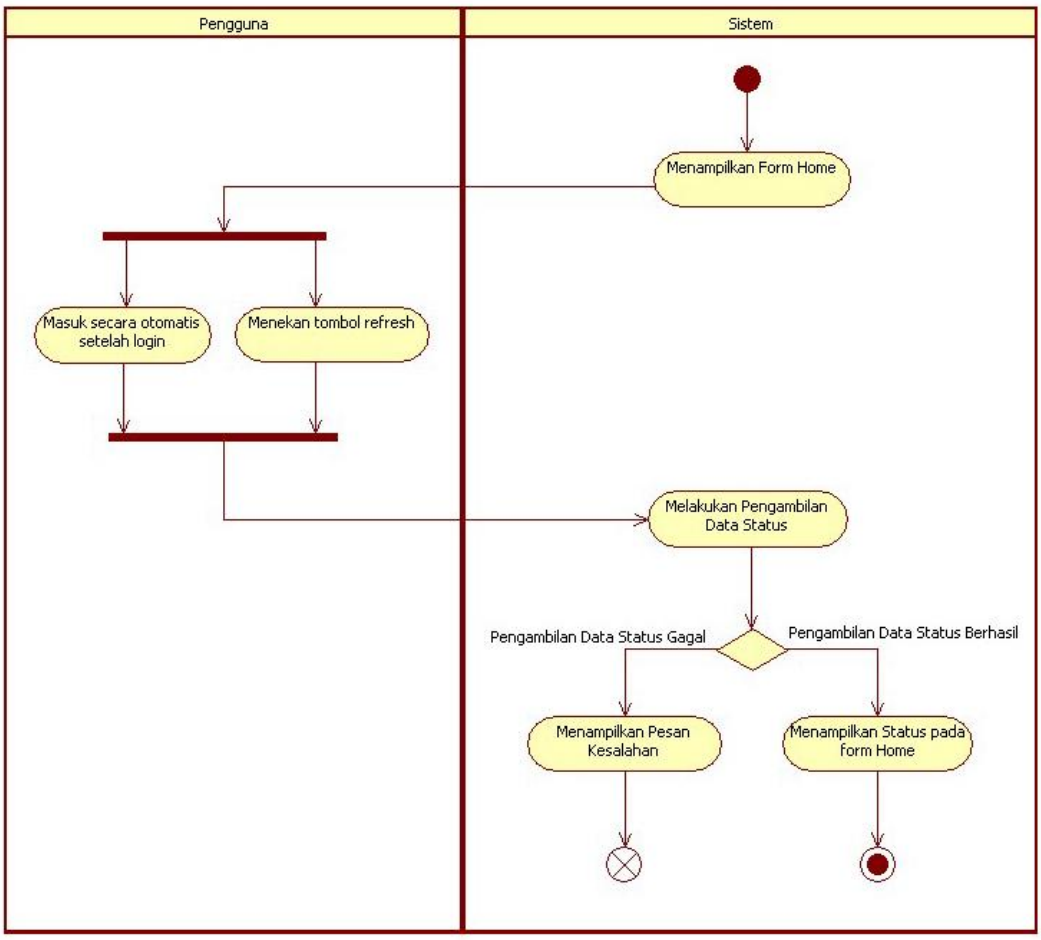
Activity diagram melihat data profil menjelaskan aliran kerja aktor pengguna untuk melihat data profil di aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat data profil dapat dilihat pada Gambar III.28.



Gambar III.28 *Activity Diagram* Melihat Data Profil

5. *Activity Diagram* Melihat Data Status

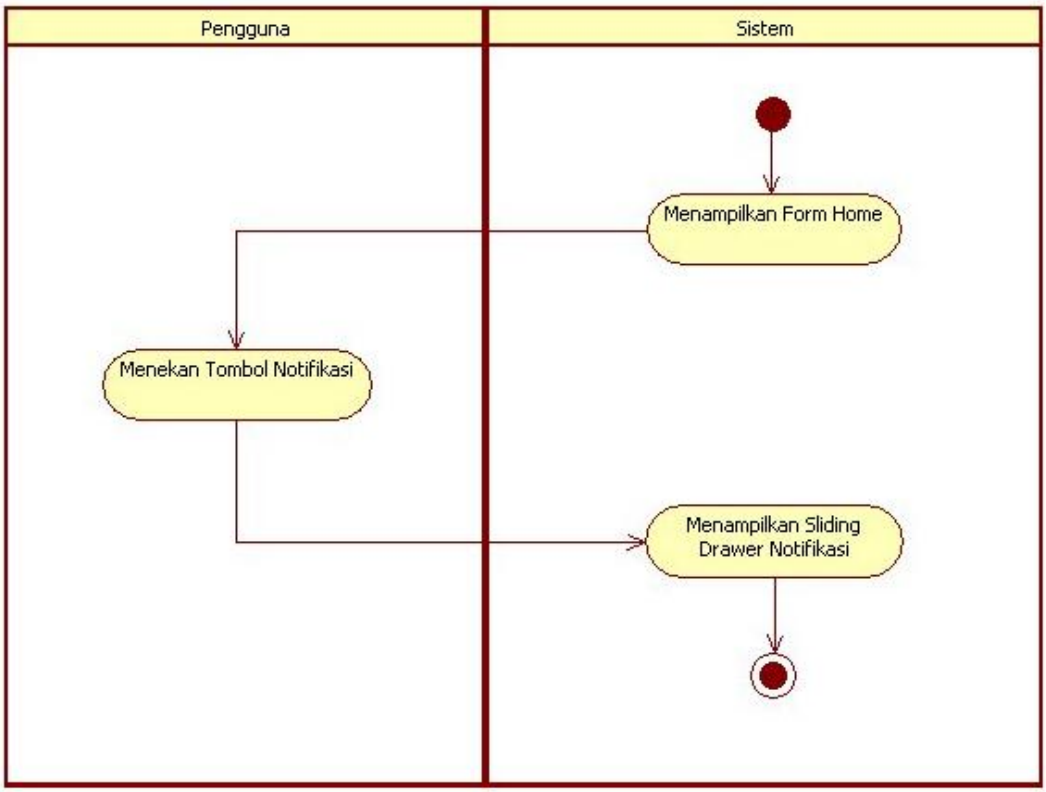
Activity diagram melihat data status menjelaskan aliran kerja aktor pengguna untuk melihat data status pada aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat data status dapat dilihat pada Gambar III.29.



Gambar III.29 *Activity Diagram* Melihat Data Status

6. *Activity Diagram* Melihat Data Notifikasi

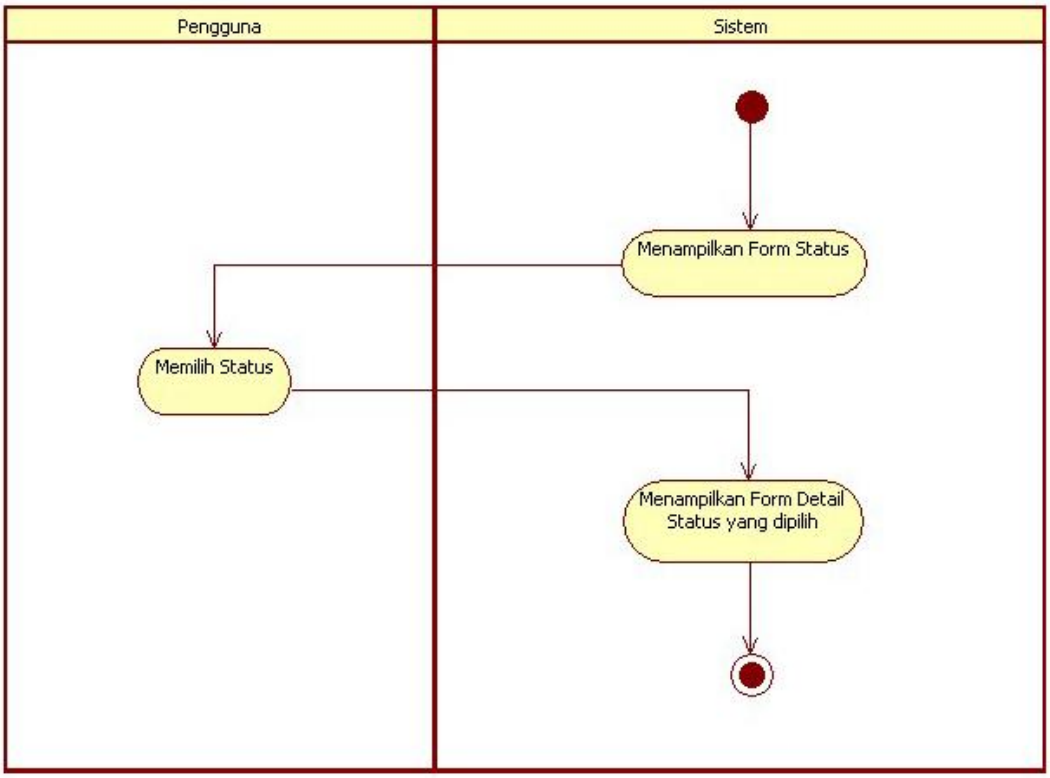
Activity diagram melihat data notifikasi menjelaskan aliran kerja aktor pengguna untuk melihat data notifikasi pada aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat data status dapat dilihat pada Gambar III.30.



Gambar III.30 *Activity Diagram* Melihat Data Notifikasi

7. *Activity Diagram* Melihat Detail Status

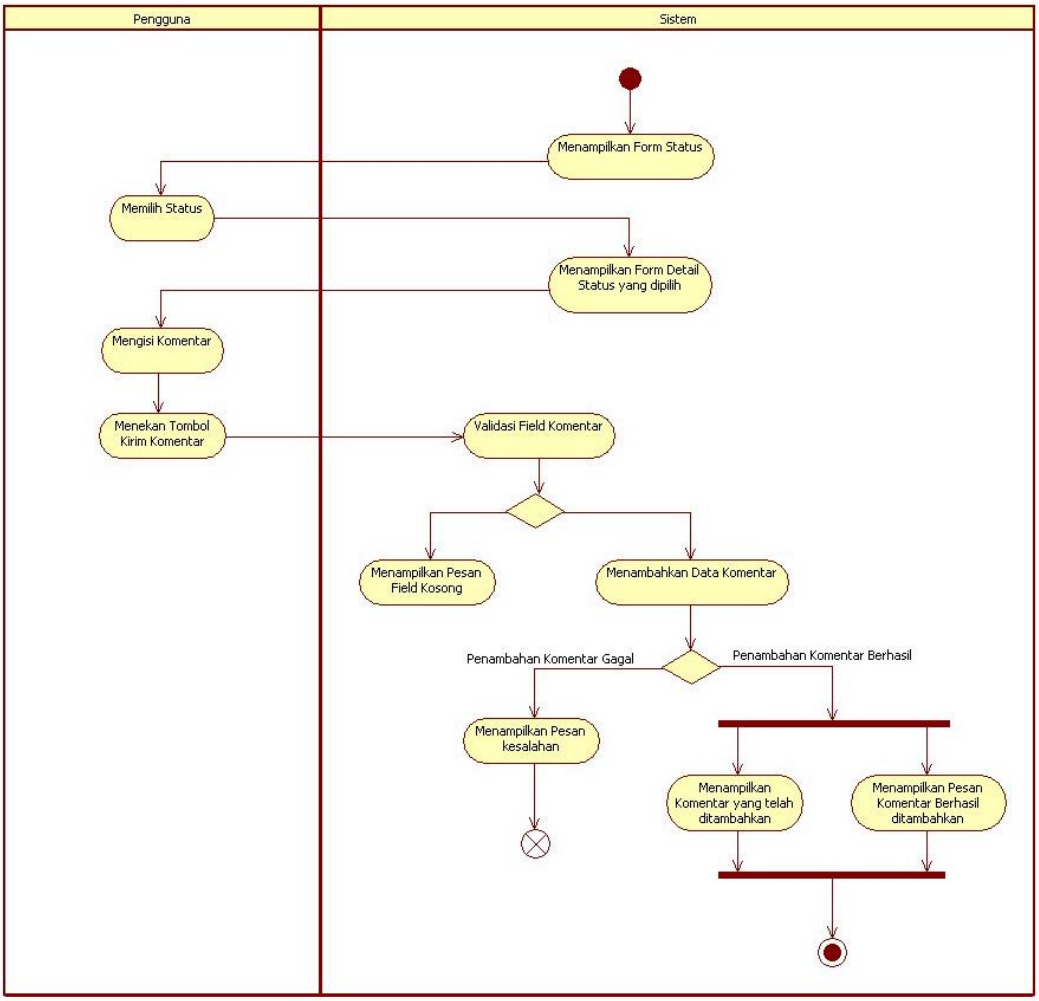
Activity diagram melihat detail status menjelaskan aliran kerja aktor pengguna untuk melihat detail status di aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat detail status dapat dilihat pada Gambar III.31.



Gambar III.31 *Activity Diagram* Melihat Detail Status

8. *Activity Diagram* Memberi Komentar

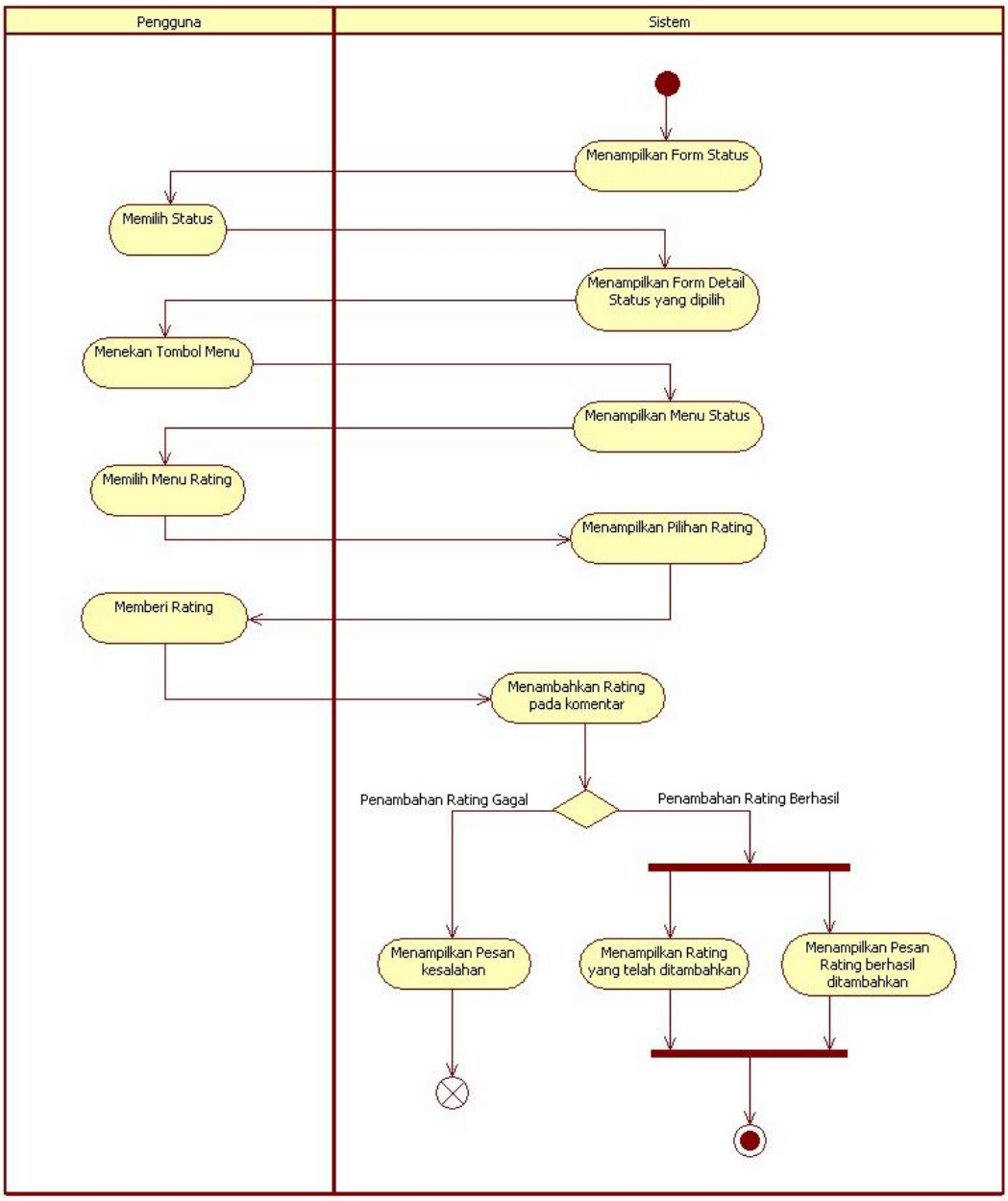
Activity diagram memberi komentar menjelaskan aliran kerja aktor pengguna pada saat memberi komentar status di aplikasi *mobile* Kampus *Online*. *Activity diagram* memberi komentar dapat dilihat pada Gambar III.32.



Gambar III.32 *Activity Diagram* Memberi Komentar

9. *Activity Diagram* Memberi *Rating*

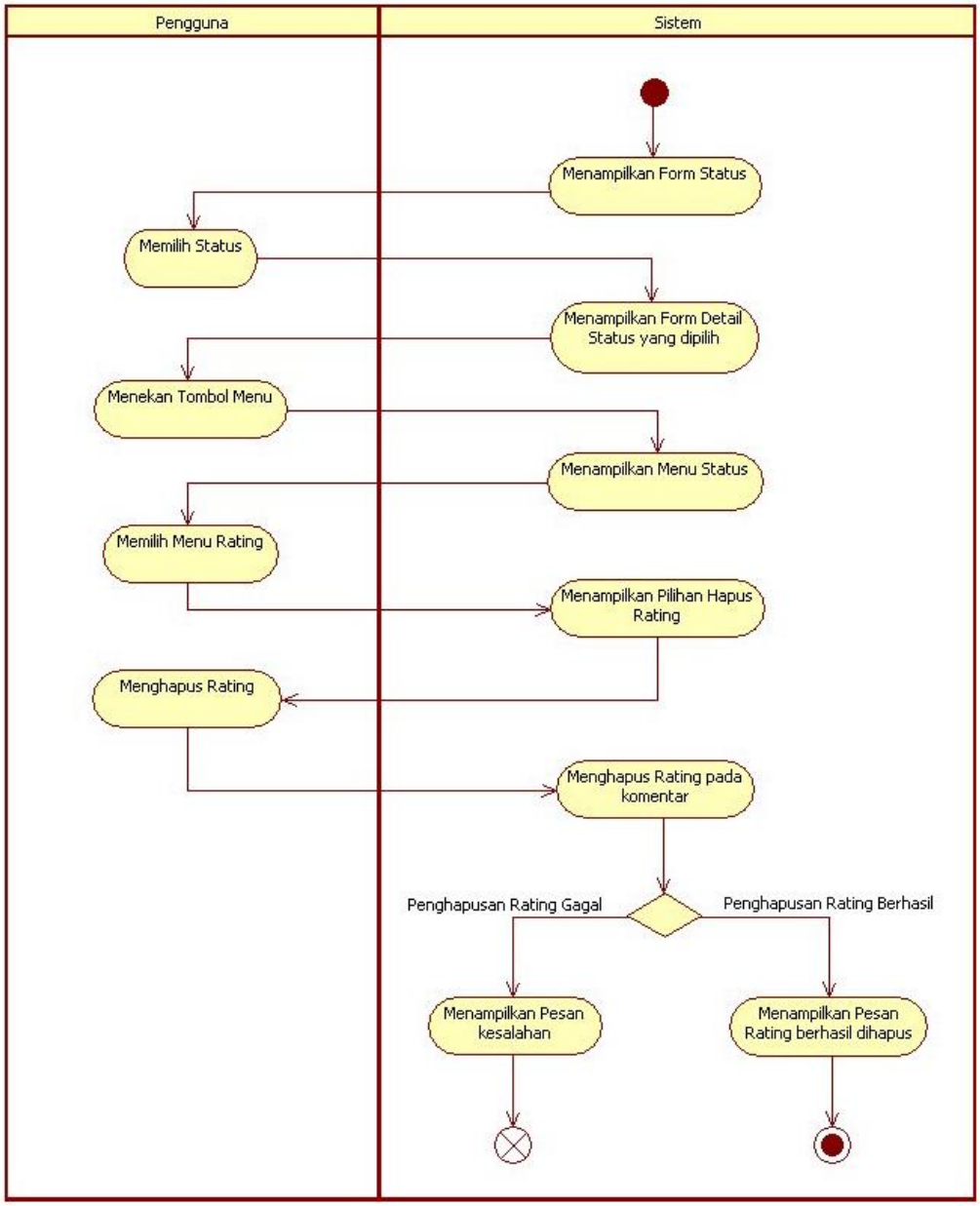
Activity diagram memberi *rating* menjelaskan aliran kerja aktor pengguna dalam memberi *rating* untuk status di aplikasi *mobile* Kampus *Online*. *Activity diagram* memberi *rating* dapat dilihat pada Gambar III.33.



Gambar III.33 *Activity Diagram* Memberi *Rating*

10. *Activity Diagram* Menghapus *Rating*

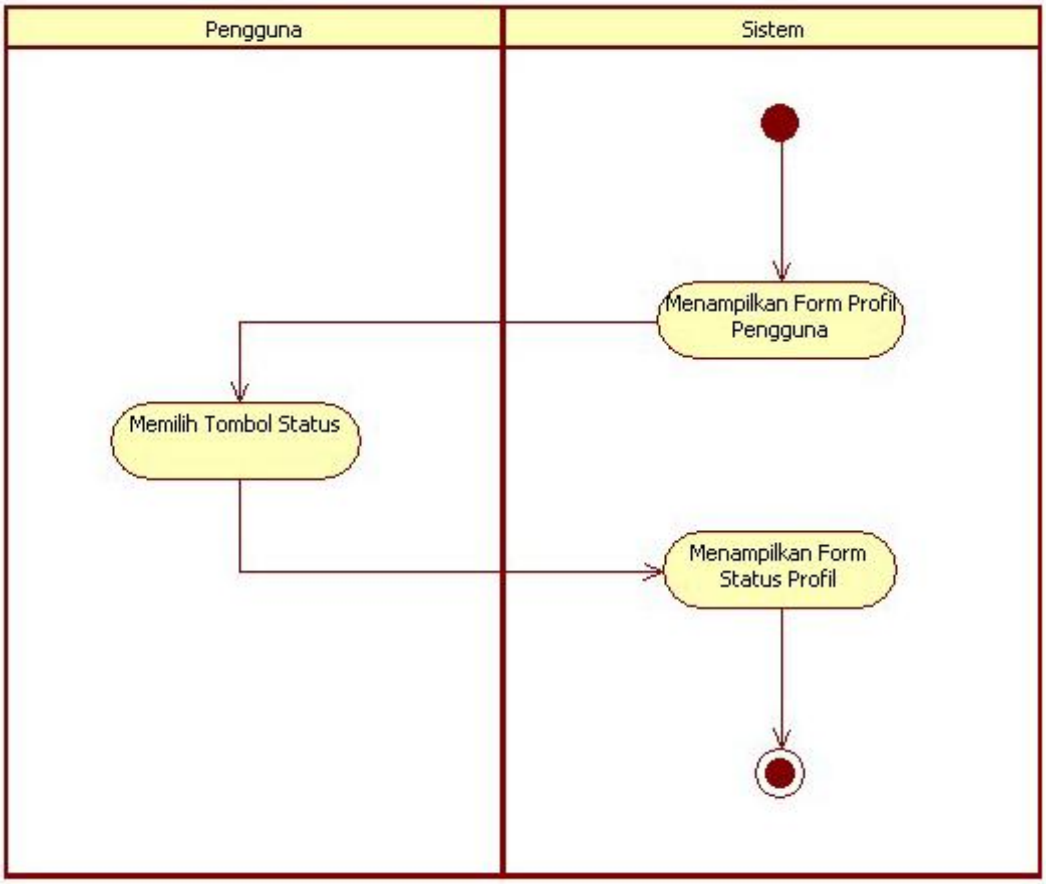
Activity diagram menghapus *rating* menjelaskan aliran kerja aktor pengguna dalam menghapus *rating* untuk status di aplikasi *mobile* Kampus *Online*. *Activity diagram* menghapus *rating* dapat dilihat pada Gambar III.34.



Gambar III.34 *Activity Diagram* Menghapus *Rating*

11. *Activity Diagram* Melihat Data Status Profil

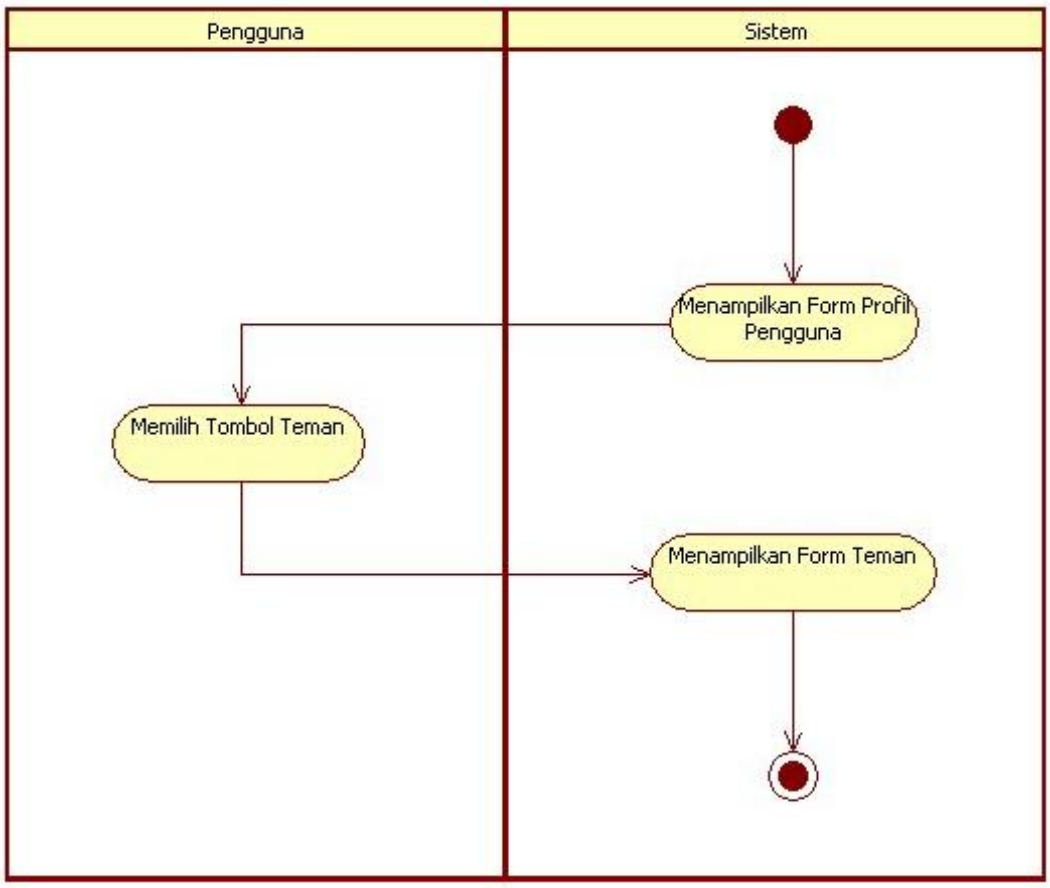
Activity diagram melihat data status profil menjelaskan aliran kerja aktor pengguna untuk melihat data status dari pengguna itu sendiri pada aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat data status profil dapat dilihat pada Gambar III.35.



Gambar III.35 *Activity Diagram* Melihat Data Status Profil

12. *Activity Diagram* Melihat Data Teman

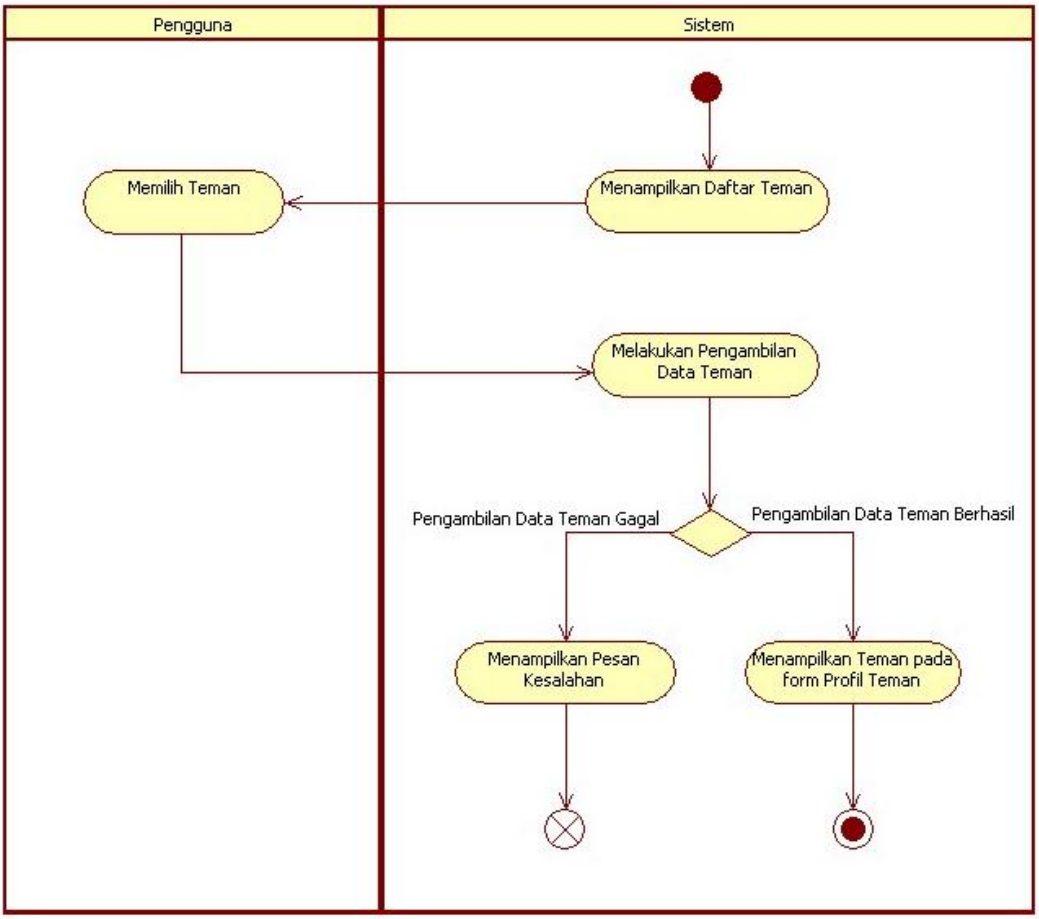
Activity diagram melihat data teman menjelaskan aliran kerja aktor pengguna untuk melihat data teman di aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat data teman dapat dilihat pada Gambar III.36.



Gambar III.36 *Activity Diagram* Melihat Data Teman

13. *Activity Diagram* Melihat Profil Teman

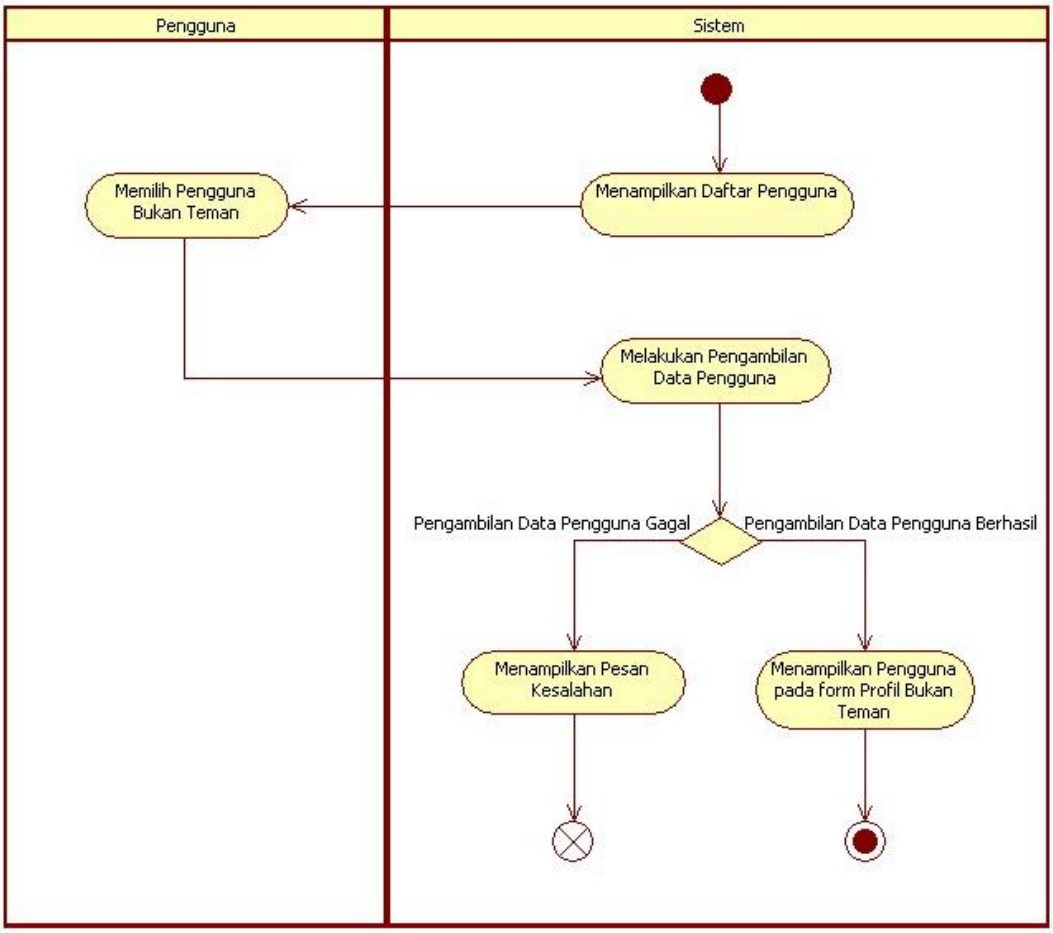
Activity diagram melihat data profil teman menjelaskan aliran kerja aktor pengguna untuk melihat data profil teman di aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat data profil teman dapat dilihat pada Gambar III.37.



Gambar III.37 *Activity Diagram* Melihat Data Profil Teman

14. *Activity Diagram* Melihat Profil Bukan Teman

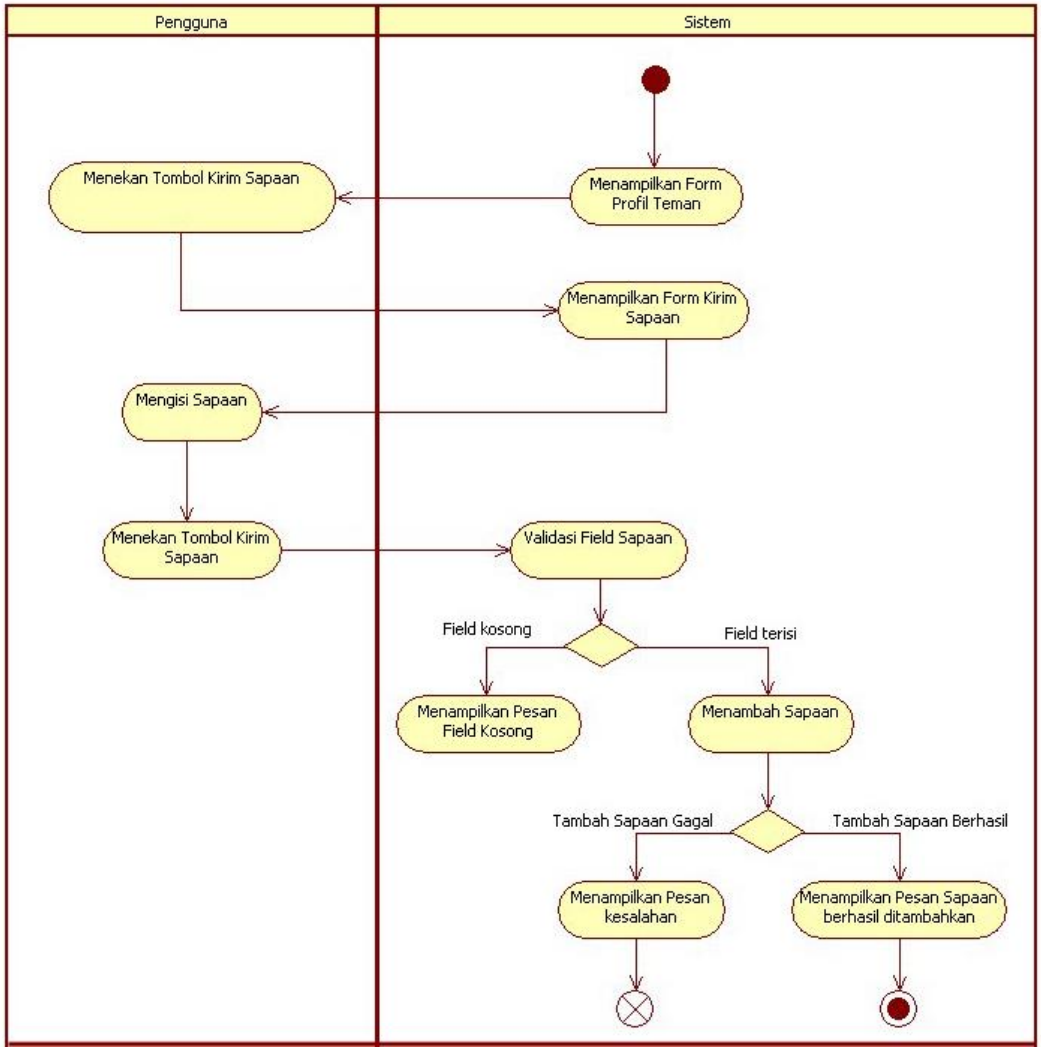
Activity diagram melihat data profil bukan teman menjelaskan aliran kerja aktor pengguna untuk melihat data profil bukan teman di aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat data profil bukan teman dapat dilihat pada Gambar III.38.



Gambar III.38 *Activity Diagram* Melihat Data Profil Bukan Teman

15. *Activity Diagram* Mengirim Sapaan

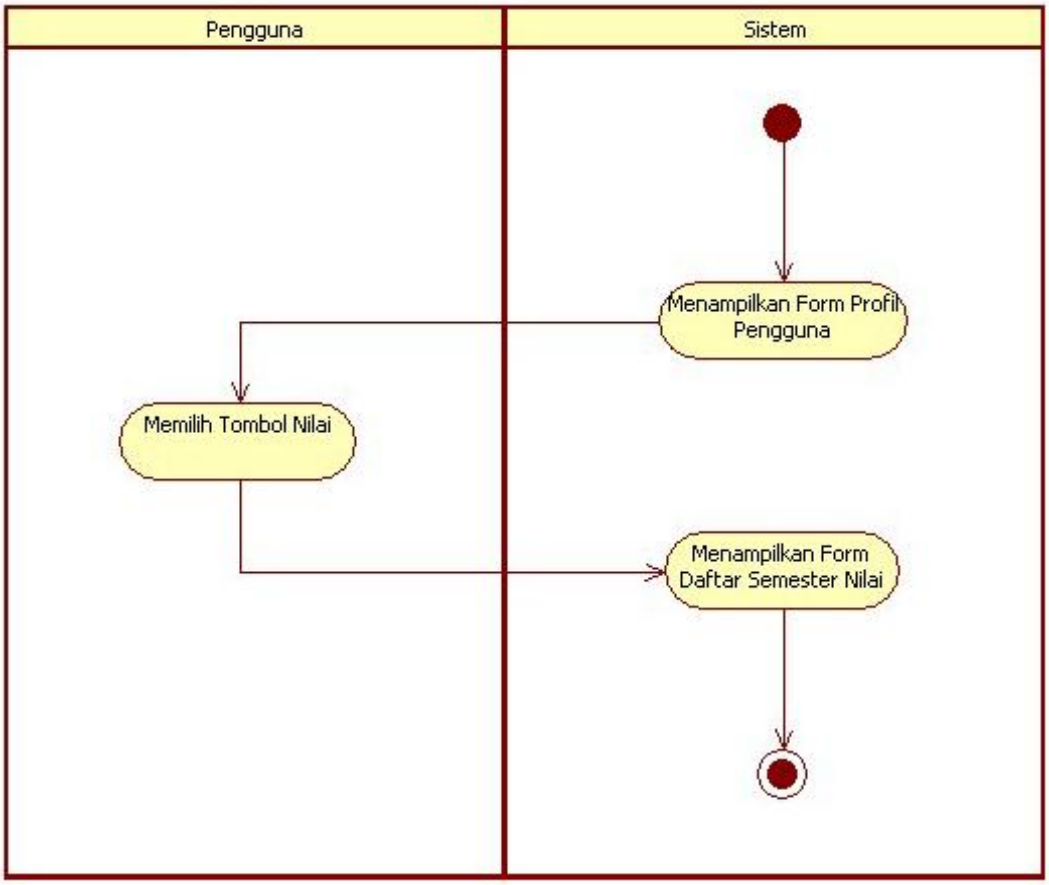
Activity diagram mengirim sapaan menjelaskan aliran kerja aktor pengguna untuk mengirim sapaan kepada teman di aplikasi *mobile* Kampus *Online*. *Activity diagram* mengirim sapaan dapat dilihat pada Gambar III.39.



Gambar III.39 *Activity Diagram* Mengirim Sapaan

16. *Activity Diagram* Melihat Daftar Semester Nilai

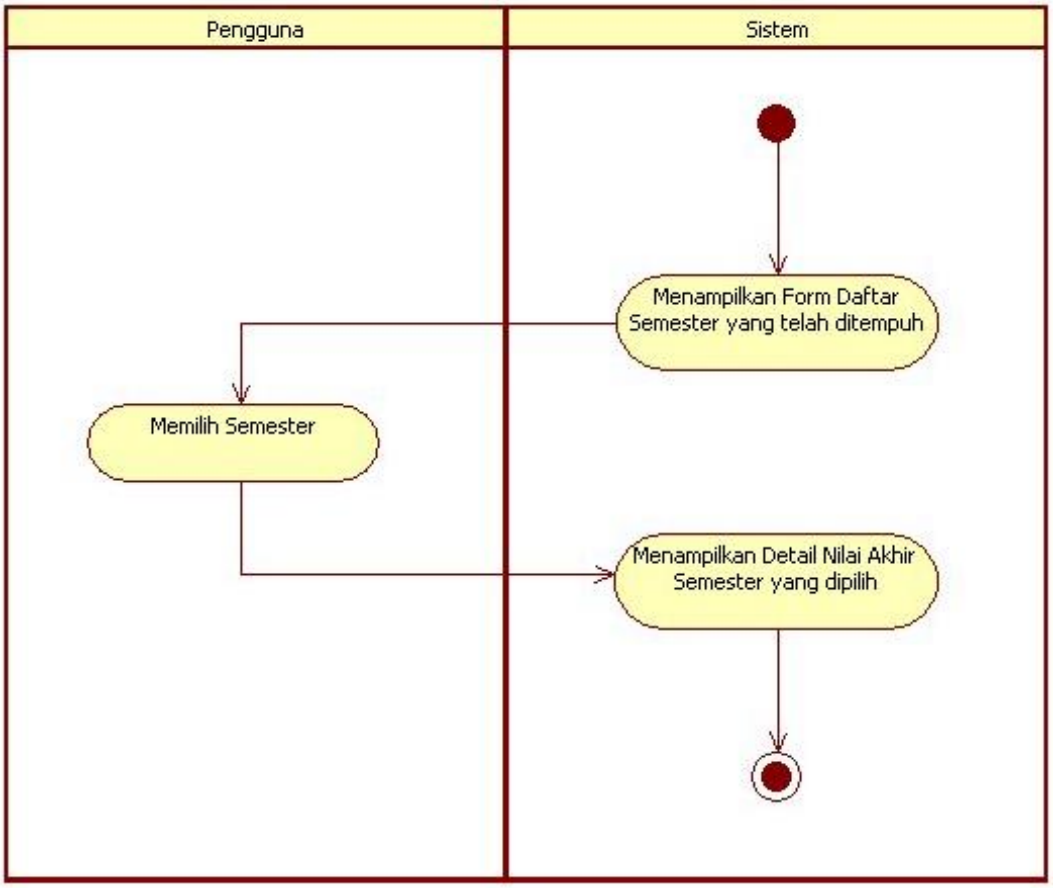
Activity diagram melihat daftar semester nilai menjelaskan aliran kerja aktor pengguna untuk melihat daftar semester nilai yang telah ditempuh di aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat daftar semester nilai dapat dilihat pada Gambar III.40.



Gambar III.40 *Activity Diagram* Melihat Daftar Semester Nilai

17. *Activity Diagram* Melihat Detail Nilai Semester

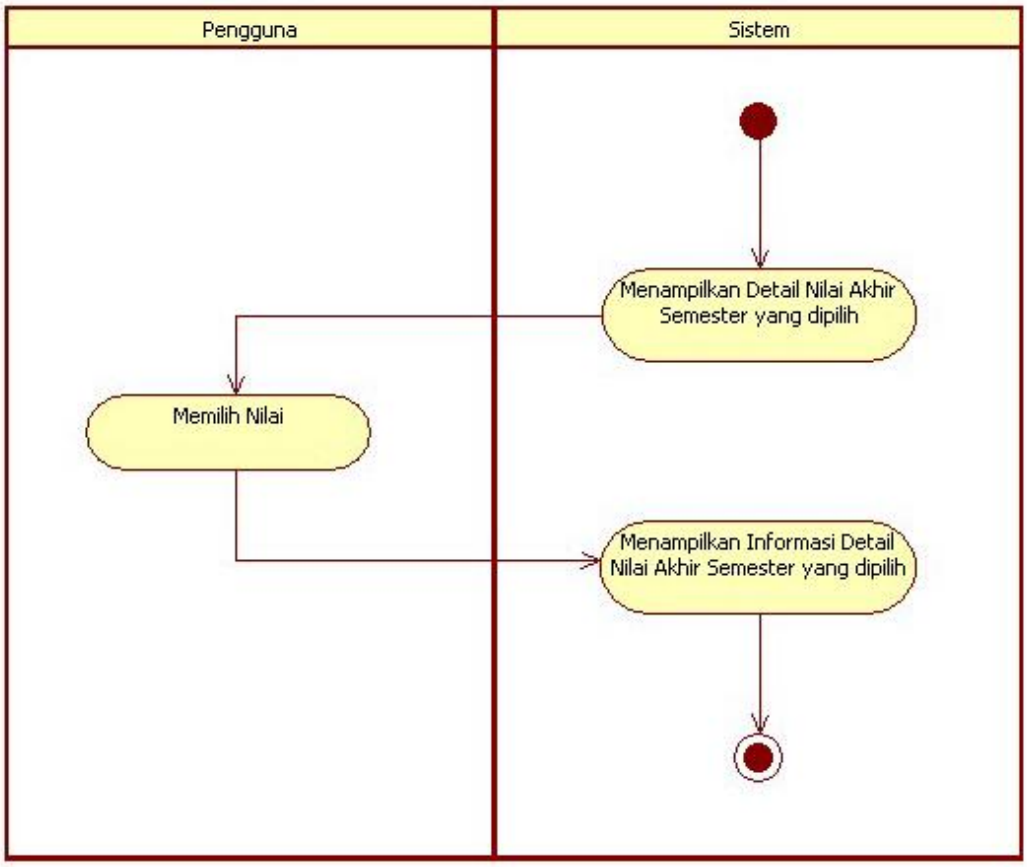
Activity diagram melihat detail nilai semester menjelaskan aliran kerja aktor pengguna untuk melihat detail nilai dari semester yang dipilih di aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat detail nilai semester dapat dilihat pada Gambar III.41.



Gambar III.41 *Activity Diagram* Melihat Detail Nilai Semester

18. *Activity Diagram* Melihat Informasi Detail Nilai

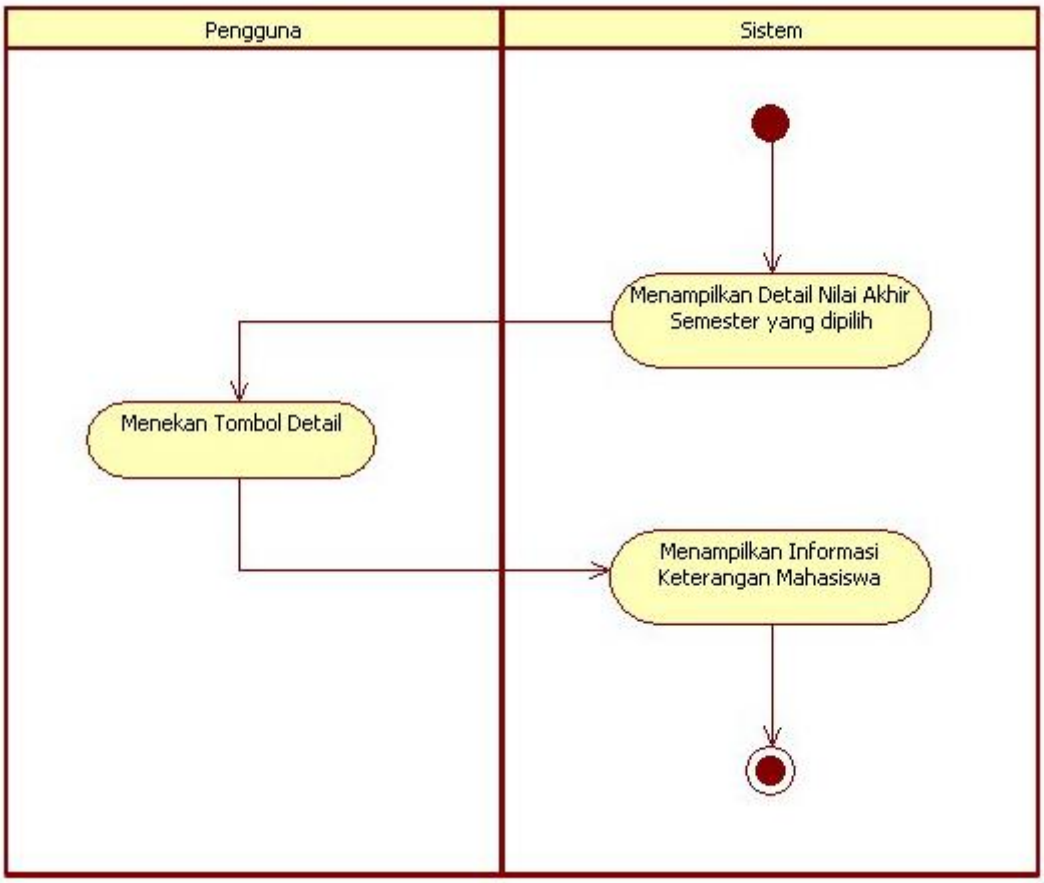
Activity diagram melihat informasi detail nilai menjelaskan aliran kerja aktor pengguna untuk melihat informasi detail nilai dari semester yang dipilih di aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat informasi detail nilai dapat dilihat pada Gambar III.42.



Gambar III.42 *Activity Diagram* Melihat Informasi Detail Nilai

19. *Activity Diagram* Melihat Keterangan Mahasiswa

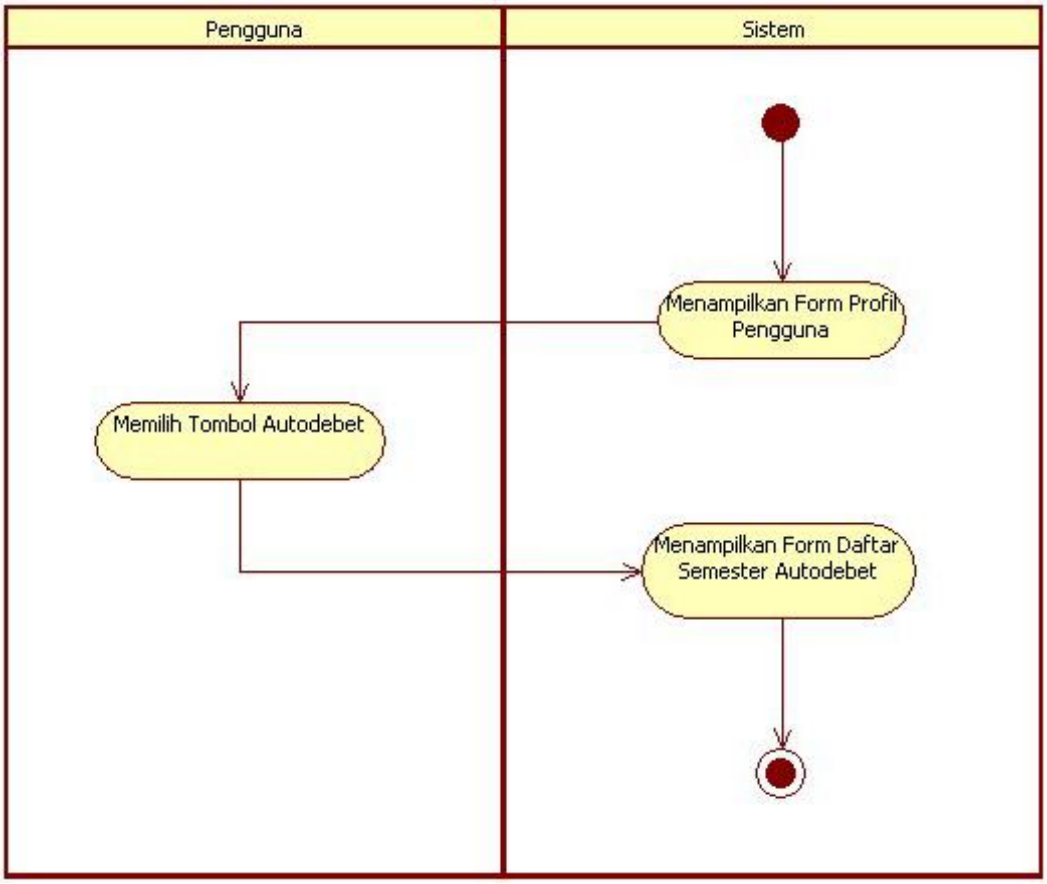
Activity diagram melihat keterangan mahasiswa menjelaskan aliran kerja aktor pengguna untuk melihat keterangan mahasiswa pengguna aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat keterangan mahasiswa dapat dilihat pada Gambar III.43.



Gambar III.43 *Activity Diagram* Melihat Keterangan Mahasiswa

20. *Activity Diagram* Melihat Daftar Semester Autodebet

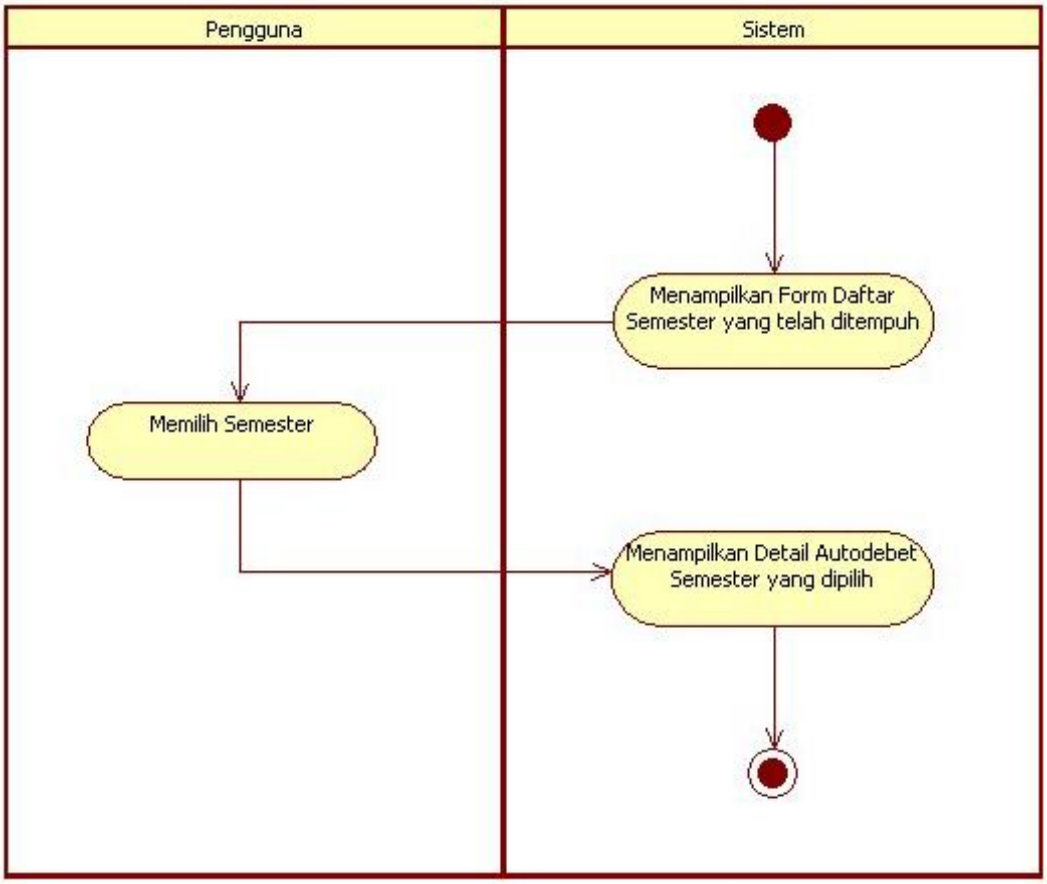
Activity diagram melihat daftar semester autodebet menjelaskan aliran kerja aktor pengguna untuk melihat daftar semester autodebet yang telah ditempuh pada aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat daftar semester autodebet dapat dilihat pada Gambar III.44.



Gambar III.44 *Activity Diagram* Melihat Daftar Semester Autodebet

21. *Activity Diagram* Melihat Detail Autodebet Semester

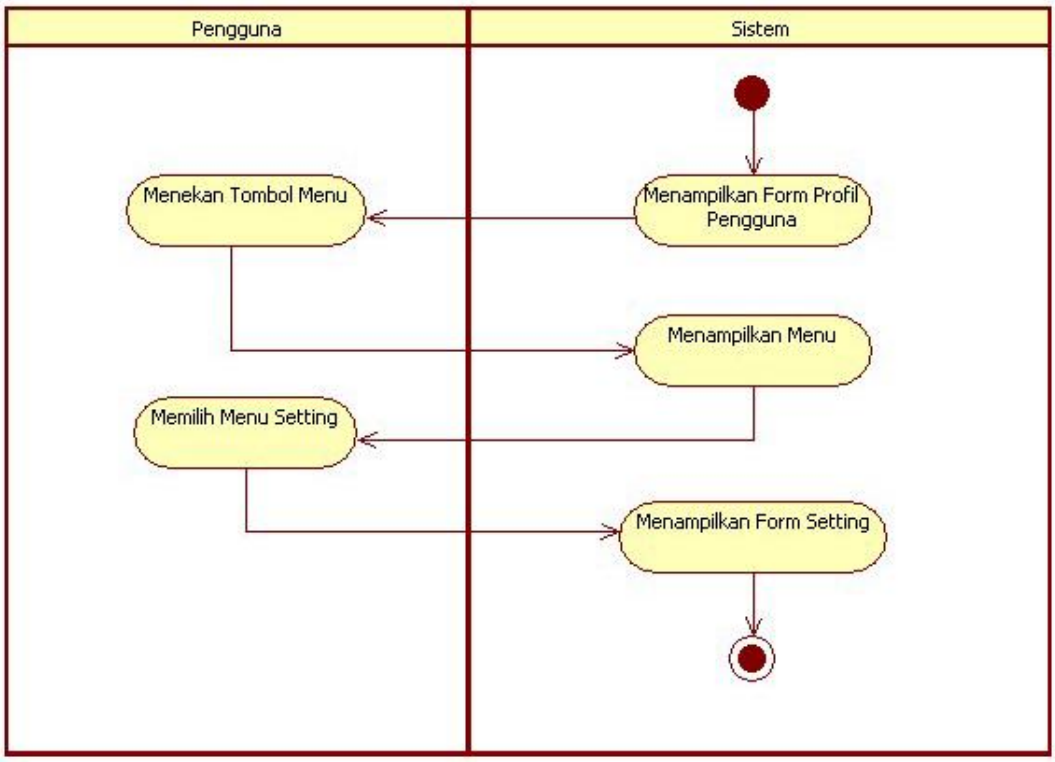
Activity diagram melihat detail autodebet semester menjelaskan aliran kerja aktor pengguna untuk melihat detail data autodebet dari semester yang dipilih di aplikasi *mobile* Kampus *Online*. *Activity diagram* melihat detail autodebet semester dapat dilihat pada Gambar III.45.



Gambar III.45 *Activity Diagram* Melihat Detail Autodebet Semester

22. *Activity Diagram Setting*

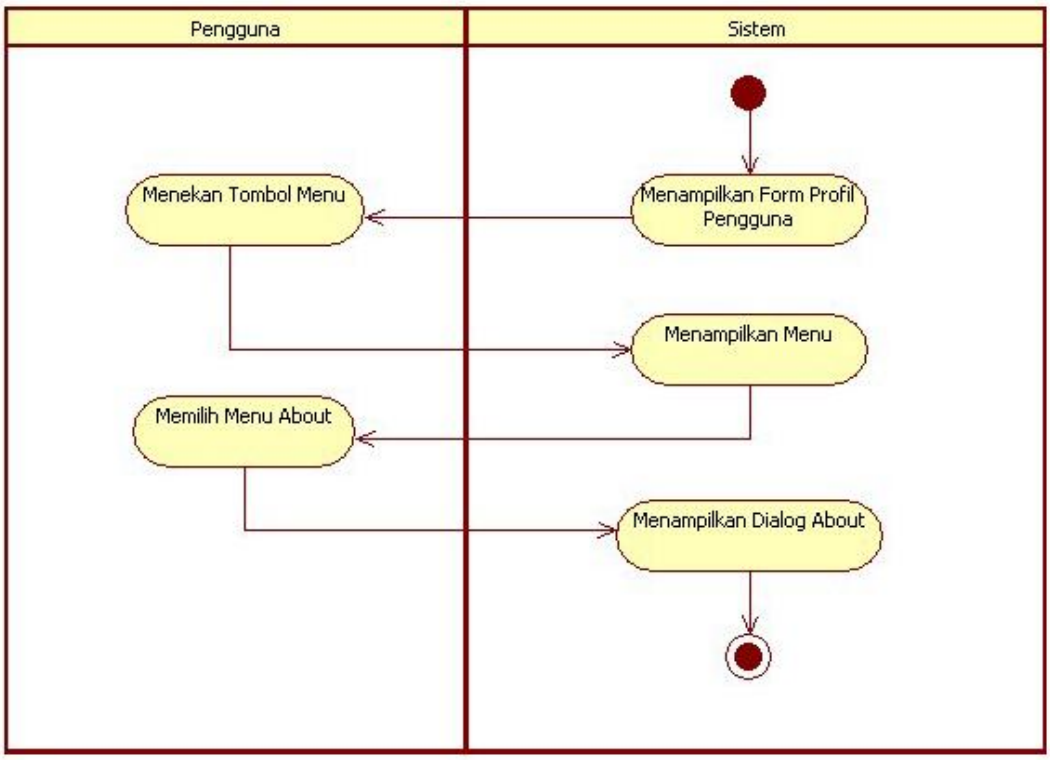
Activity diagram setting menjelaskan aliran kerja aktor pengguna untuk melakukan pengaturan aplikasi *mobile* Kampus *Online*. *Activity diagram setting* dapat dilihat pada Gambar III.46.



Gambar III.46 *Activity Diagram Setting*

23. *Activity Diagram About*

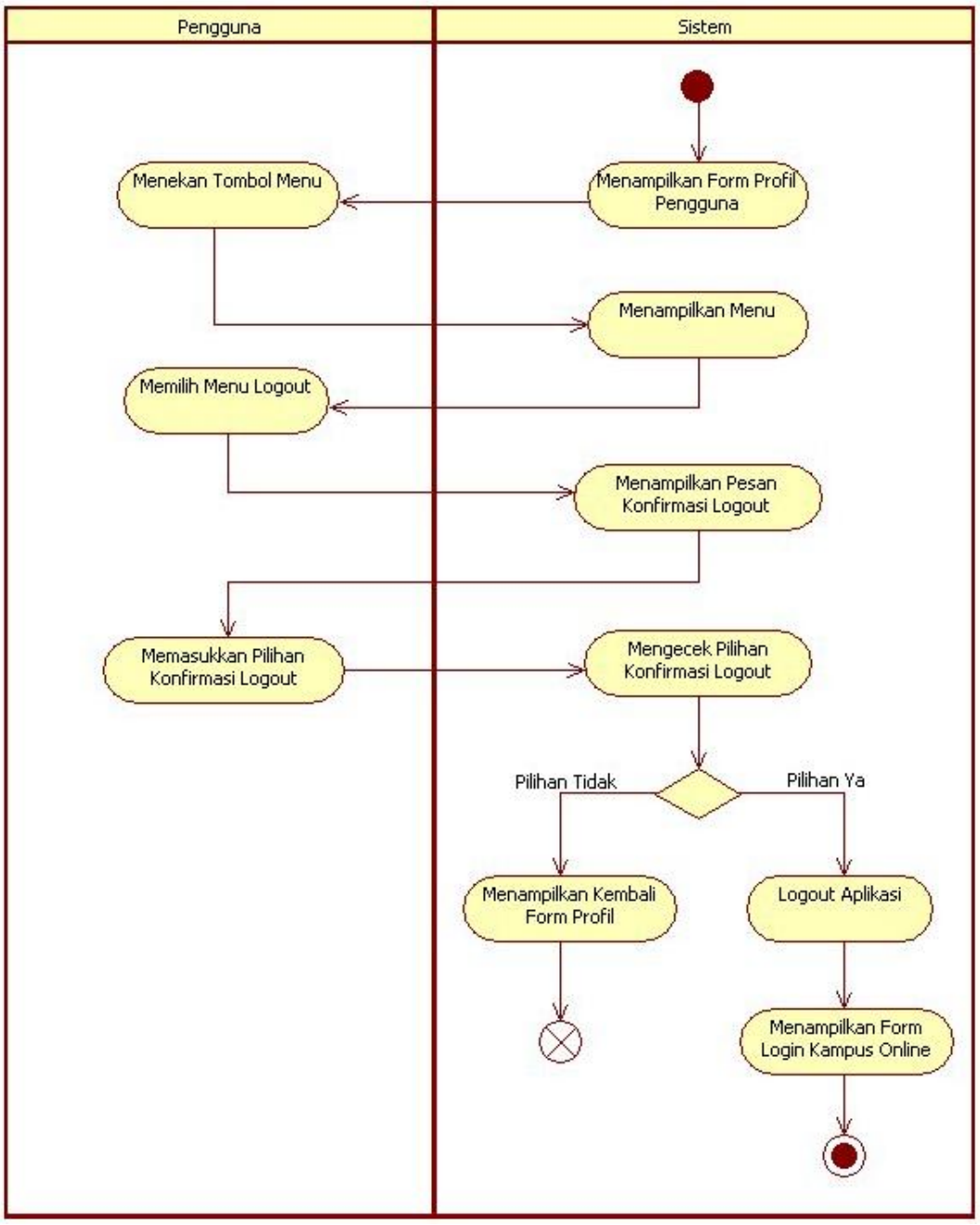
Activity diagram about menjelaskan aliran kerja aktor pengguna untuk melihat halaman *about* aplikasi *mobile* Kampus *Online*. *Activity diagram about* dapat dilihat pada Gambar III.47.



Gambar III.47 *Activity Diagram About*

24. *Activity Diagram Logout*

Activity diagram logout menjelaskan aliran kerja aktor pengguna pada saat *logout* dari aplikasi *mobile* Kampus *Online*. *Activity diagram logout* dapat dilihat pada Gambar III.48.



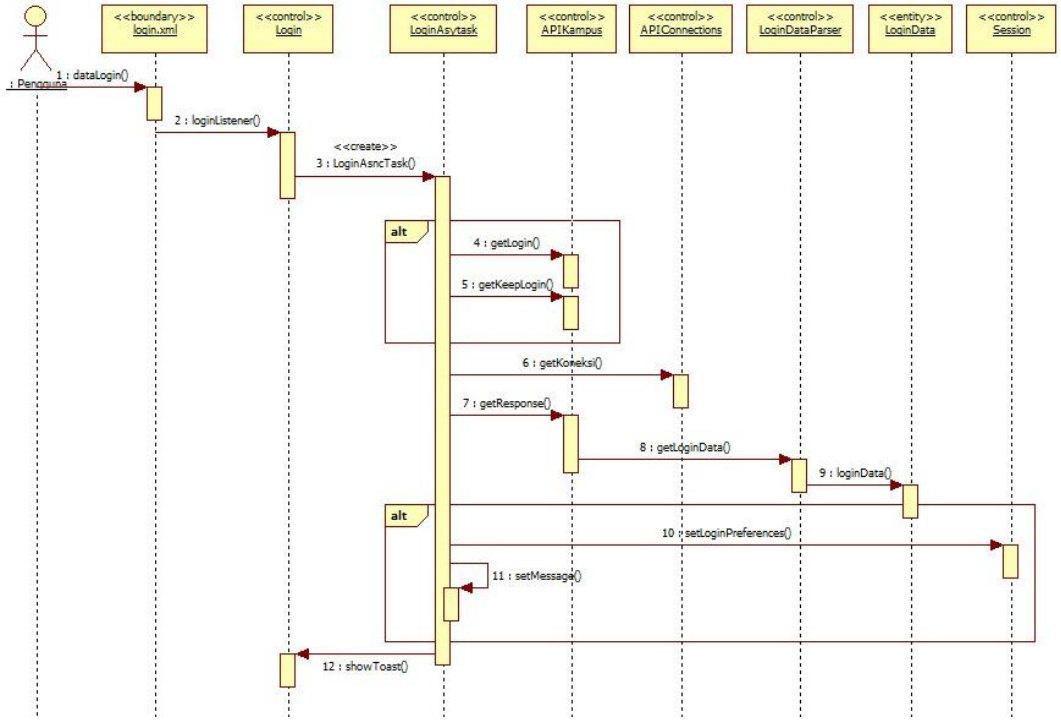
Gambar III.48 *Activity Diagram Logout*

3. 1. 9. 4. ***Squence Diagram***

Squence diagram merupakan gambaran interaksi antar masing-masing objek pada setiap *use case* dalam urutan waktu. Interaksi ini berupa pengiriman serangkaian data antar objek-objek yang saling berinteraksi. *Squence diagram* pada aplikasi Kampus *Online* Unikom terdiri dari *squence diagram login*, update status, mencari data pengguna lain, melihat data profil, melihat detail status, memberi komentar, memberi *rating*, melihat data status profil, melihat data teman, melihat daftar semester nilai, melihat detail nilai semester, melihat daftar semester autodebet, melihat detail autodebet semester, melihat informasi detail nilai, melihat keterangan mahasiswa, *setting*, *about*, dan *logout*. Berikut ini pemaparan dari masing-masing *squence diagram* yang terlihat pada Gambar III.49 sampai dengan Gambar III.72.

1. *Squence Diagram Login*

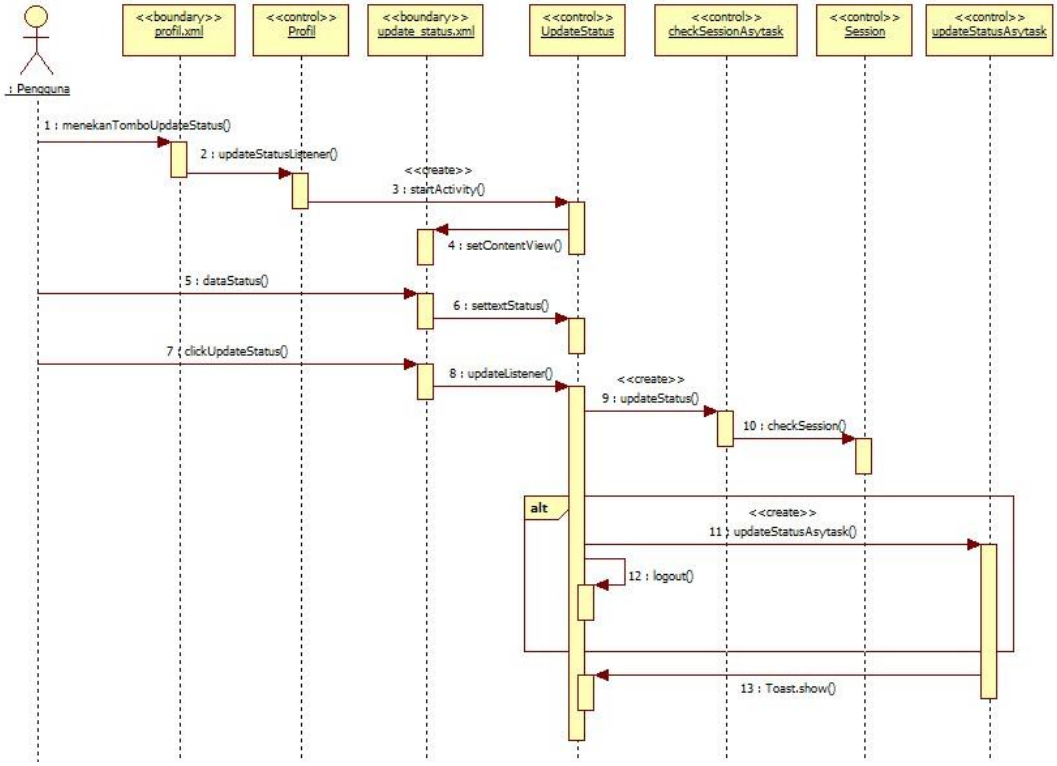
Squence diagram login menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses *login* pada aplikasi *mobile* Kampus *Online*. *Squence diagram login* dapat dilihat pada Gambar III.49.



Gambar III.49 *Squence Diagram Login*

2. *Squence Diagram Update Status*

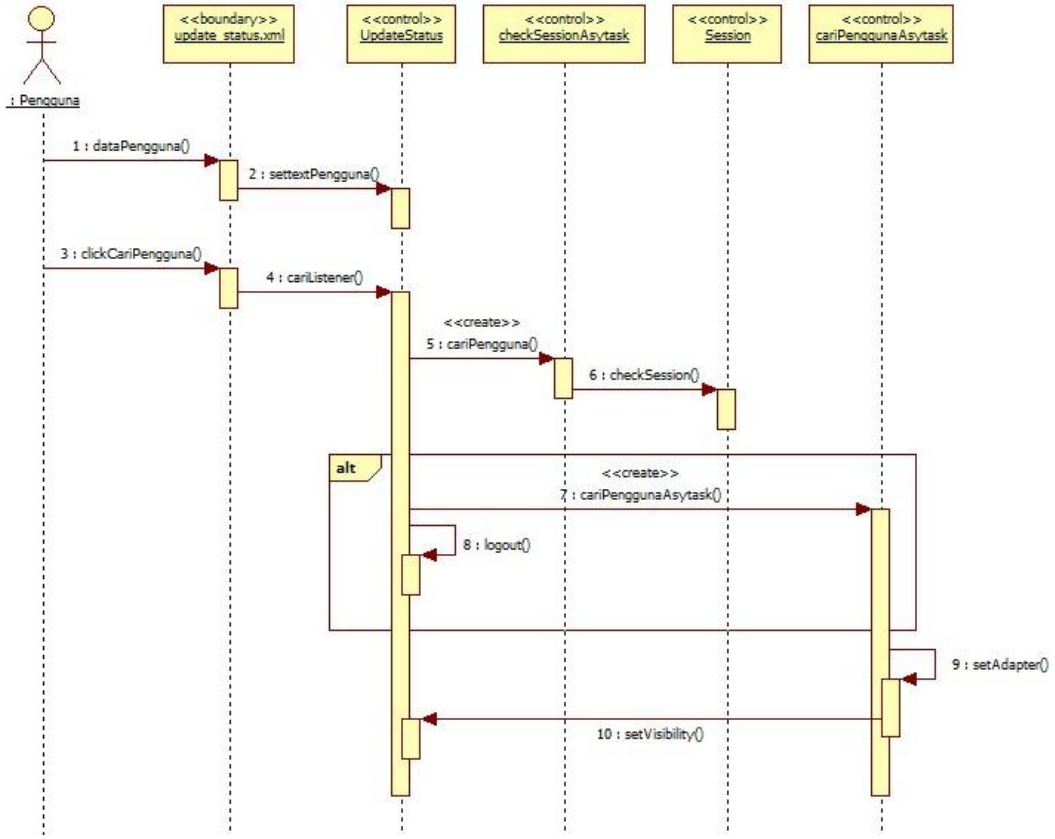
Squence diagram update status menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses *update status* pada aplikasi *mobile* Kampus *Online*. *Squence diagram update status* dapat dilihat pada Gambar III.50.



Gambar III.50 *Squence Diagram Update Status*

3. *Squence Diagram* Mencari Data Pengguna

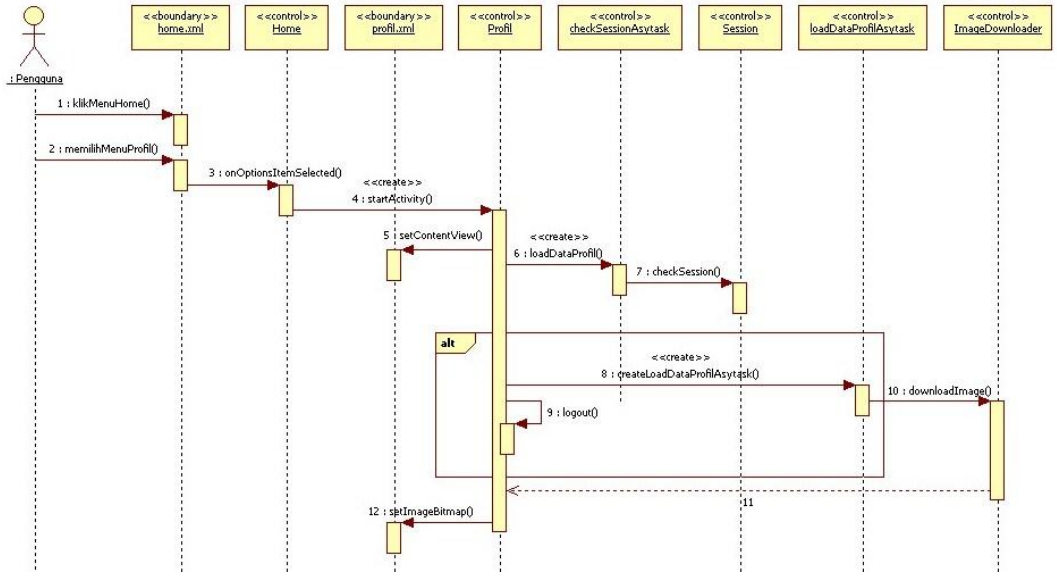
Squence diagram mencari data pengguna menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam mencari data pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* mencari data pengguna dapat dilihat pada Gambar III.51.



Gambar III.51 *Squence Diagram* Mencari Data Pengguna

4. *Squence Diagram* Melihat Data Profil

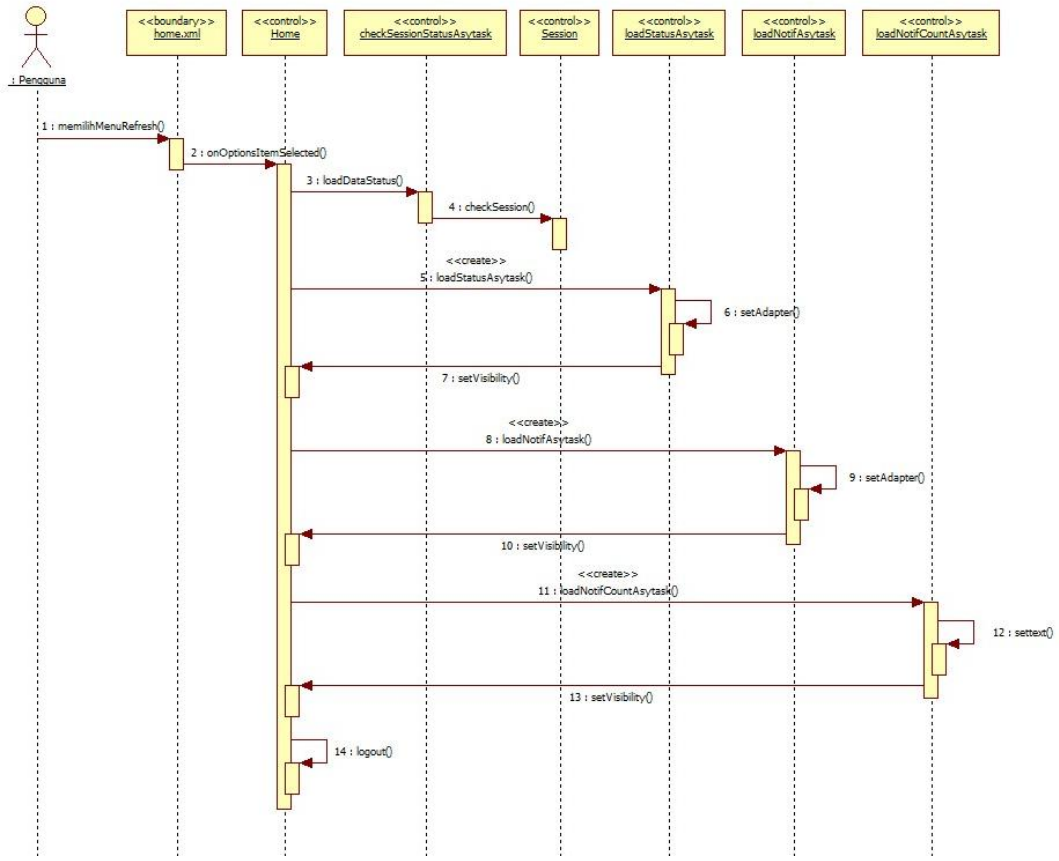
Squence diagram melihat data profil menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses melihat data profil pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat data profil dapat dilihat pada Gambar III.52.



Gambar III.52 *Squence Diagram* Melihat Data Profil

5. *Squence Diagram* Melihat Data Status

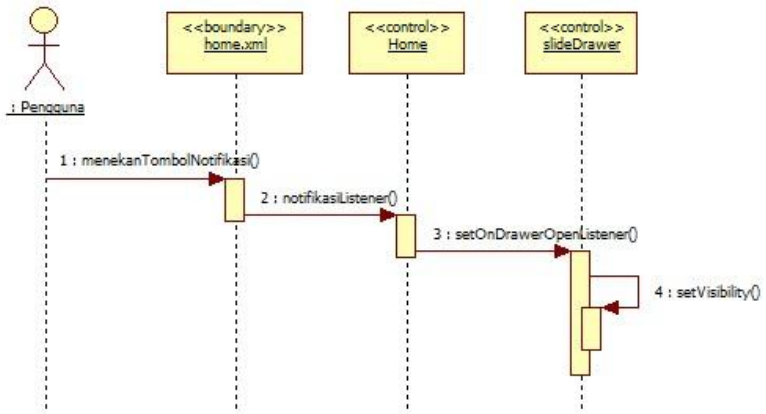
Squence diagram melihat data status menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses melihat data status pada aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat data status dapat dilihat pada Gambar III.53.



Gambar III.53 *Squence Diagram* Melihat Data Status

6. *Squence Diagram* Melihat Data Notifikasi

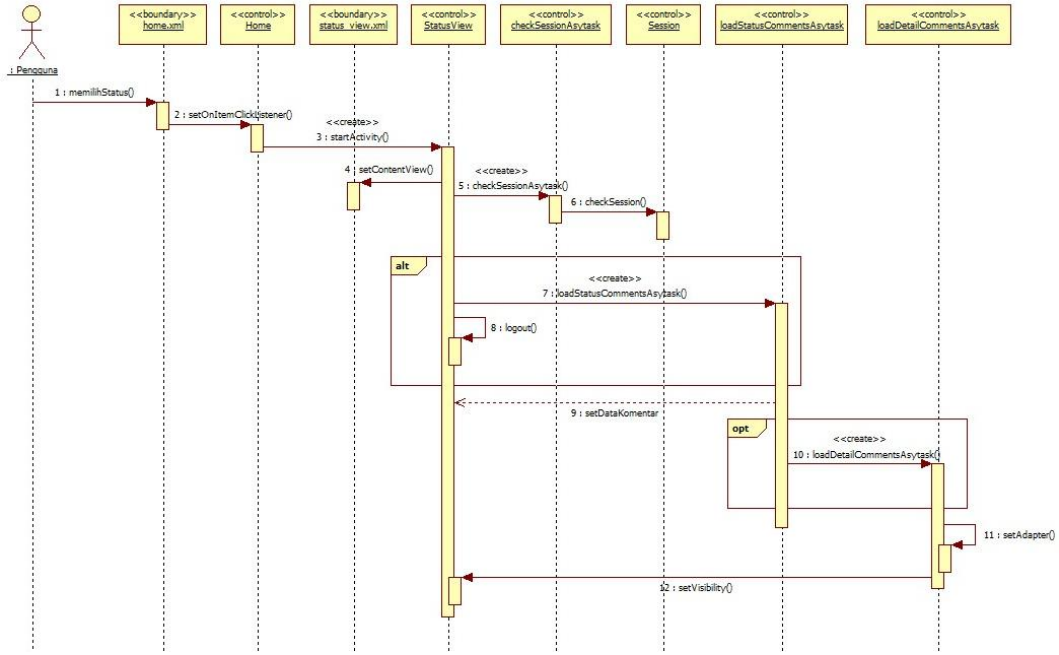
Squence diagram melihat data notifikasi menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses melihat data notifikasi pada aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat data notifikasi dapat dilihat pada Gambar III.54.



Gambar III.54 *Squence Diagram* Melihat Data Notifikasi

7. *Squence Diagram* Melihat Detail Status

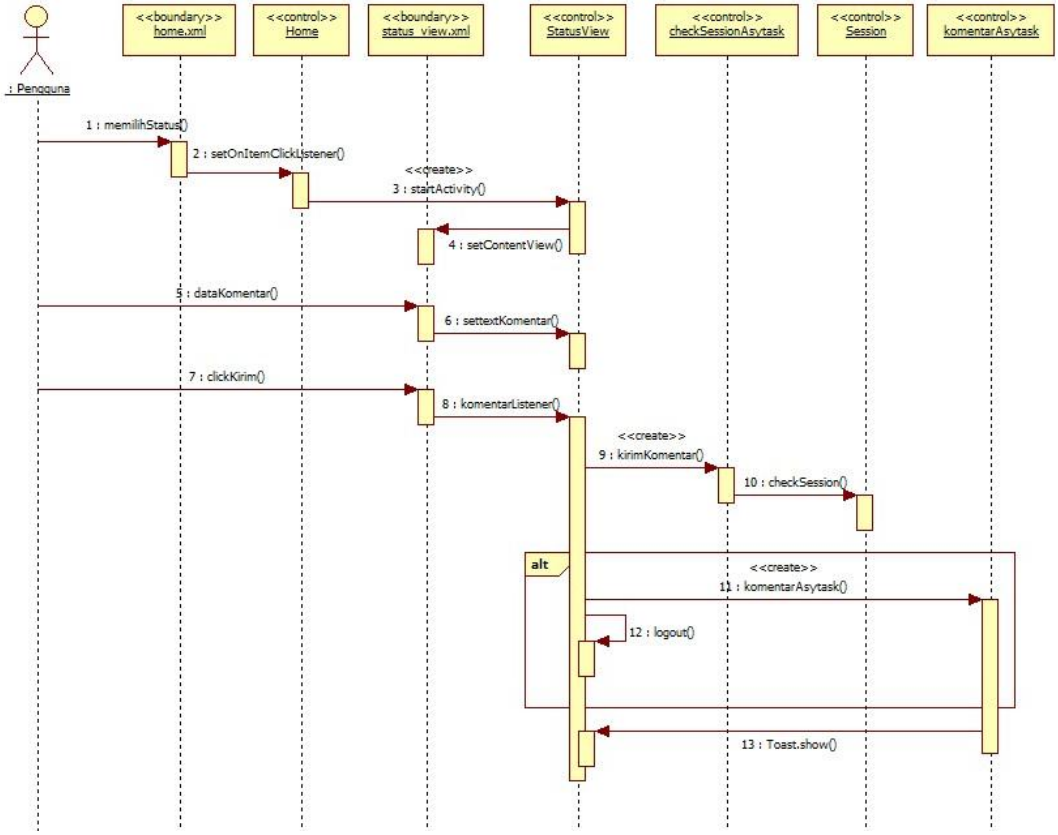
Squence diagram melihat detail status menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses melihat detail status pada aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat detail status dapat dilihat pada Gambar III.55.



Gambar III.55 *Squence Diagram* Melihat Detail Status

8. *Squence Diagram* Memberi Komentar

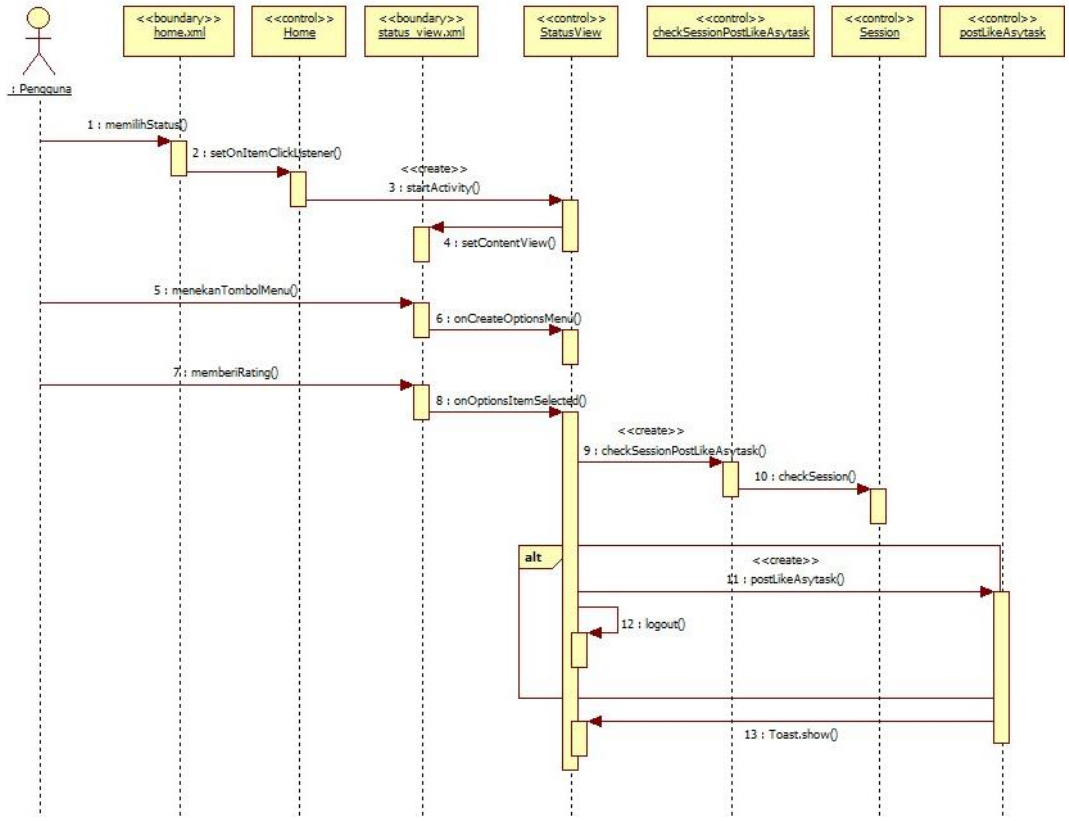
Squence diagram memberi komentar menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses memberi komentar pada status pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* memberi komentar dapat dilihat pada Gambar III.56.



Gambar III.56 *Squence Diagram* Memberi Komentar

9. *Squence Diagram* Memberi *Rating*

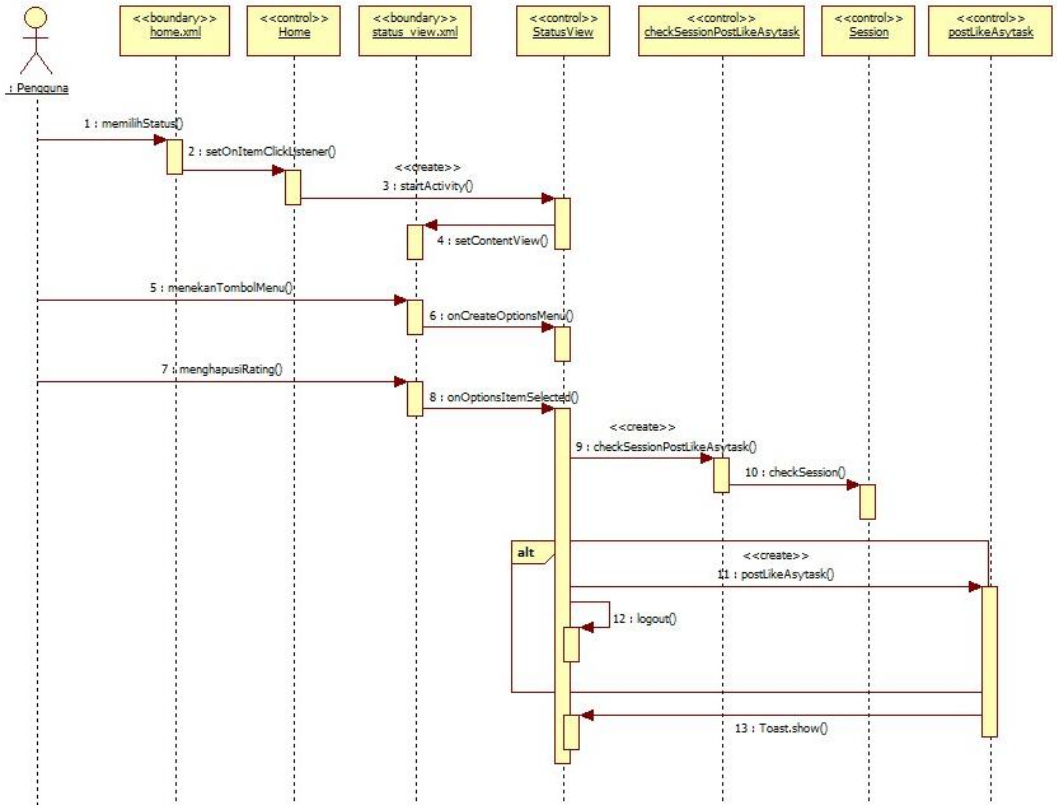
Squence diagram memberi *rating* menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses memberi *rating* pada status pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* memberi *rating* dapat dilihat pada Gambar III.57.



Gambar III.57 *Squence Diagram* Memberi *Rating*

10. *Squence Diagram* Menghapus *Rating*

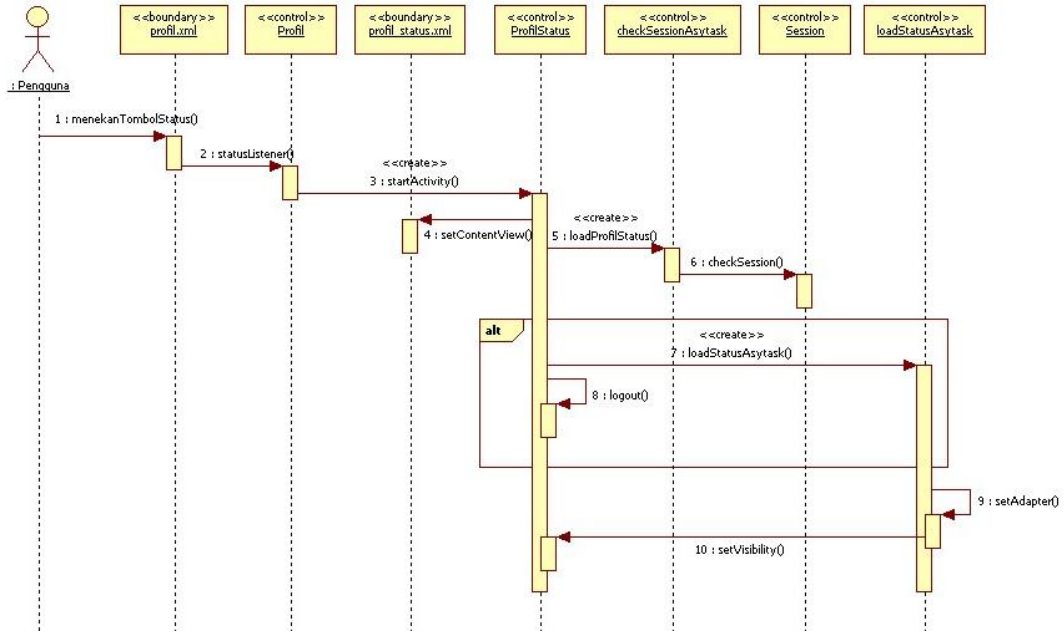
Squence diagram menghapus *rating* menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses menghapus *rating* pada status pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* menghapus *rating* dapat dilihat pada Gambar III.58.



Gambar III.58 *Squence Diagram* Menghapus *Rating*

11. *Squence Diagram* Melihat Data Status Profil

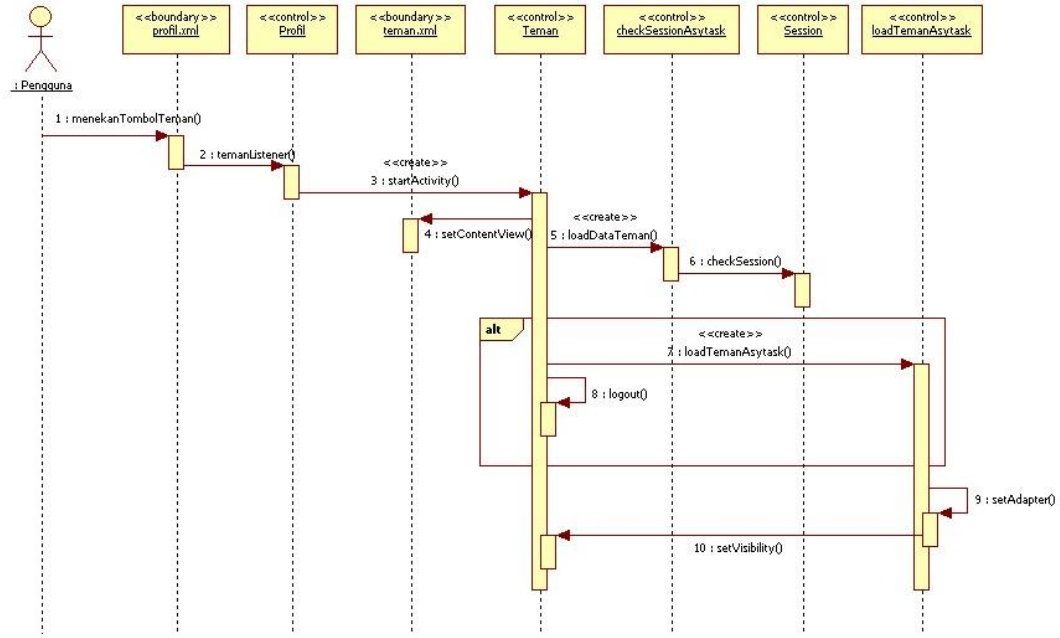
Squence diagram melihat data status profil menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat data status profil pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat data status profil dapat dilihat pada Gambar III.59.



Gambar III.59 *Squence Diagram* Melihat Data Status Profil

12. *Squence Diagram* Melihat Data Teman

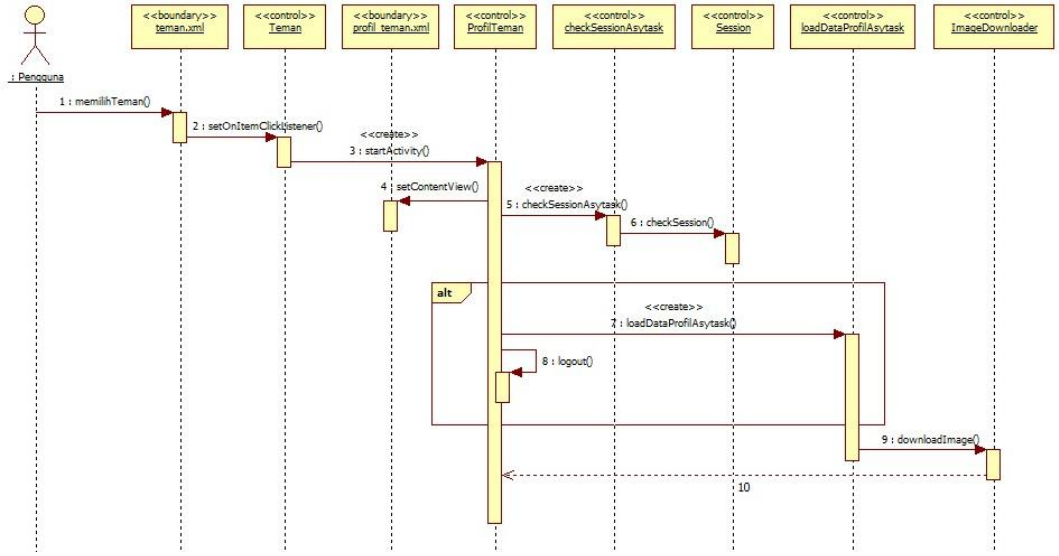
Squence diagram melihat data teman menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat data teman pada aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat data teman dapat dilihat pada Gambar III.60.



Gambar III.60 *Squence Diagram* Melihat Data Teman

13. *Squence Diagram* Melihat Profil Teman

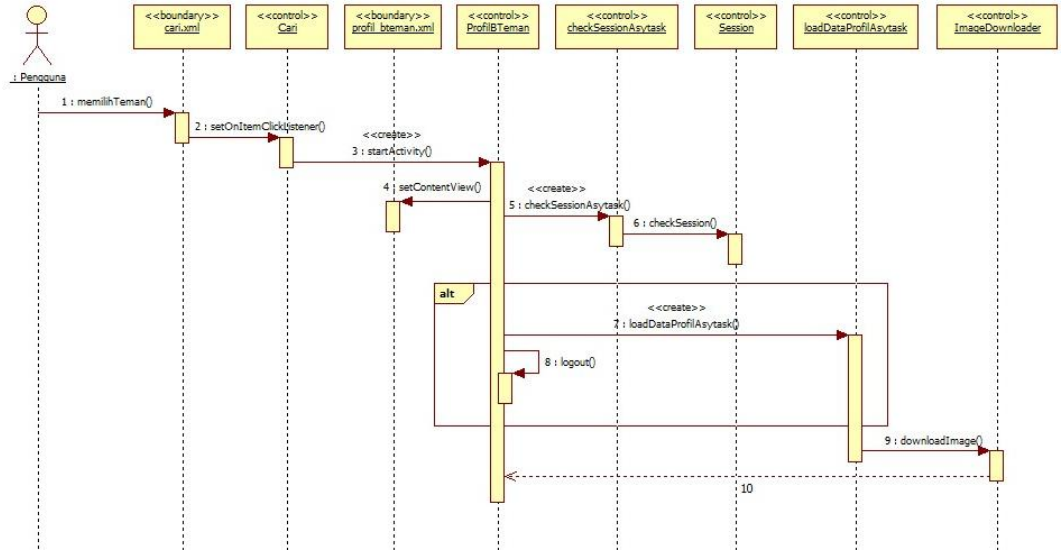
Squence diagram melihat profil teman menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat profil teman pada aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat profil teman dapat dilihat pada Gambar III.61.



Gambar III.61 *Squence Diagram* Melihat Profil Teman

14. *Squence Diagram* Melihat Profil Bukan Teman

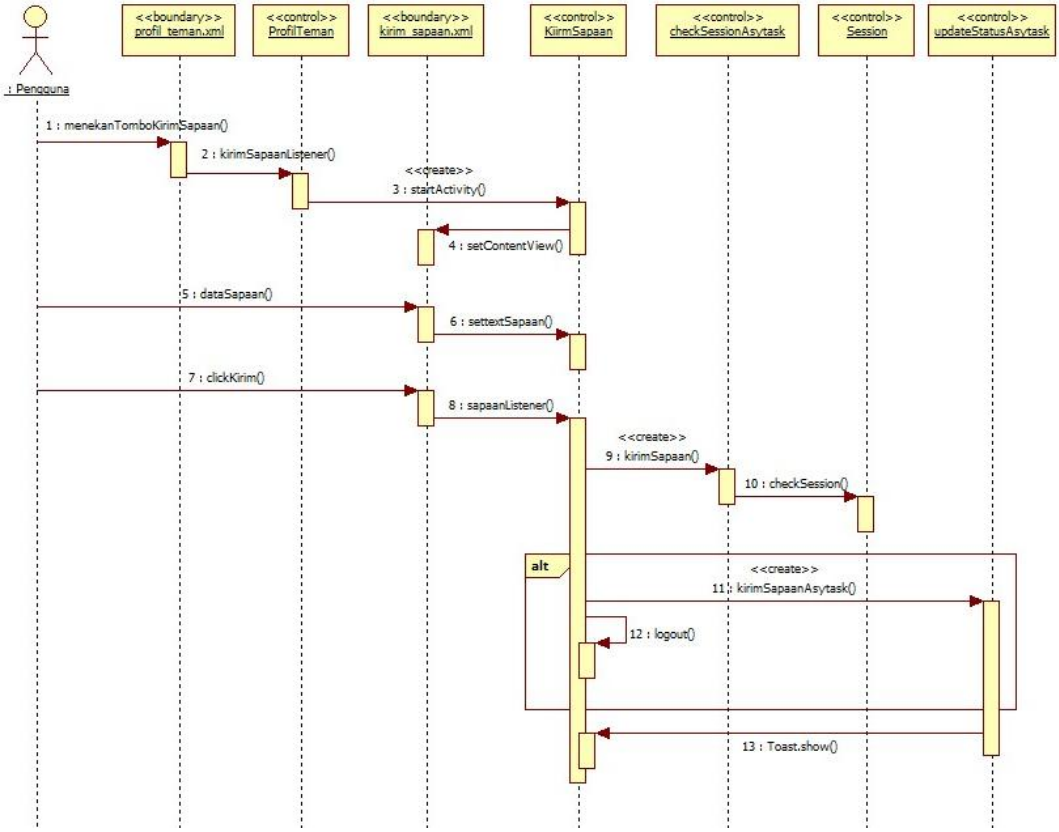
Squence diagram melihat profil bukan teman menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat profil bukan teman pada aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat profil bukan teman dapat dilihat pada Gambar III.62.



Gambar III.62 *Squence Diagram* Melihat Profil Bukan Teman

15. *Squence Diagram* Mengirim Sapaan

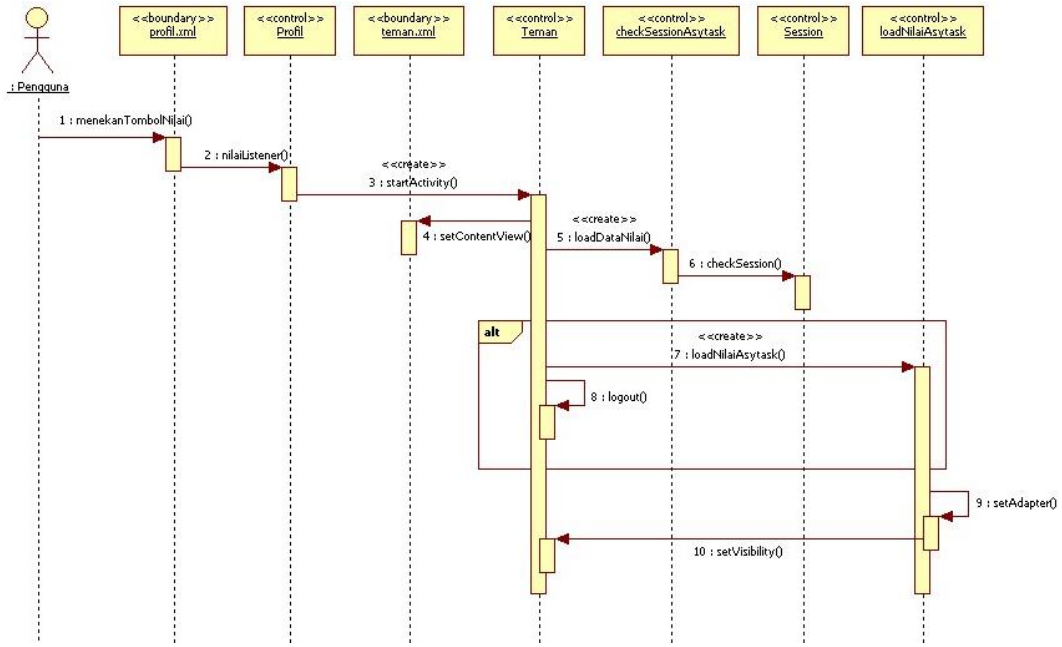
Squence diagram melihat data teman menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat data teman pada aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat data teman dapat dilihat pada Gambar III.63.



Gambar III.63 *Squence Diagram* Mengirim Sapaan

16. *Squence Diagram* Melihat Daftar Semester Nilai

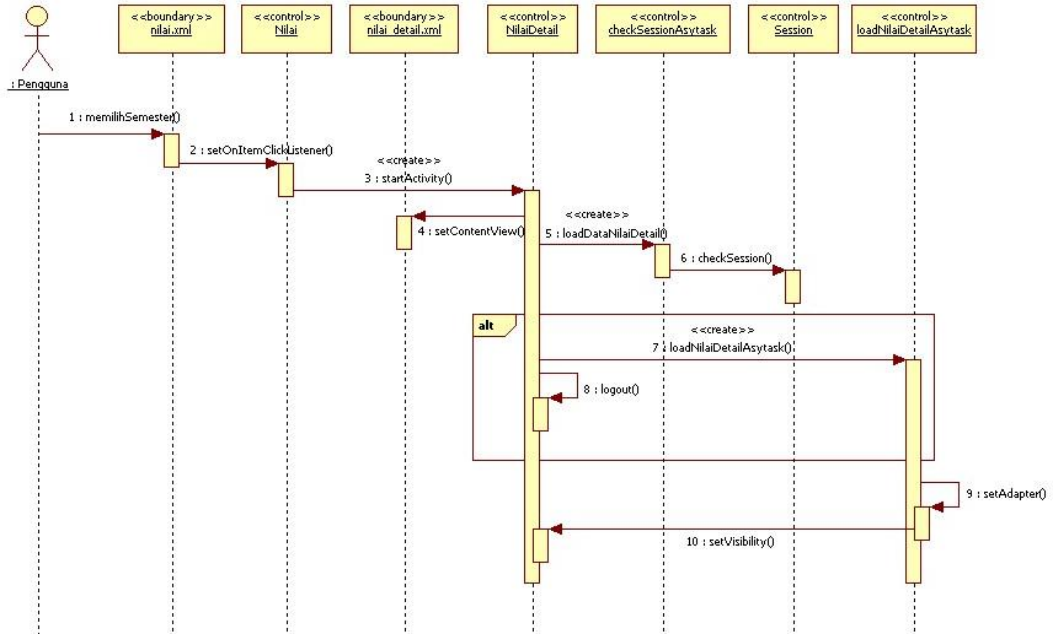
Squence diagram melihat daftar semester nilai menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat data daftar semester nilai yang telah ditempuh pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat daftar semester nilai dapat dilihat pada Gambar III.64.



Gambar III.64 *Squence Diagram* Melihat Daftar Semester Nilai

17. *Squence Diagram* Melihat Detail Nilai Semester

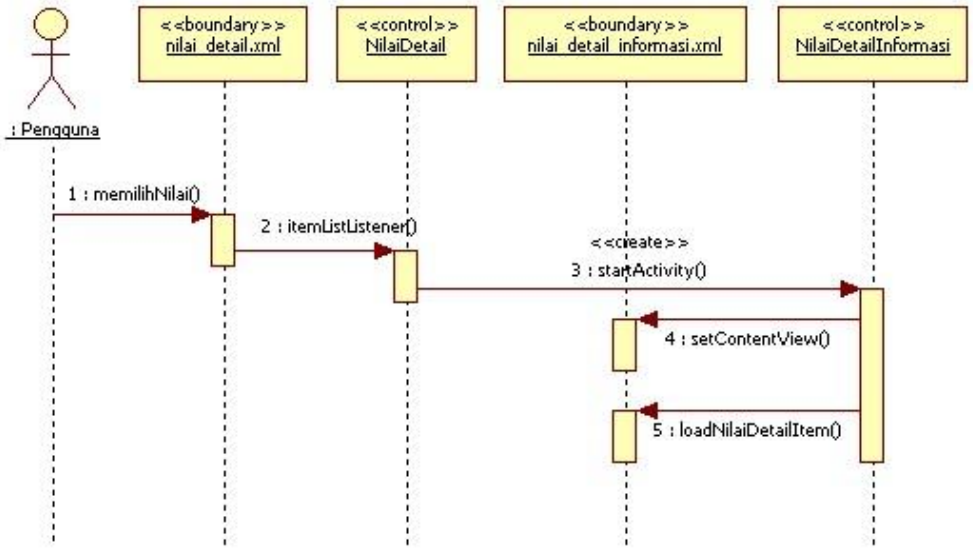
Squence diagram melihat detail nilai semester menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat detail nilai semester pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat detail nilai semester dapat dilihat pada Gambar III.65.



Gambar III.65 *Squence Diagram* Melihat Detail Nilai Semester

18. *Squence Diagram* Melihat Informasi Detail Nilai

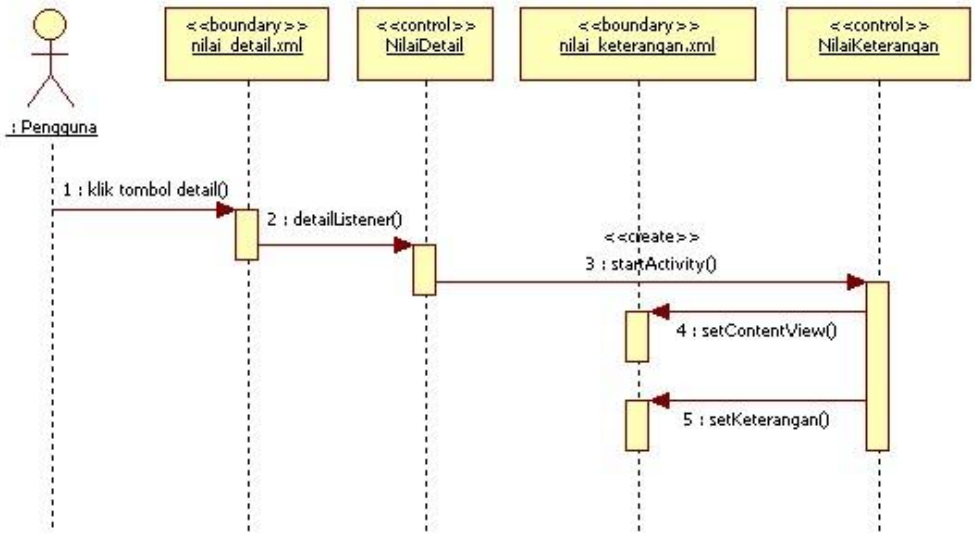
Squence diagram melihat informasi detail nilai menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat informasi detail nilai pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat informasi detail nilai dapat dilihat pada Gambar III.66.



Gambar III.66 *Squence Diagram* Melihat Informasi Detail Nilai

19. *Squence Diagram* Melihat Keterangan Mahasiswa

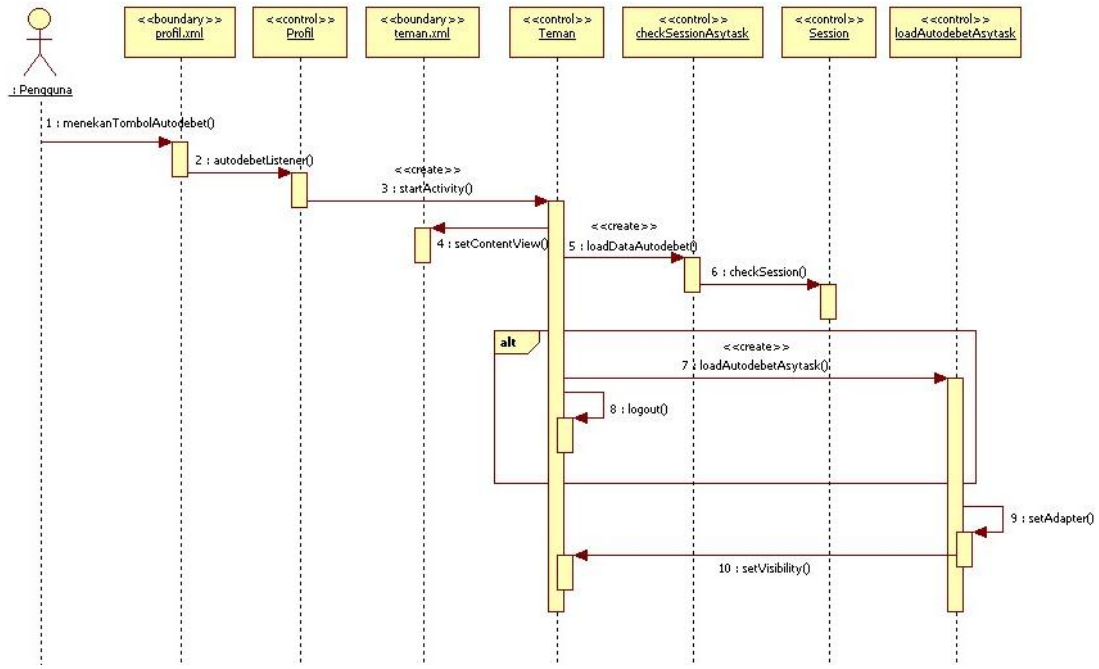
Squence diagram melihat keterangan mahasiswa menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat keterangan mahasiswa pada aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat keterangan mahasiswa dapat dilihat pada Gambar III.67.



Gambar III.67 *Squence Diagram* Melihat Informasi Detail Nilai

20. *Squence Diagram* Melihat Daftar Semester Autodebet

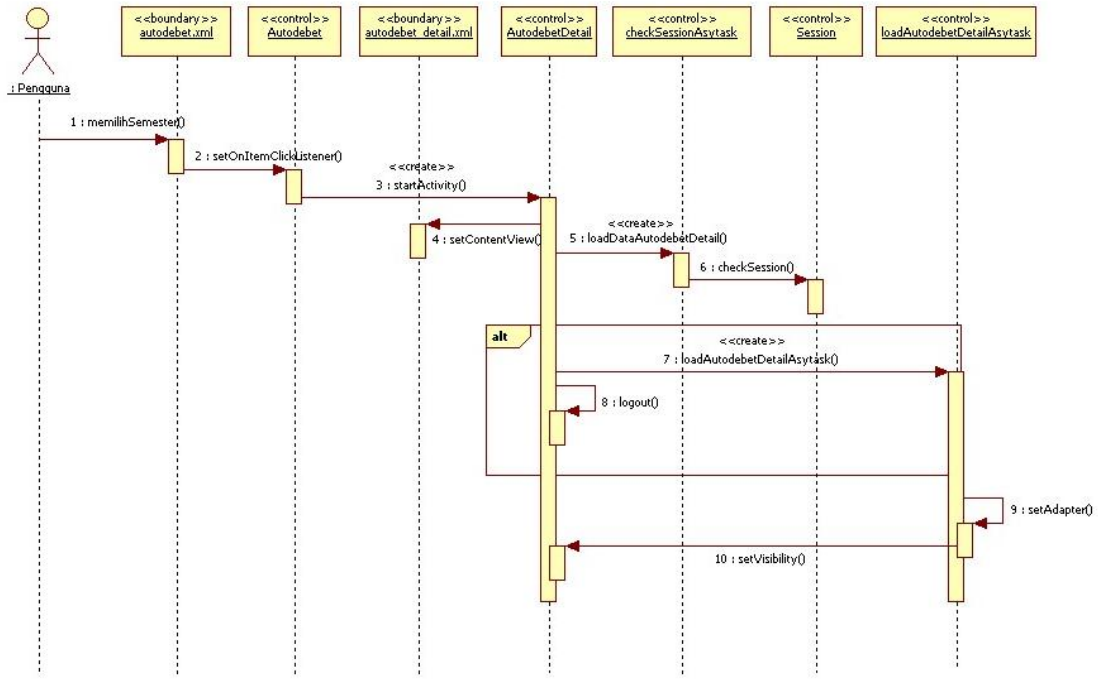
Squence diagram melihat daftar semester autodebet menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat daftar semester autodebet yang telah ditempuh pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat daftar semester autodebet dapat dilihat pada Gambar III.68.



Gambar III.68 *Squence Diagram* Melihat Daftar Semester Autodebet

21. *Squence Diagram* Melihat Detail Autodebet Semester

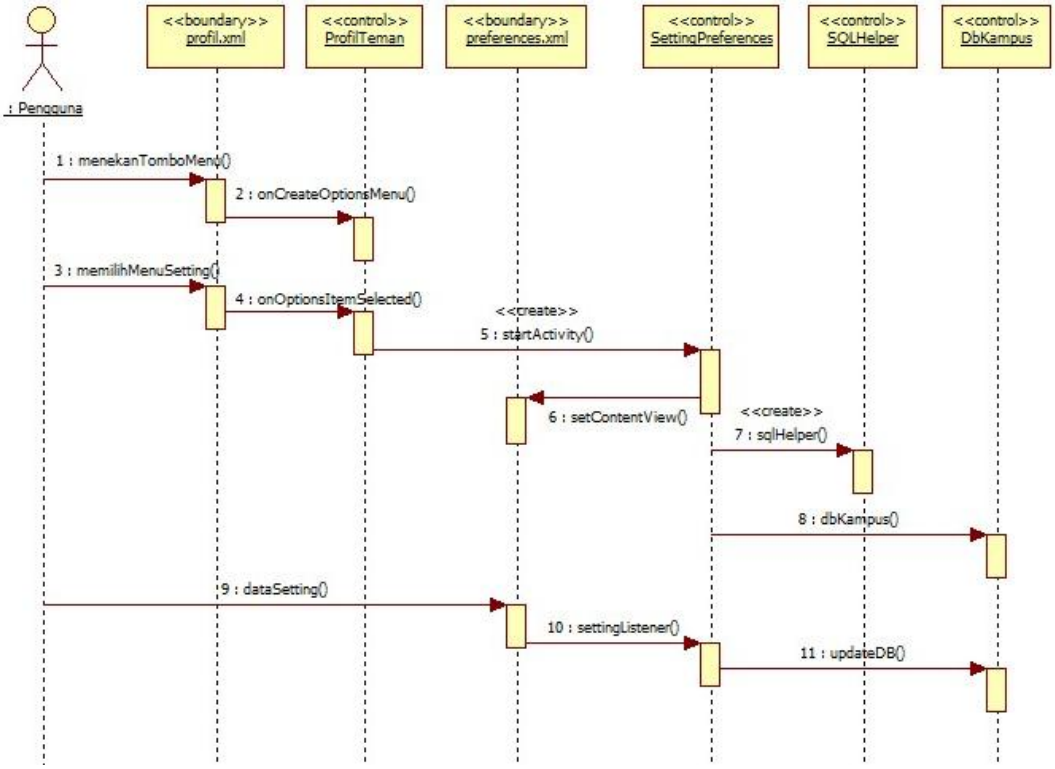
Squence diagram melihat detail autodebet semester menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat detail autodebet semester pengguna aplikasi *mobile* Kampus *Online*. *Squence diagram* melihat detail autodebet semester dapat dilihat pada Gambar III.69.



Gambar III.69 *Squence Diagram* Melihat Detail Autodebet Semester

22. *Squence Diagram Setting*

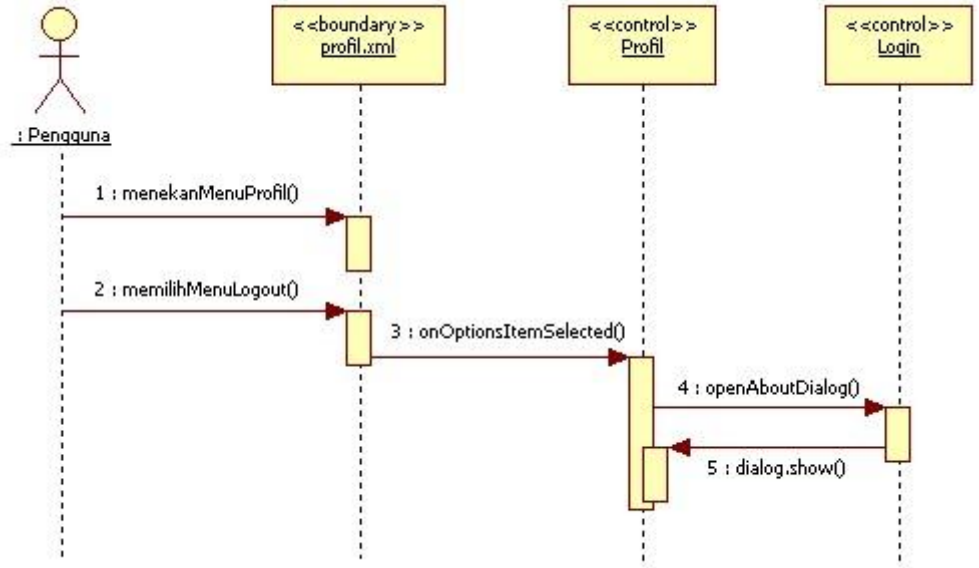
Squence diagram setting menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat halaman pengaturan atau *setting* aplikasi *mobile* Kampus *Online*. *Squence diagram setting* dapat dilihat pada Gambar III.70.



Gambar III.70 *Squence Diagram Setting*

23. *Squence Diagram About*

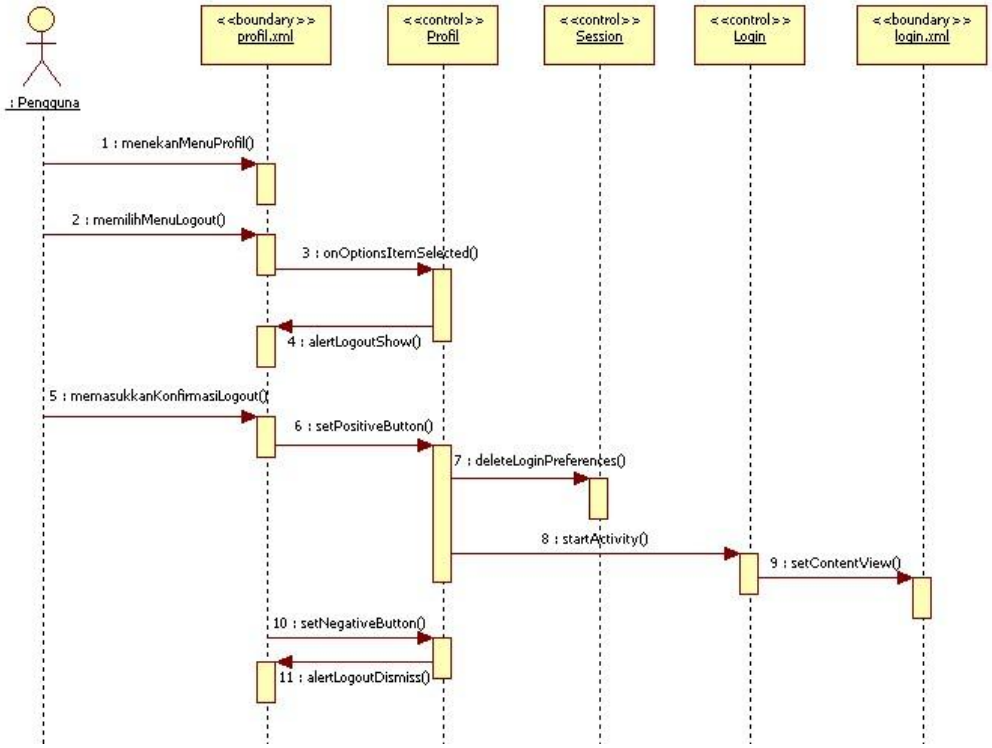
Squence diagram about menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam melihat halaman *about* aplikasi *mobile* Kampus *Online*. *Squence diagram about* dapat dilihat pada Gambar III.71.



Gambar III.71 *Squence Diagram About*

24. *Squence Diagram Logout*

Squence diagram logout menunjukkan rangkaian pesan yang dikirim antara objek pengguna dengan objek lainnya dalam proses *logout* pada aplikasi *mobile* Kampus *Online*. *Squence diagram logout* dapat dilihat pada Gambar III.72.



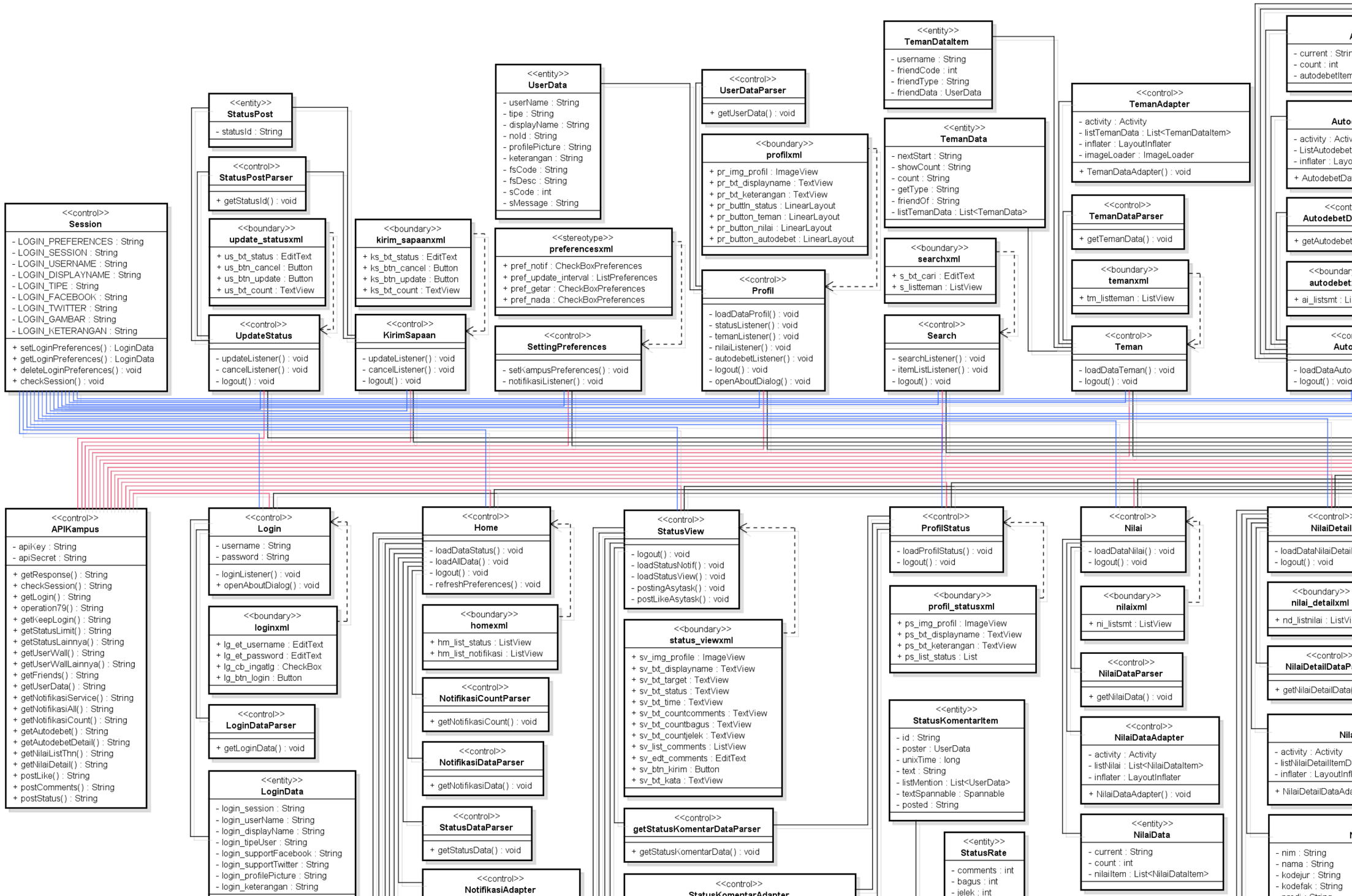
Gambar III.72 *Squence Diagram Logout*

3. 1. 9. 5. **Deskripsi Kelas**

Deskripsi kelas berisi pemaparan *class diagram* aplikasi *mobile* Kampus *Online* yang dibangun beserta daftar kelas yang dikelompokkan berdasarkan jenis kelas, dan spesifikasi masing-masing kelas yang memaparkan fungsi kelas dari segi atribut dan metode yang dimiliki setiap kelas.

3. 1. 9. 5. 1. *Class Diagram*

Class Diagram menggambarkan struktur dan hubungan antar objek-objek yang ada pada sistem. Struktur itu meliputi atribut-atribut dan metode-metode yang ada pada masing-masing kelas. Adapun gambaran *class diagram* dari aplikasi *mobile* Kampus *Online* yang dibangun dapat dilihat pada Gambar III.73.


<<control>>
Session
- LOGIN_PREFERENCES : String
- LOGIN_SESSION : String
- LOGIN_USERNAME : String
- LOGIN_DISPLAYNAME : String
- LOGIN_TIPE : String
- LOGIN_FACEBOOK : String
- LOGIN_TWITTER : String
- LOGIN_GAMBAR : String
- LOGIN_KETERANGAN : String
+ setLoginPreferences() : LoginData
+ getLoginPreferences() : LoginData
+ deleteLoginPreferences() : void
+ checkSession() : void
<<entity>>
StatusPost
- statusId : String
<<control>>
StatusPostParser
+ getStatusId() : void
<<boundary>>
update_statusxml
+ us_txt_status : EditText
+ us_btn_cancel : Button
+ us_btn_update : Button
+ us_txt_count : TextView
<<control>>
UpdateStatus
- updateListener() : void
- cancelListener() : void
- logout() : void
<<boundary>>
kirim_sapaanxml
+ ks_txt_status : EditText
+ ks_btn_cancel : Button
+ ks_btn_update : Button
+ ks_txt_count : TextView
<<control>>
KirimSapaan
- updateListener() : void
- cancelListener() : void
- logout() : void
<<entity>>
UserData
- userName : String
- tipe : String
- displayName : String
- noId : String
- profilePicture : String
- keterangan : String
- fsCode : String
- fsDesc : String
- sCode : int
- sMessage : String
<<stereotype>>
preferencesxml
+ pref_notif : CheckBoxPreferences
+ pref_update_interval : ListPreferences
+ pref_getar : CheckBoxPreferences
+ pref_nada : CheckBoxPreferences
<<control>>
SettingPreferences
- setKampusPreferences() : void
- notifikasiListener() : void
<<control>>
UserDataParser
+ getUserData() : void
<<boundary>>
profilxml
+ pr_img_profil : ImageView
+ pr_txt_displayname : TextView
+ pr_txt_keterangan : TextView
+ pr_buttln_status : LinearLayout
+ pr_button_teman : LinearLayout
+ pr_button_nilai : LinearLayout
+ pr_button_autodebet : LinearLayout
<<control>>
Profil
- loadDataProfil() : void
- statusListener() : void
- temanListener() : void
- nilaiListener() : void
- autodebetListener() : void
- logout() : void
- openAboutDialog() : void
<<entity>>
TemanDataItem
- username : String
- friendCode : int
- friendType : String
- friendData : UserData
<<entity>>
TemanData
- nextStart : String
- showCount : String
- count : String
- getType : String
- friendOf : String
- listTemanData : List<TemanData>
<<boundary>>
searchxml
+ s_txt_cari : EditText
+ s_listteman : ListView
<<control>>
Search
- searchListener() : void
- itemListListener() : void
- logout() : void
<<control>>
TemanAdapter
- activity : Activity
- listTemanData : List<TemanDataItem>
- inflater : LayoutInflater
- imageLoader : ImageLoader
+ TemanDataAdapter() : void
<<control>>
TemanDataParser
+ getTemanData() : void
<<boundary>>
temanxml
+ tm_listteman : ListView
<<control>>
Teman
- loadDataTeman() : void
- logout() : void
- current : Strin
- count : int
- autodebetItem
- activity : Activ
- ListAutodebet
- inflater : Layo
+ AutodebetDa
+ getAutodebet
+ ai_listsmt : Li
- loadDataAuto
- logout() : void
<<control>>
APIKampus
- apiKey : String
- apiSecret : String
+ getResponse() : String
+ checkSession() : String
+ getLogin() : String
+ operation79() : String
+ getKeepLogin() : String
+ getStatusLimit() : String
+ getStatusLainnya() : String
+ getUserWall() : String
+ getUserWallLainnya() : String
+ getFriends() : String
+ getUserData() : String
+ getNotifikasiService() : String
+ getNotifikasiAll() : String
+ getNotifikasiCount() : String
+ getAutodebet() : String
+ getAutodebetDetail() : String
+ getNilaiListThn() : String
+ getNilaiDetail() : String
+ postLike() : String
+ postComments() : String
+ postStatus() : String
<<control>>
Login
- username : String
- password : String
- loginListener() : void
+ openAboutDialog() : void
<<boundary>>
loginxml
+ lg_et_username : EditText
+ lg_et_password : EditText
+ lg_cb_ingatlg : CheckBox
+ lg_btn_login : Button
<<control>>
LoginDataParser
+ getLoginData() : void
<<entity>>
LoginData
- login_session : String
- login_userName : String
- login_displayName : String
- login_tipeUser : String
- login_supportFacebook : String
- login_supportTwitter : String
- login_profilePicture : String
- login_keterangan : String
<<control>>
Home
- loadDataStatus() : void
- loadAllData() : void
- logout() : void
- refreshPreferences() : void
<<boundary>>
homexml
+ hm_list_status : ListView
+ hm_list_notifikasi : ListView
<<control>>
NotifikasiCountParser
+ getNotifikasiCount() : void
<<control>>
NotifikasiDataParser
+ getNotifikasiData() : void
<<control>>
StatusDataParser
+ getStatusData() : void
<<control>>
NotifikasiAdapter
<<control>>
StatusView
- logout() : void
- loadStatusNotif() : void
- loadStatusView() : void
- postingAsytask() : void
- postLikeAsytask() : void
<<boundary>>
status_viewxml
+ sv_img_profile : ImageView
+ sv_txt_displayname : TextView
+ sv_txt_target : TextView
+ sv_txt_status : TextView
+ sv_txt_time : TextView
+ sv_txt_countcomments : TextView
+ sv_txt_countbagus : TextView
+ sv_txt_countjelek : TextView
+ sv_list_comments : ListView
+ sv_edt_comments : EditText
+ sv_btn_kirim : Button
+ sv_txt_kata : TextView
<<control>>
getStatusKomentarDataParser
+ getStatusKomentarData() : void
<<control>>
StatusKomentarAdapter
<<control>>
ProfilStatus
- loadProfilStatus() : void
- logout() : void
<<boundary>>
profil_statusxml
+ ps_img_profil : ImageView
+ ps_txt_displayname : TextView
+ ps_txt_keterangan : TextView
+ ps_list_status : List
<<entity>>
StatusKomentarItem
- id : String
- poster : UserData
- unixTime : long
- text : String
- listMention : List<UserData>
- textSpannable : Spannable
- posted : String
<<entity>>
StatusRate
- comments : int
- bagus : int
- jelek : int
<<control>>
Nilai
- loadDataNilai() : void
- logout() : void
<<boundary>>
nilaixml
+ ni_listsmt : ListView
<<control>>
NilaiDataParser
+ getNilaiData() : void
<<control>>
NilaiDataAdapter
- activity : Activity
- listNilai : List<NilaiDataItem>
- inflater : LayoutInflater
+ NilaiDataAdapter() : void
<<entity>>
NilaiData
- current : String
- count : int
- nilaiItem : List<NilaiDataItem>
- loadDataNilaiDetai
- logout() : void
<<boundary>>
nilai_detailxml
+ nd_listnilai : ListVi
<<control>>
NilaiDetailDataP
+ getNilaiDetailData
- activity : Activity
- listNilaiDetailItemD
- inflater : LayoutInfl
+ NilaiDetailDataAda
- nim : String
- nama : String
- kodejur : String
- kodefak : String

3. 1. 9. 5. 2. Daftar Kelas

Spesifikasi kelas berisi pemaparan setiap kelas yang terdapat pada *class diagram* aplikasi *mobile* Kampus *Online* (Gambar III.73). *Class diagram* aplikasi *mobile* Kampus *Online* yang dibangun terdiri dari tiga jenis kelas, yaitu *class boundary*, *class control*, dan *class entity*. Adapun daftar setiap kelas yang terdapat pada *class diagram* aplikasi *mobile* Kampus *Online* berdasarkan jenisnya dapat dilihat pada Tabel III.55, Tabel III.56, dan Tabel III.57.

Tabel III.55 Daftar Spesifikasi Kelas

No	Nama Kelas	Jenis Kelas
1	login.xml	*Boundary*
2	home.xml	*Boundary*
3	status_view.xml	*Boundary*
4	profil.xml	*Boundary*
5	profil_status.xml	*Boundary*
6	teman.xml	*Boundary*
7	update_status.xml	*Boundary*
8	kirim_sapaan.xml	*Boundary*
9	preferences.xml	*Boundary*
10	nilai.xml	*Boundary*
11	nilai_detail_data.xml	*Boundary*
12	nilai_detail_informasi.xml	*Boundary*
13	nilai_keterangan.xml	*Boundary*
14	autodebet.xml	*Boundary*
15	autodebet_detail.xml	*Boundary*
16	search.xml	*Boundary*

Tabel III.56 Daftar Spesifikasi Kelas (Lanjutan)

No	Nama Kelas	Jenis Kelas
17	Login	*Control*
18	Home	*Control*
19	StatusView	*Control*
20	Profil	*Control*
21	ProfilStatus	*Control*
22	Teman	*Control*
23	UpdateStatus	*Control*
24	KirimSapaan	*Control*
25	SettingPreferences	*Control*
26	Nilai	*Control*
27	NilaiDetail	*Control*
28	NilaiDetailInformasi	*Control*
29	NilaiKeterangan	*Control*
30	Autodebet	*Control*
31	AutodebetDetail	*Control*
32	Search	*Control*
33	LoginDataParser	*Control*
34	NotifikasiCountParser	*Control*
35	NotifikasiDataParser	*Control*
36	StatusDataParser	*Control*
37	StatusKomentarParser	*Control*
38	UserDataParser	*Control*
39	TemanDataParser	*Control*
40	StatusPostParser	*Control*
41	NilaiDataParser	*Control*
42	NilaiDetailDataParser	*Control*
43	AutodebetDataParser	*Control*
44	AutodebetDetailDataParser	*Control*

Tabel III.57 Daftar Spesifikasi Kelas (Lanjutan)

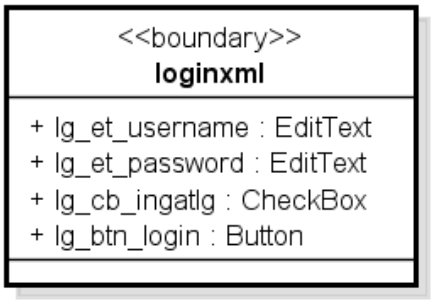
No	Nama Kelas	Jenis Kelas
45	APIKampus	*Control*
46	APIConnections	*Control*
47	Session	*Control*
48	NotifikasiAdapter	*Control*
49	StatusAdapter	*Control*
50	StatusKomentarAdapter	*Control*
51	TemanAdapter	*Control*
52	NilaiDataAdapter	*Control*
53	NilaiDetailDataAdapter	*Control*
54	AutodebetDataAdapter	*Control*
55	AutodebetDetailDataAdapter	*Control*
56	LoginData	*Entity*
57	StatusData	*Entity*
58	StatusItem	*Entity*
59	StatusRate	*Entity*
60	NotifikasiData	*Entity*
61	NotifikasiItem	*Entity*
62	StatusKomentar	*Entity*
63	StatusKomentarItem	*Entity*
64	UserData	*Entity*
65	TemanData	*Entity*
66	TemanDataItem	*Entity*
67	StatusPost	*Entity*
68	NilaiData	*Entity*
69	NilaiDetailData	*Entity*
70	NilaiDetailItemData	*Entity*
71	AutodebetData	*Entity*
72	AutodebetDataItem	*Entity*
73	AutodebetDetailData	*Entity*

3. 1. 9. 5. 3. Spesifikasi Kelas

Spesifikasi kelas berisi penjelasan detail atribut dan fungsi metode yang dimiliki setiap kelas. Berikut ini penjelasan dari setiap kelas yang telah didaftarkan pada Tabel III.55, Tabel III.56, dan Tabel III.57.

1. Kelas login.xml

Kelas login.xml merupakan kelas yang digunakan untuk menampilkan halaman *login* aplikasi *mobile* Kampus *Online* Unikom. Berikut ini atribut dan metode yang dimiliki oleh kelas login.xml yang dapat dilihat pada Gambar III.74.



Gambar III.74 Kelas login.xml

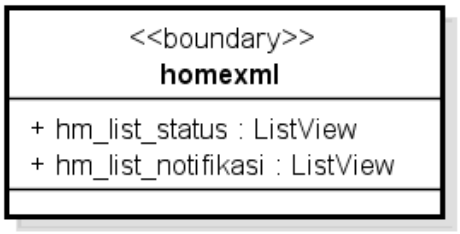
Adapun penjelasan dari setiap atribut dari kelas login.xml dapat dijelaskan pada Tabel III.58

Tabel III.58 Deskripsi Atribut Kelas login.xml

Nama	Tipe	Keterangan
lg_et_username	EditText	Digunakan untuk mengambil *username* pengguna pada saat *login*
lg_et_password	EditText	Digunakan untuk mengambil *password* pengguna pada saat *login*
lg_cb_ingatlg	CheckBox	Digunakan untuk mengambil pilihan pengguna berupa ingat *login* atau *login* biasa
lg_btn_login	Button	Digunakan untuk melakukan aksi *login*

2. Kelas home.xml

Kelas home.xml merupakan kelas yang digunakan untuk menampilkan halaman *home* dan notifikasi aplikasi *mobile* Kampus *Online* Unikom. Berikut ini atribut dan metode yang dimiliki oleh kelas home.xml yang dapat dilihat pada Gambar III.75.



Gambar III.75 Kelas home.xml

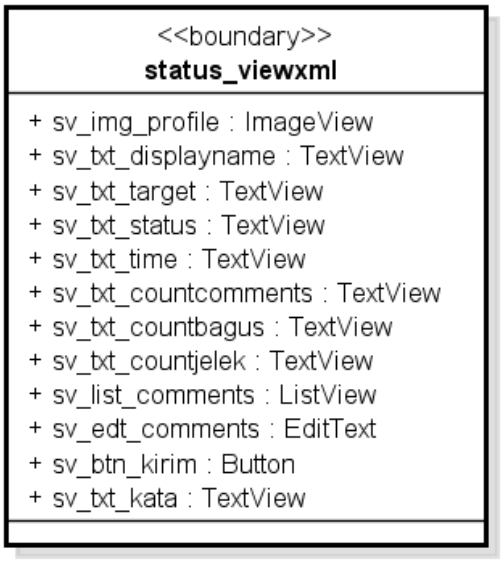
Adapun penjelasan dari setiap atribut dari kelas home.xml dapat dijelaskan pada Tabel III.59

Tabel III.59 Deskripsi Atribut Kelas home.xml

Nama	Tipe	Keterangan
hm_list_status	ListView	Digunakan untuk menampilkan daftar status di halaman *home* pengguna
hm_list_notifikasi	ListView	Digunakan untuk menampilkan daftar notifikasi yang diterima pengguna

3. Kelas status_view.xml

Kelas status_view.xml merupakan kelas yang digunakan untuk menampilkan detail dari status. Berikut ini atribut dan metode yang dimiliki oleh kelas status_view.xml yang dapat dilihat pada Gambar III.76.



Gambar III.76 Kelas status_view.xml

Adapun penjelasan dari setiap atribut dari kelas status_view.xml dapat dijelaskan pada Tabel III.60 dan Tabel III.61.

Tabel III.60 Deskripsi Atribut Kelas status_view.xml

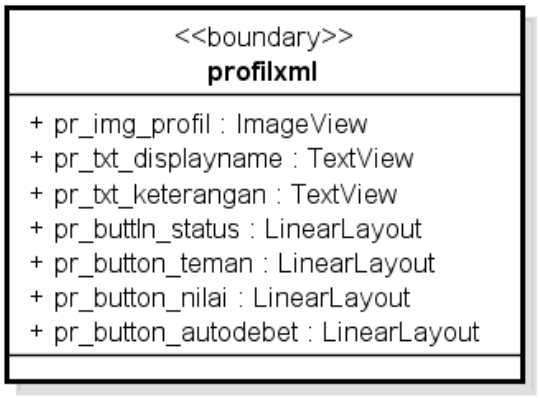
Nama	Tipe	Keterangan
sv_img_profile	ImageView	Digunakan untuk menampilkan foto pengguna
sv_txt_displayname	TextView	Digunakan untuk menampilkan nama pengguna pemilik status
sv_txt_target	TextView	Digunakan untuk menampilkan nama target jika status berupa sapaan
sv_txt_status	TextView	Digunakan untuk menampilkan status
sv_txt_time	TextView	Digunakan untuk menampilkan waktu pembuatan status
sv_txt_countcomments	TextView	Digunakan untuk menampilkan jumlah komentar dari status
sv_txt_countbagus	TextView	Digunakan untuk menampilkan jumlah rating bagus
sv_txt_countjelek	TextView	Digunakan untuk menampilkan jumlah rating jelek

Tabel III.61 Deskripsi Atribut Kelas status_view.xml (Lanjutan)

Nama	Tipe	Keterangan
sv_list_comments	ListView	Digunakan untuk menampilkan komentar dari status yang bersangkutan
sv_edt_comments	EditText	Digunakan untuk mengambil komentar yang ditambahkan oleh pengguna
sv_btn_kirim	Button	Digunakan untuk mengirimkan komentar terhadap status
sv_txt_kata	TextView	Digunakan untuk menampilkan jumlah sisa karakter dari komentar

4. Kelas profil.xml

Kelas profil.xml merupakan kelas yang digunakan untuk menampilkan data profil pengguna. Berikut ini atribut dan metode yang dimiliki oleh kelas profil.xml yang dapat dilihat pada Gambar III.77.



Gambar III.77 Kelas profil.xml

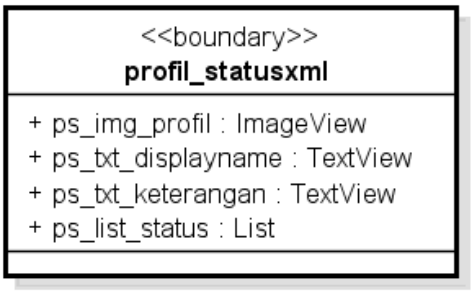
Adapun penjelasan dari setiap atribut dari kelas profil.xml dapat dijelaskan pada Tabel III.62.

Tabel III.62 Deskripsi Atribut Kelas profil.xml

Nama	Tipe	Keterangan
pr_img_profil	ImageView	Digunakan untuk menampilkan foto profil pengguna
pr_txt_displayname	TextView	Digunakan untuk menampilkan nama pengguna
pr_txt_keterangan	TextView	Digunakan untuk menampilkan keterangan dari pengguna
pr_button_status	LinearLayout	Digunakan untuk menuju ke halaman status profil
pr_button_teman	LinearLayout	Digunakan untuk menuju ke halaman teman
pr_button_nilai	LinearLayout	Digunakan untuk menuju ke halaman nilai
pr_button_autodebet	LinearLayout	Digunakan untuk menuju ke halaman autodebet

5. Kelas profil_status.xml

Kelas profil_status.xml merupakan kelas yang digunakan untuk menampilkan data status dari pengguna. Berikut ini atribut dan metode yang dimiliki oleh kelas profil_status.xml yang dapat dilihat pada Gambar III.78.



Gambar III.78 Kelas profil_status.xml

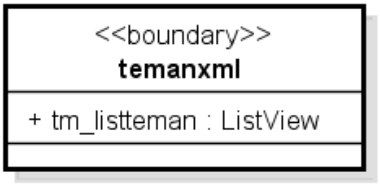
Adapun penjelasan dari setiap atribut dari kelas profil_status.xml dapat dijelaskan pada Tabel III.63.

Tabel III.63 Deskripsi Atribut Kelas profil_status.xml

Nama	Tipe	Keterangan
ps_img_profil	ImageView	Digunakan untuk menampilkan foto profil pengguna
ps_txt_displayname	TextView	Digunakan untuk menampilkan nama pengguna
ps_txt_keterangan	TextView	Digunakan untuk menampilkan keterangan dari pengguna
ps_list_status	ListView	Digunakan untuk menampilkan daftar status dari pengguna

6. Kelas teman.xml

Kelas teman.xml merupakan kelas yang digunakan untuk menampilkan daftar teman dari pengguna. Berikut ini atribut dan metode yang dimiliki oleh kelas teman.xml yang dapat dilihat pada Gambar III.79.



Gambar III.79 Kelas teman.xml

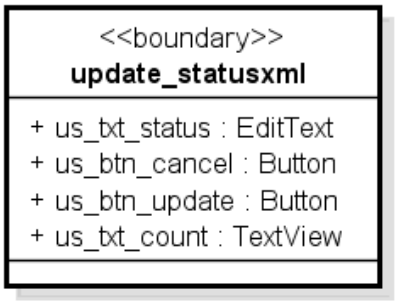
Adapun penjelasan dari setiap atribut dari kelas teman.xml dapat dijelaskan pada Tabel III.64.

Tabel III.64 Deskripsi Atribut Kelas teman.xml

Nama	Tipe	Keterangan
tm_listteman	ListView	Digunakan untuk menampilkan daftar teman dari pengguna

7. Kelas update_status.xml

Kelas update_status.xml merupakan kelas yang digunakan untuk menampilkan halaman *update status*. Berikut ini atribut dan metode yang dimiliki oleh kelas update_status.xml yang dapat dilihat pada Gambar III.80.



Gambar III.80 Kelas update_status.xml

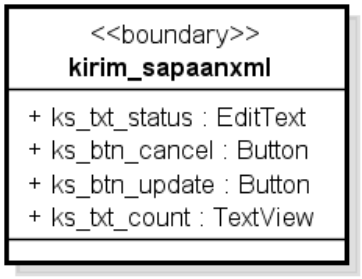
Adapun penjelasan dari setiap atribut dari kelas update_status.xml dapat dijelaskan pada Tabel III.65.

Tabel III.65 Deskripsi Atribut Kelas update_status.xml

Nama	Tipe	Keterangan
us_txt_status	EditText	Digunakan untuk mengambil status pengguna
us_btn_cancel	Button	Digunakan untuk membatalkan *update status*
us_btn_update	Button	Digunakan untuk mengirimkan status
us_txt_count	TextView	Digunakan untuk menampilkan jumlah sisa karakter dari status

8. Kelas kirim_sapaan.xml

Kelas kirim_sapaan.xml merupakan kelas yang digunakan untuk menampilkan halaman kirim sapaan. Berikut ini atribut dan metode yang dimiliki oleh kelas kirim_sapaan.xml yang dapat dilihat pada Gambar III.81.



Gambar III.81 Kelas kirim_sapaan.xml

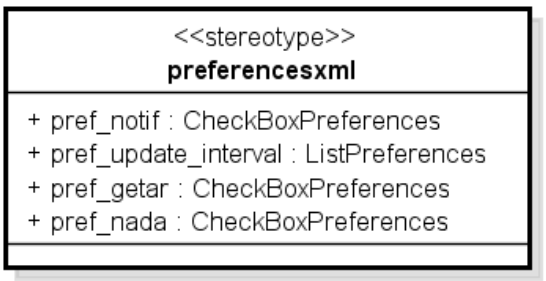
Adapun penjelasan dari setiap atribut dari kelas kirim_sapaan.xml dapat dijelaskan pada Tabel III.66.

Tabel III.66 Deskripsi Atribut Kelas kirim_sapaan.xml

Nama	Tipe	Keterangan
ks_txt_status	EditText	Digunakan untuk mengambil sapaan dari pengguna
ks_btn_cancel	Button	Digunakan untuk membatalkan kirim sapaan
ks_btn_update	Button	Digunakan untuk mengirimkan sapaan
ks_txt_count	TextView	Digunakan untuk menampilkan jumlah sisa karakter dari sapaan

9. Kelas preferences.xml

Kelas preferences.xml merupakan kelas yang digunakan untuk menampilkan halaman pengaturan aplikasi *mobile* Kampus *Online* Unikom. Berikut ini atribut dan metode yang dimiliki oleh kelas preferences.xml yang dapat dilihat pada Gambar III.82.



Gambar III.82 Kelas preferences.xml

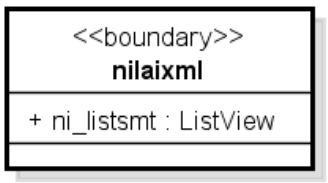
Adapun penjelasan dari setiap atribut dari kelas preferences.xml dapat dijelaskan pada Tabel III.67.

Tabel III.67 Deskripsi Atribut Kelas preferences.xml

Nama	Tipe	Keterangan
pref_notif	CheckBoxPreferences	Digunakan untuk mengatur notifikasi
pref_update_interval	ListPreferences	Digunakan untuk mengatur interval update otomatis
pref_getar	CheckBoxPreferences	Digunakan untuk mengatur getar notifikasi
pref_nada	CheckBoxPreferences	Digunakan untuk mengatur nada notifikasi

10. Kelas nilai.xml

Kelas nilai.xml merupakan kelas yang digunakan untuk menampilkan daftar semester yang telah ditempuh. Berikut ini atribut dan metode yang dimiliki oleh kelas nilai.xml yang dapat dilihat pada Gambar III.83.



Gambar III.83 Kelas nilai.xml

Adapun penjelasan dari setiap atribut dari kelas nilai.xml dapat dijelaskan pada Tabel III.68.

Tabel III.68 Deskripsi Atribut Kelas nilai.xml

Nama	Tipe	Keterangan
ni_listsmt	ListView	Digunakan untuk menampilkan daftar semester yang telah ditempuh

11. Kelas nilai_detail.xml

Kelas nilai_detail.xml merupakan kelas yang digunakan untuk menampilkan detail nilai yang diperoleh oleh mahasiswa berdasarkan semester yang dipilih. Berikut ini atribut dan metode yang dimiliki oleh kelas nilai_detail.xml yang dapat dilihat pada Gambar III.84.



Gambar III.84 Kelas nilai_detail.xml

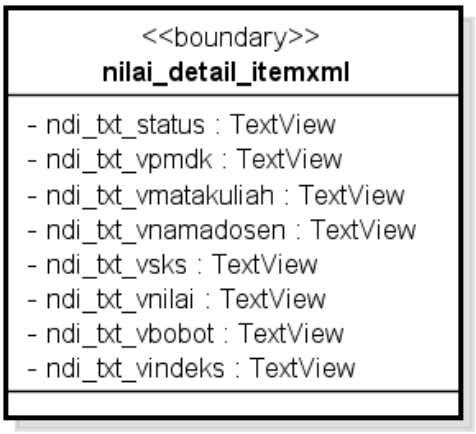
Adapun penjelasan dari setiap atribut dari kelas nilai_detail.xml dapat dijelaskan pada Tabel III.69.

Tabel III.69 Deskripsi Atribut Kelas nilai_detail.xml

Nama	Tipe	Keterangan
nd_listnilai	ListView	Digunakan untuk menampilkan detail nilai yang diperoleh oleh mahasiswa berdasarkan semester yang dipilih

12. Kelas nilai_detail_informasi.xml

Kelas nilai_detail_informasi.xml merupakan kelas yang digunakan untuk menampilkan data detail dari item nilai yang ditampilkan. Berikut ini atribut dan metode yang dimiliki oleh kelas nilai_detail_informasi.xml yang dapat dilihat pada Gambar III.85.



Gambar III.85 Kelas nilai_detail_informasi.xml

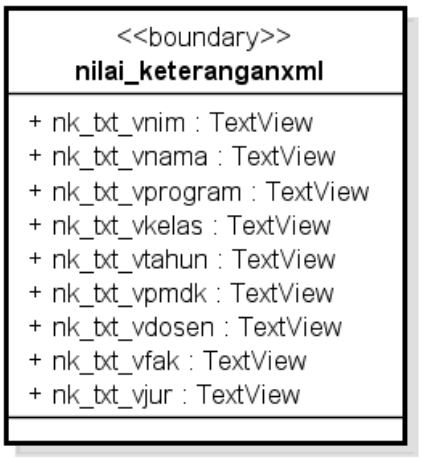
Adapun penjelasan dari setiap atribut dari kelas nilai_detail_informasi.xml dapat dijelaskan pada Tabel III.70.

Tabel III.70 Deskripsi Atribut Kelas nilai_detail_informasi.xml

Nama	Tipe	Keterangan
ndi_txt_vstatus	TextView	Digunakan untuk menampilkan status nilai sudah di *upload* atau belum.
ndi_txt_vkdmk	TextView	Digunakan untuk menampilkan kode mata kuliah
ndi_txt_vmatakuliah	TextView	Digunakan untuk menampilkan nama mata kuliah
ndi_txt_vnamadosen	TextView	Digunakan untuk menampilkan nama dosen
ndi_txt_vsks	TextView	Digunakan untuk menampilkan sks dari mata kuliah yang bersangkutan
ndi_txt_vnilai	TextView	Digunakan untuk menampilkan nilai
ndi_txt_vbobot	TextView	Digunakan untuk menampilkan bobot (sks x nilai)
ndi_txt_vindeks	TextView	Digunakan untuk menampilkan indeks dari mata kuliah

13. Kelas nilai_keterangan.xml

Kelas nilai_keterangan.xml merupakan kelas yang digunakan untuk menampilkan data lengkap dari mahasiswa yang *login*. Berikut ini atribut dan metode yang dimiliki oleh kelas nilai_keterangan.xml yang dapat dilihat pada Gambar III.86.



Gambar III.86 Kelas nilai_keterangan.xml

Adapun penjelasan dari setiap atribut dari kelas nilai_keterangan.xml dapat dijelaskan pada Tabel III.71 dan Tabel III.72.

Tabel III.71 Deskripsi Atribut Kelas nilai_keterangan.xml

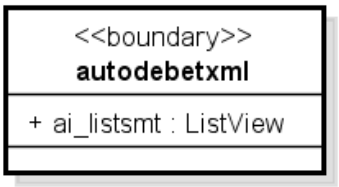
Nama	Tipe	Keterangan
nk_txt_vnim	TextView	Digunakan untuk menampilkan nim dari mahasiswa yang *login*
nk_txt_vnama	TextView	Digunakan untuk menampilkan nama dari mahasiswa yang *login*
nk_txt_vprogram	TextView	Digunakan untuk menampilkan program studi yang diambil oleh mahasiswa yang *login*
nk_txt_vkelas	TextView	Digunakan untuk menampilkan kelas dari mahasiswa yang *login*

Tabel III.72 Deskripsi Atribut Kelas nilai_keterangan.xml (Lanjutan)

Nama	Tipe	Keterangan
nk_txt_vtahun	TextView	Digunakan untuk menampilkan tahun masuk dari mahasiswa yang *login*
nk_txt_vpmdk	TextView	Digunakan untuk menampilkan status PMDK atau tidak dari mahasiswa yang *login*
nk_txt_vdosen	TextView	Digunakan untuk menampilkan nama wali dosen dari mahasiswa yang *login*
nk_txt_vfak	TextView	Digunakan untuk menampilkan fakultas dari mahasiswa yang *login*
nk_txt_vjur	TextView	Digunakan untuk menampilkan jurusan dari mahasiswa yang *login*

14. Kelas autodebet.xml

Kelas autodebet.xml merupakan kelas yang digunakan untuk menampilkan daftar tahun akademik yang telah ditempuh untuk mengecek data autodebet mahasiswa. Berikut ini atribut dan metode yang dimiliki oleh kelas autodebet.xml yang dapat dilihat pada Gambar III.87.



Gambar III.87 Kelas autodebet.xml

Adapun penjelasan dari setiap atribut dari kelas nilai.xml dapat dijelaskan pada Tabel III.73.

Tabel III.73 Deskripsi Atribut Kelas autodebet.xml

Nama	Tipe	Keterangan
ai_listsmt	ListView	Digunakan untuk menampilkan daftar tahun akademik yang telah ditempuh

15. Kelas autodebet_detail.xml

Kelas autodebet_detail.xml merupakan kelas yang digunakan untuk menampilkan detail biaya autodebet yang telah dibayarkan oleh mahasiswa berdasarkan tahun akademik yang dipilih. Berikut ini atribut dan metode yang dimiliki oleh kelas autodebet_detail.xml yang dapat dilihat pada Gambar III.88.



Gambar III.88 Kelas autodebet_detail.xml

Adapun penjelasan dari setiap atribut dari kelas autodebet_detail.xml dapat dijelaskan pada Tabel III.74.

Tabel III.74 Deskripsi Atribut Kelas autodebet_detail.xml

Nama	Tipe	Keterangan
ad_listautodebet	ListView	Digunakan untuk menampilkan detail biaya autodebet yang telah dibayarkan oleh mahasiswa berdasarkan tahun akademik yang dipilih

16. Kelas search.xml

Kelas search.xml merupakan kelas yang digunakan untuk mengambil dan menampilkan data pengguna Kampus *Online*. Berikut ini atribut dan metode yang dimiliki oleh kelas search.xml yang dapat dilihat pada Gambar III.89.



Gambar III.89 Kelas search.xml

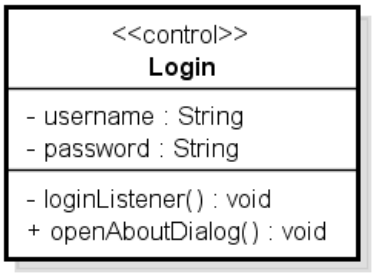
Adapun penjelasan dari setiap atribut dari kelas search.xml dapat dijelaskan pada Tabel III.75.

Tabel III.75 Deskripsi Atribut Kelas search.xml

Nama	Tipe	Keterangan
s_txt_cari	EditText	Digunakan untuk mengambil data *username* pengguna yang akan dicari
s_listteman	ListView	Digunakan untuk menampilkan hasil pencarian pengguna Kampus *Online*

17. Kelas *Login*

Kelas *login* merupakan kelas yang digunakan untuk mendapatkan hak akses masuk ke dalam aplikasi *mobile* Kampus *Online*. Berikut ini atribut dan metode yang dimiliki oleh kelas *login* yang dapat dilihat pada Gambar III.90.



Gambar III.90 Kelas *Login*

Adapun penjelasan dari setiap atribut dari kelas *login* dapat dijelaskan pada Tabel III.76.

Tabel III.76 Deskripsi Atribut Kelas *Login*

Nama	Tipe	Keterangan
username	String	Digunakan untuk mengambil *username* pengguna yang akan *login*
password	String	Digunakan untuk mengambil *password* pengguna yang akan *login*

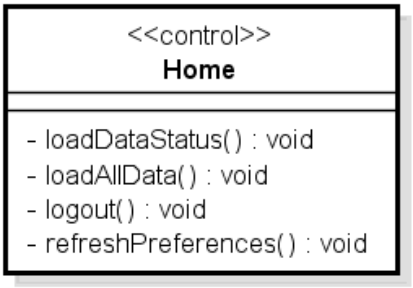
Sementara itu penjelasan dari setiap metode dari kelas *login* dapat dijelaskan pada Tabel III.77.

Tabel III.77 Deskripsi Metode Kelas *Login*

Nama	Keterangan
loginListener()	Digunakan sebagai aksi saat pengguna menekan tombol *login* untuk masuk ke dalam aplikasi *mobile* Kampus *Online*
openAboutDialog()	Digunakan sebagai aksi saat pengguna menu *about* untuk menampilkan keterangan aplikasi *mobile* Kampus *Online*

18. Kelas *Home*

Kelas *home* merupakan kelas yang digunakan untuk menampilkan halaman *home* dan notifikasi aplikasi *mobile* Kampus *Online*. Berikut ini atribut dan metode yang dimiliki oleh kelas *home* yang dapat dilihat pada Gambar III.91.



Gambar III.91 Kelas *Home*

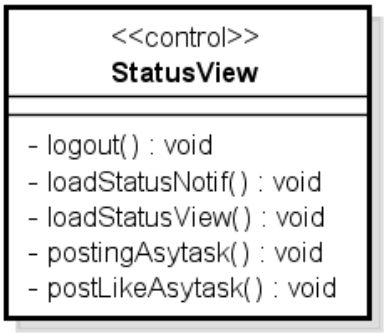
Adapun penjelasan dari setiap metode dari kelas *home* dapat dijelaskan pada Tabel III.78.

Tabel III.78 Deskripsi Metode Kelas *Home*

Nama	Keterangan
loadDataStatus()	Digunakan untuk mengambil data status pengguna
loadAllData()	Digunakan untuk mengambil data status dan data notifikasi dari pengguna yang *login* ke dalam aplikasi *mobile* Kampus *Online*
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*
refreshPreferences()	Digunakan untuk mengambil data pengaturan aplikasi *mobile* Kampus *Online*

19. Kelas *StatusView*

Kelas *StatusView* merupakan kelas yang digunakan untuk menampilkan detail dari status. Berikut ini atribut dan metode yang dimiliki oleh kelas *StatusView* yang dapat dilihat pada Gambar III.92.



Gambar III.92 Kelas *StatusView*

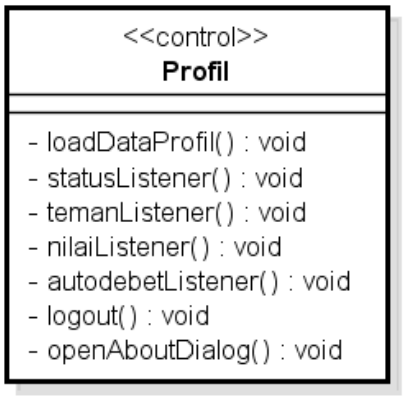
Adapun penjelasan dari setiap metode dari kelas *StatusView* dapat dijelaskan pada Tabel III.79.

Tabel III.79 Deskripsi Metode Kelas *StatusView*

Nama	Keterangan
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*
loadStatusNotif()	Digunakan untuk mengambil data status yang diambil dari notifikasi pada halaman *Home*
loadStatusView()	Digunakan untuk mengambil data status yang diambil dari daftar status pada halaman *Home*
postingAsytask()	Digunakan untuk menambahkan data komentar pada status
postLikeAsytask()	Digunakan untuk menambahkan data *rating* pada status

20. Kelas Profil

Kelas Profil merupakan kelas yang digunakan untuk menampilkan data pengguna Kampus *Online*. Berikut ini atribut dan metode yang dimiliki oleh kelas Profil yang dapat dilihat pada Gambar III.93.



Gambar III.93 Kelas Profil

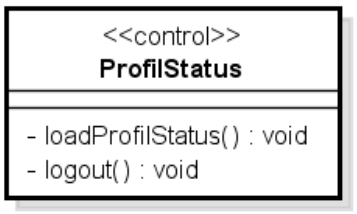
Adapun penjelasan dari setiap metode dari kelas Profil dapat dijelaskan pada Tabel III.80.

Tabel III.80 Deskripsi Metode Kelas Profil

Nama	Keterangan
loadDataProfil()	Digunakan untuk mengambil data pengguna
statusListener ()	Digunakan untuk menuju ke halaman status profil
temanListener()	Digunakan untuk menuju ke halaman teman
nilaiListener()	Digunakan untuk menuju ke halaman nilai
autodebetListener()	Digunakan untuk menuju ke halaman autodebet
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*
openAboutDialog()	Digunakan sebagai aksi saat pengguna menu *about* untuk menampilkan keterangan aplikasi *mobile* Kampus *Online*

21. Kelas ProfilStatus

Kelas ProfilStatus merupakan kelas yang digunakan untuk menampilkan data status dari satu pengguna tertentu. Berikut ini atribut dan metode yang dimiliki oleh kelas ProfilStatus yang dapat dilihat pada Gambar III.94.



Gambar III.94 Kelas ProfilStatus

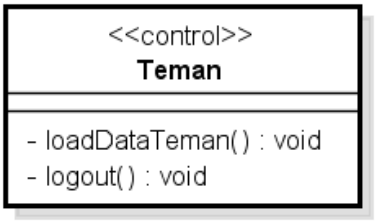
Adapun penjelasan dari setiap metode dari kelas ProfilStatus dapat dijelaskan pada Tabel III.81.

Tabel III.81 Deskripsi Metode Kelas ProfilStatus

Nama	Keterangan
loadProfilStatus()	Digunakan untuk mengambil data status dari satu pengguna tertentu
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*

22. Kelas Teman

Kelas Teman merupakan kelas yang digunakan untuk menampilkan data teman dari satu pengguna tertentu. Berikut ini atribut dan metode yang dimiliki oleh kelas Teman yang dapat dilihat pada Gambar III.95.



Gambar III.95 Kelas Teman

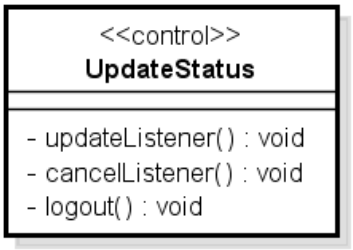
Adapun penjelasan dari setiap metode dari kelas Teman dapat dijelaskan pada Tabel III.82.

Tabel III.82 Deskripsi Metode Kelas Teman

Nama	Keterangan
loadDataTeman()	Digunakan untuk mengambil data teman dari satu pengguna tertentu
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*

23. Kelas UpdateStatus

Kelas UpdateStatus merupakan kelas yang digunakan untuk melakukan *update status*. Berikut ini atribut dan metode yang dimiliki oleh kelas UpdateStatus yang dapat dilihat pada Gambar III.96.



Gambar III.96 Kelas UpdateStatus

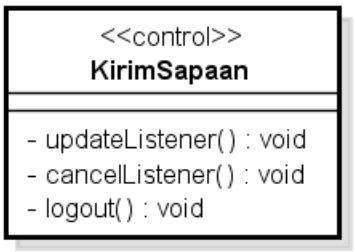
Adapun penjelasan dari setiap metode dari kelas UpdateStatus dapat dijelaskan pada Tabel III.83.

Tabel III.83 Deskripsi Metode Kelas UpdateStatus

Nama	Keterangan
updateListener()	Digunakan untuk melakukan aksi *update status*
cancelListener()	Digunakan untuk membatalkan *update status*
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*

24. Kelas KirimSapaan

Kelas KirimSapaan merupakan kelas yang digunakan untuk mengirim sapaan ke pengguna Kampus Online lainnya. Berikut ini atribut dan metode yang dimiliki oleh kelas KirimSapaan yang dapat dilihat pada Gambar III.97.



Gambar III.97 Kelas KirimSapaan

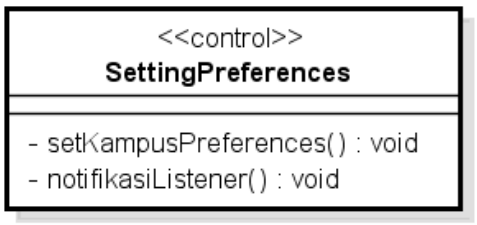
Adapun penjelasan dari setiap metode dari kelas KirimSapaan dapat dijelaskan pada Tabel III.84.

Tabel III.84 Deskripsi Metode Kelas KirimSapaan

Nama	Keterangan
updateListener()	Digunakan untuk melakukan aksi kirim sapaan
cancelListener()	Digunakan untuk membatalkan kirim sapaan
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*

25. Kelas SettingPreferences

Kelas SettingPreferences merupakan kelas yang digunakan untuk melakukan pengaturan aplikasi *mobile* Kampus *Online*. Berikut ini atribut dan metode yang dimiliki oleh kelas SettingPreferences yang dapat dilihat pada Gambar III.98.



Gambar III.98 Kelas SettingPreferences

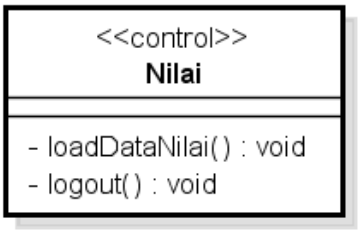
Adapun penjelasan dari setiap metode dari kelas SettingPreferences dapat dijelaskan pada Tabel III.85.

Tabel III.85 Deskripsi Metode Kelas SettingPreferences

Nama	Keterangan
setKampusPreferences ()	Digunakan untuk mengaktifkan atau menonaktifkan pengaturan
notifikasiListener ()	Digunakan untuk mengambil aksi dari pengguna terhadap pengaturan notifikasi

26. Kelas Nilai

Kelas Nilai merupakan kelas yang digunakan untuk mengambil data semester yang telah ditempuh oleh pengguna. Berikut ini atribut dan metode yang dimiliki oleh kelas Nilai yang dapat dilihat pada Gambar III.99.



Gambar III.99 Kelas Nilai

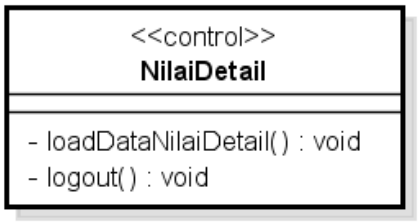
Adapun penjelasan dari setiap metode dari kelas Nilai dapat dijelaskan pada Tabel III.86.

Tabel III.86 Deskripsi Metode Kelas Nilai

Nama	Keterangan
loadDataNilai()	Digunakan untuk mengambil data semester yang telah ditempuh oleh pengguna
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*

27. Kelas NilaiDetail

Kelas NilaiDetail merupakan kelas yang digunakan untuk mengambil data detail nilai berdasarkan semester yang dipilih oleh pengguna. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiDetail yang dapat dilihat pada Gambar III.100.



Gambar III.100 Kelas NilaiDetail

Adapun penjelasan dari setiap metode dari kelas NilaiDetail dapat dijelaskan pada Tabel III.87.

Tabel III.87 Deskripsi Metode Kelas NilaiDetail

Nama	Keterangan
loadDataNilaiDetail()	Digunakan untuk mengambil data nilai dari semester yang dipilih oleh pengguna
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*

28. Kelas NilaiDetailInformasi

Kelas NilaiDetailInformasi merupakan kelas yang digunakan untuk mengambil data item nilai. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiDetailInformasi yang dapat dilihat pada Gambar III.101.



Gambar III.101 Kelas NilaiDetailInformasi

Adapun penjelasan dari setiap metode dari kelas NilaiDetailInformasi dapat dijelaskan pada Tabel III.88.

Tabel III.88 Deskripsi Metode Kelas NilaiDetailInformasi

Nama	Keterangan
loadNilaiDetailItem()	Digunakan untuk mengambil data detail item nilai dari nilai detail yang dipilih oleh pengguna

29. Kelas NilaiKeterangan

Kelas NilaiKeterangan merupakan kelas yang digunakan untuk mengambil data lengkap mahasiswa yang *login* ke aplikasi *mobile* Kampus *Online*. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiKeterangan yang dapat dilihat pada Gambar III.102.



Gambar III.102 Kelas NilaiKeterangan

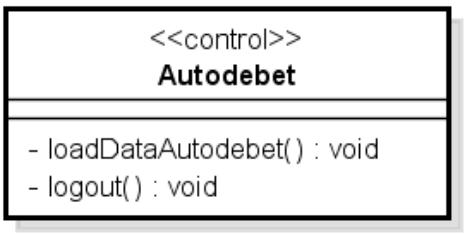
Adapun penjelasan dari setiap metode dari kelas NilaiKeterangan dapat dijelaskan pada Tabel III.89.

Tabel III.89 Deskripsi Metode Kelas NilaiKeterangan

Nama	Keterangan
setKeterangan()	Digunakan untuk menampilkan data lengkap mahasiswa yang *login* ke aplikasi *mobile* Kampus *Online*

30. Kelas Autodebet

Kelas Autodebet merupakan kelas yang digunakan untuk mengambil data tahun akademik yang telah ditempuh oleh pengguna. Berikut ini atribut dan metode yang dimiliki oleh kelas Autodebet yang dapat dilihat pada Gambar III.103.



Gambar III.103 Kelas Autodebet

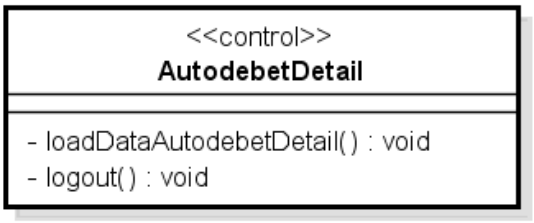
Adapun penjelasan dari setiap metode dari kelas Autodebet dapat dijelaskan pada Tabel III.90.

Tabel III.90 Deskripsi Metode Kelas Autodebet

Nama	Keterangan
loadDataAutodebet()	Digunakan untuk mengambil data tahun akademik yang telah ditempuh oleh pengguna
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*

31. Kelas AutodebetDetail

Kelas AutodebetDetail merupakan kelas yang digunakan untuk mengambil data detail autodebet berdasarkan tahun akademik yang dipilih oleh pengguna. Berikut ini atribut dan metode yang dimiliki oleh kelas AutodebetDetail yang dapat dilihat pada Gambar III.104.



Gambar III.104 Kelas AutodebetDetail

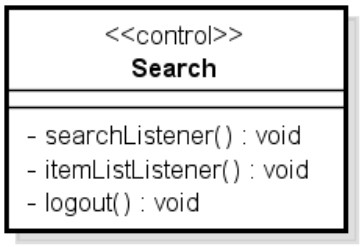
Adapun penjelasan dari setiap metode dari kelas AutodebetDetail dapat dijelaskan pada Tabel III.91.

Tabel III.91 Deskripsi Metode Kelas AutodebetDetail

Nama	Keterangan
loadDataAutodebetDetail()	Digunakan untuk mengambil data autodebet dari tahun akademik yang dipilih oleh pengguna
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*

32. Kelas Search

Kelas Search merupakan kelas yang digunakan untuk mencari data pengguna Kampus *Online*. Berikut ini atribut dan metode yang dimiliki oleh kelas Search yang dapat dilihat pada Gambar III.105.



Gambar III.105 Kelas Search

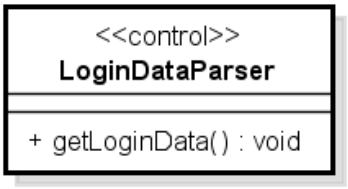
Adapun penjelasan dari setiap metode dari kelas Search dijelaskan pada Tabel III.92.

Tabel III.92 Deskripsi Metode Kelas Search

Nama	Keterangan
searchListener()	Digunakan untuk melakukan pencarian data pengguna Kampus *Online*
itemListListener()	Digunakan untuk melihat profil pengguna hasil pencarian
logout()	Digunakan oleh sistem untuk keluar secara otomatis dari aplikasi *mobile* Kampus *Online*

33. Kelas LoginDataParser

Kelas LoginDataParser merupakan kelas yang digunakan untuk memetakkan respon data *login* dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas LoginDataParser yang dapat dilihat pada Gambar III.106.



Gambar III.106 Kelas LoginDataParser

Adapun penjelasan dari setiap metode dari kelas LoginDataParser dijelaskan pada Tabel III.93.

Tabel III.93 Deskripsi Metode Kelas LoginDataParser

Nama	Keterangan
getLoginData()	Digunakan untuk memetakkan respon data *login* dari *server*

34. Kelas NotifikasiCountParser

Kelas NotifikasiCountParser merupakan kelas yang digunakan untuk memetakkan respon jumlah notifikasi dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas NotifikasiCountParser yang dapat dilihat pada Gambar III.107.



Gambar III.107 Kelas NotifikasiCountParser

Adapun penjelasan dari setiap metode dari kelas NotifikasiCountParser dijelaskan pada Tabel III.94.

Tabel III.94 Deskripsi Metode Kelas NotifikasiCountParser

Nama	Keterangan
getNotifikasiCount()	Digunakan untuk memetakkan respon jumlah notifikasi dari *server*

35. Kelas NotifikasiDataParser

Kelas NotifikasiDataParser merupakan kelas yang digunakan untuk memetakkan respon data notifikasi dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas NotifikasiDataParser yang dapat dilihat pada Gambar III.108.



Gambar III.108 Kelas NotifikasiDataParser

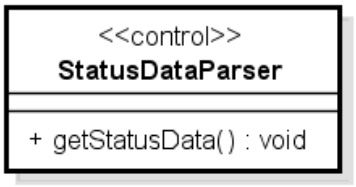
Adapun penjelasan dari setiap metode dari kelas NotifikasiDataParser dijelaskan pada Tabel III.95.

Tabel III.95 Deskripsi Metode Kelas NotifikasiDataParser

Nama	Keterangan
getNotifikasiData()	Digunakan untuk memetakkan respon data notifikasi dari *server*

36. Kelas StatusDataParser

Kelas StatusDataParser merupakan kelas yang digunakan untuk memetakkan respon data status dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusDataParser yang dapat dilihat pada Gambar III.109.



Gambar III.109 Kelas StatusDataParser

Adapun penjelasan dari setiap metode dari kelas StatusDataParser dijelaskan pada Tabel III.96.

Tabel III.96 Deskripsi Metode Kelas StatusDataParser

Nama	Keterangan
getStatusData()	Digunakan untuk memetakkan respon data status dari *server*

37. Kelas StatusKomentarDataParser

Kelas StatusKomentarDataParser merupakan kelas yang digunakan untuk memetakkan respon data status dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusKomentarDataParser yang dapat dilihat pada Gambar III.110.



Gambar III.110 Kelas StatusKomentarDataParser

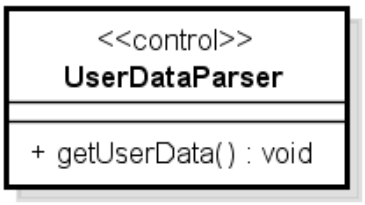
Adapun penjelasan dari setiap metode dari kelas StatusKomentarDataParser dijelaskan pada Tabel III.97.

Tabel III.97 Deskripsi Metode Kelas StatusKomentarDataParser

Nama	Keterangan
getStatusKomentarData()	Digunakan untuk memetakkan respon data komentar status dari *server*

38. Kelas UserDataParser

Kelas UserDataParser merupakan kelas yang digunakan untuk memetakkan respon data pengguna dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas UserDataParser yang dapat dilihat pada Gambar III.111.



Gambar III.111 Kelas UserDataParser

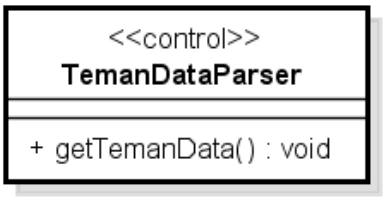
Adapun penjelasan dari setiap metode dari kelas UserDataParser dijelaskan pada Tabel III.98.

Tabel III.98 Deskripsi Metode Kelas UserDataParser

Nama	Keterangan
getUserData()	Digunakan untuk memetakkan respon data pengguna dari *server*

39. Kelas TemanDataParser

Kelas TemanDataParser merupakan kelas yang digunakan untuk memetakkan respon data pengguna dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas TemanDataParser yang dapat dilihat pada Gambar III.112.



Gambar III.112 Kelas UserDataParser

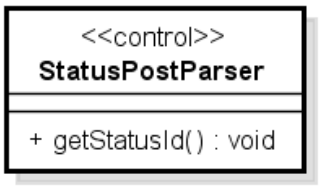
Adapun penjelasan dari setiap metode dari kelas TemanDataParser dijelaskan pada Tabel III.99.

Tabel III.99 Deskripsi Metode Kelas UserDataParser

Nama	Keterangan
getTemanData()	Digunakan untuk memetakkan respon data teman dari *server*

40. Kelas StatusPostParser

Kelas StatusPostParser merupakan kelas yang digunakan untuk memetakkan respon data dari *server* hasil *update status*. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusPostParser yang dapat dilihat pada Gambar III.113.



Gambar III.113 Kelas StatusPostParser

Adapun penjelasan dari setiap metode dari kelas StatusPostParser dijelaskan pada Tabel III.100.

Tabel III.100 Deskripsi Metode Kelas UserDataParser

Nama	Keterangan
getStatusId()	Digunakan untuk memetakkan respon data teman dari *server*

41. Kelas NilaiDataParser

Kelas NilaiDataParser merupakan kelas yang digunakan untuk memetakkan respon data nilai dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiDataParser yang dapat dilihat pada Gambar III.114.



Gambar III.114 Kelas NilaiDataParser

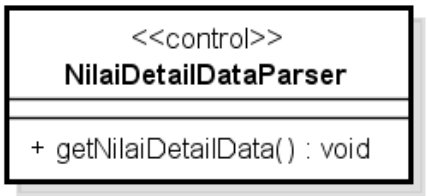
Adapun penjelasan dari setiap metode dari kelas NilaiDataParser dijelaskan pada Tabel III.101.

Tabel III.101 Deskripsi Metode Kelas NilaiDataParser

Nama	Keterangan
getNilaiData()	Digunakan untuk memetakkan respon data nilai dari *server*

42. Kelas NilaiDetailDataParser

Kelas NilaiDetailDataParser merupakan kelas yang digunakan untuk memetakkan respon data detail nilai dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiDetailDataParser yang dapat dilihat pada Gambar III.115.



Gambar III.115 Kelas NilaiDetailDataParser

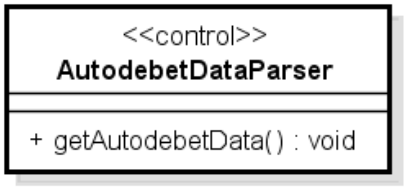
Adapun penjelasan dari setiap metode dari kelas NilaiDataParser dijelaskan pada Tabel III.102.

Tabel III.102 Deskripsi Metode Kelas NilaiDetailDataParser

Nama	Keterangan
getNilaiDetailData()	Digunakan untuk memetakkan respon data nilai detail dari *server*

43. Kelas AutodebetDataParser

Kelas AutodebetDataParser merupakan kelas yang digunakan untuk memetakkan respon data autodebet dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas AutodebetDataParser yang dapat dilihat pada Gambar III.116.



Gambar III.116 Kelas AutodebetDataParser

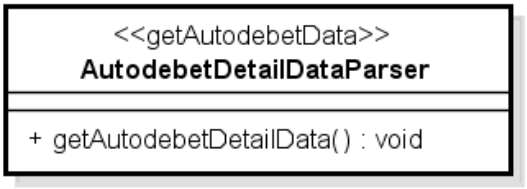
Adapun penjelasan dari setiap metode dari kelas AutodebetDataParser dijelaskan pada Tabel III.103.

Tabel III.103 Deskripsi Metode Kelas AutodebetDataParser

Nama	Keterangan
getAutodebetData()	Digunakan untuk memetakkan respon data autodebet dari *server*

44. Kelas AutodebetDetailDataParser

Kelas AutodebetDetailDataParser merupakan kelas yang digunakan untuk memetakkan respon data detail autodebet dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas AutodebetDetailDataParser yang dapat dilihat pada Gambar III.117.



Gambar III.117 Kelas AutodebetDetailDataParser

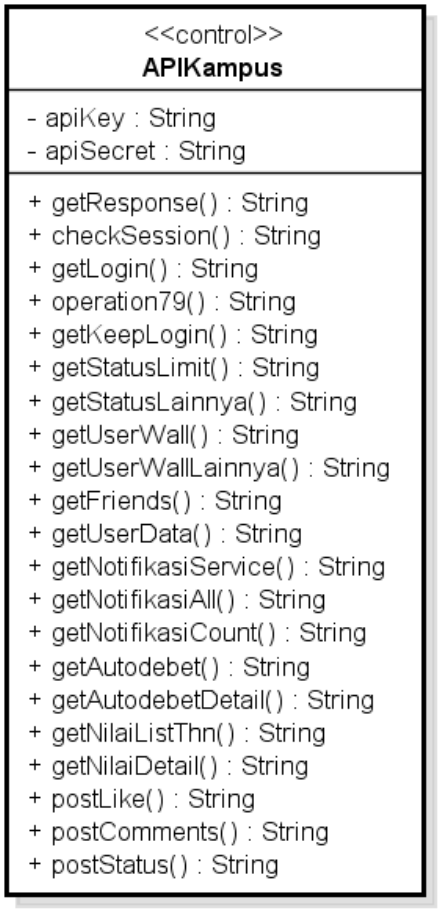
Adapun penjelasan dari setiap metode dari kelas AutodebetDetailDataParser dijelaskan pada Tabel III.104.

Tabel III.104 Deskripsi Metode Kelas AutodebetDetailDataParser

Nama	Keterangan
getAutodebetDetailData()	Digunakan untuk memetakkan respon data autodebet detail dari *server*

45. Kelas APIKampus

Kelas APIKampus merupakan kelas *utility* yang berisi metode-metode dalam API Kampus *Online* dan berfungsi dalam pengambilan data dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas APIKampus yang dapat dilihat pada Gambar III.118.



Gambar III.118 Kelas APIKampus

Adapun penjelasan dari setiap atribut dari kelas APIKampus dapat dijelaskan pada Tabel III.105.

Tabel III.105 Deskripsi Atribut Kelas APIKampus

Nama	Tipe	Keterangan
apiKey	String	Digunakan sebagai kunci pengenal aplikasi yang dibangun dalam melakukan permintaan data terhadap *server*
apiSecret	String	Digunakan sebagai *salt* dalam enkripsi metode-metode API Kampus *Online*

Sementara itu penjelasan dari setiap metode dari kelas APIKampus dapat dijelaskan pada Tabel III.106 dan Tabel III.107.

Tabel III.106 Deskripsi Metode Kelas APIKampus

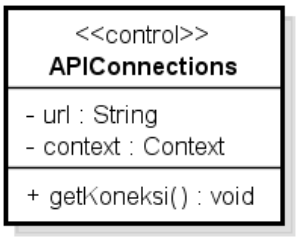
Nama	Keterangan
getRespone()	Digunakan untuk memetakkan respon data dari *server*
checkSession()	Digunakan untuk memetakkan respon data *session* dari *server*
getLogin()	Digunakan untuk memetakkan respon data *login* dari *server*
getKeepLogin()	Digunakan untuk memetakkan respon data ingat *login* dari *server*
getStatusLimit()	Digunakan untuk memetakkan respon data status dari *server*
getStatusLimitLainnya()	Digunakan untuk memetakkan respon data status lainnya dari *server*
getStatusComments()	Digunakan untuk memetakkan respon data komentar dari *server*
getUserWall()	Digunakan untuk memetakkan respon data status satu pengguna dari *server*
getUserWallLainnya()	Digunakan untuk memetakkan respon data status satu pengguna lainnya dari *server*
getFriends()	Digunakan untuk memetakkan respon data teman dari *server*
getUserData()	Digunakan untuk memetakkan respon data pengguna dari *server*
getNotifikasiService()	Digunakan untuk memetakkan respon data notifikasi yang belum dibaca dari *server*
getNotifikasiAll()	Digunakan untuk memetakkan respon data semua notifikasi dari *server*
getNotifikasiCount()	Digunakan untuk memetakkan respon data jumlah notifikasi dari *server*
getAutodebet()	Digunakan untuk memetakkan respon data autodebet dari *server*
getAutodebetDetail()	Digunakan untuk memetakkan respon data detail autodebet dari *server*
getNilaiListThn()	Digunakan untuk memetakkan respon data nilai dari *server*

Tabel III.107 Deskripsi Metode Kelas APIKampus (Lanjutan)

Nama	Keterangan
getNilaiDetail()	Digunakan untuk memetakkan respon data detail nilai dari *server*
postLike()	Digunakan untuk menambahkan data *rating* status
postComments()	Digunakan untuk menambahkan data komentar terhadap status
postStatus()	Digunakan untuk menambahkan data status

46. Kelas APIConnections

Kelas APIConnections merupakan kelas *utility* yang berfungsi dalam pengambilan data dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas APIConnections yang dapat dilihat pada Gambar III.119.



Gambar III.119 Kelas APIConnections

Adapun penjelasan dari setiap atribut dari kelas APIConnections dapat dijelaskan pada Tabel III.108.

Tabel III.108 Deskripsi Atribut Kelas APIConnections

Nama	Tipe	Keterangan
url	String	Digunakan sebagai kunci pengenal aplikasi yang dibangun dalam melakukan permintaan data terhadap *server*
apiSecret	String	Digunakan sebagai *salt* dalam enkripsi metode-metode API Kampus *Online*

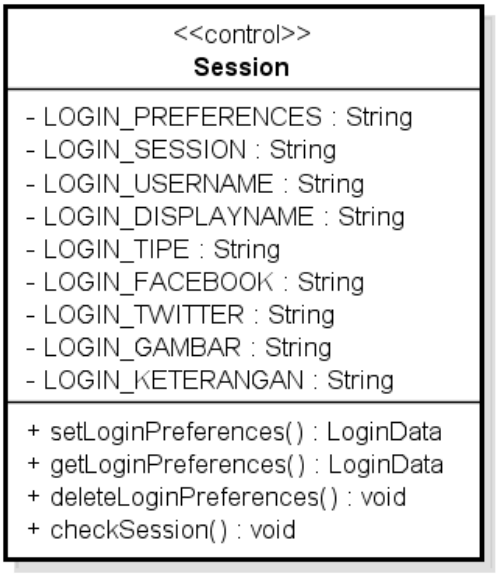
Sementara itu penjelasan dari setiap metode dari kelas APIConnections dapat dijelaskan pada Tabel III.109.

Tabel III.109 Deskripsi Metode Kelas APIConnections

Nama	Keterangan
getKoneksi ()	Digunakan untuk mendapatkan respon data dari *server*

47. Kelas *Session*

Kelas *Session* merupakan kelas *utility* yang berfungsi untuk menyimpan data *login* dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas *Session* yang dapat dilihat pada Gambar III.120.



Gambar III.120 Kelas *Session*

Adapun penjelasan dari setiap atribut dari kelas *Session* dapat dijelaskan pada Tabel III.110.

Tabel III.110 Deskripsi Atribut Kelas *Session*

Nama	Tipe	Keterangan
LOGIN_PREFERENCES	String	Digunakan sebagai kunci dalam menyimpan dan mengambil data *preferences*
LOGIN_SESSION	String	Digunakan untuk menyimpan data *session* dari *server*
LOGIN_USERNAME	String	Digunakan untuk menyimpan data *session* dari *server*
LOGIN_DISPLAYNAME	String	Digunakan untuk menyimpan data *session* dari *server*
LOGIN_TIPE	String	Digunakan untuk menyimpan data *session* dari *server*
LOGIN_FACEBOOK	String	Digunakan untuk menyimpan data *session* dari *server*
LOGIN_TWITTER	String	Digunakan untuk menyimpan data *session* dari *server*
LOGIN_GAMBAR	String	Digunakan untuk menyimpan data *session* dari *server*
LOGIN_KETERANGAN	String	Digunakan untuk menyimpan data *session* dari *server*

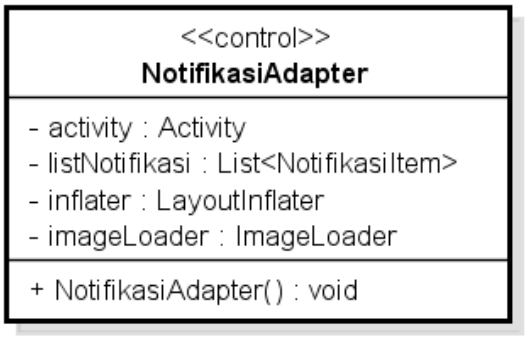
Sementara itu penjelasan dari setiap metode dari kelas *Session* dapat dijelaskan pada Tabel III.111.

Tabel III.111 Deskripsi Metode Kelas *Session*

Nama	Keterangan
setLoginPreferences ()	Digunakan untuk menyimpan data *login* dari *server*
getLoginPreferences()	Digunakan untuk mengambil data *login* yang telah disimpan sebelumnya
deleteLoginPreferences()	Digunakan untuk menghapus data *login* yang telah disimpan sebelumnya
checkSession()	Digunakan untuk mengecek apakah ada data *login* atau tidak

48. Kelas NotifikasiAdapter

Kelas NotifkasiAdapter merupakan kelas yang berfungsi untuk memasangkan data notifikasi yang telah dipetakan oleh kelas *parser* ke dalam ListView Android. Berikut ini atribut dan metode yang dimiliki oleh kelas NotifkasiAdapter yang dapat dilihat pada Gambar III.121.



Gambar III.121 Kelas NotifkasiAdapter

Adapun penjelasan dari setiap atribut dari kelas NotifkasiAdapter dapat dijelaskan pada Tabel III.112.

Tabel III.112 Deskripsi Atribut Kelas NotifkasiAdapter

Nama	Tipe	Keterangan
activity	Activity	Merupakan *references* kelas activity yang menggunakan kelas NotifikasiAdapter
listNotifikasi	List<NotifikasiItem>	Berisi daftar notifikasi yang akan ditampilkan
inflater	LayoutInflater	Berfungsi untuk menginstansiasi tata letak *file* XML ke dalam objek notifikasi
imageLoader	ImageLoader	Digunakan untuk mengambil gambar tipe notifikasi Kampus *Online* dari *server*

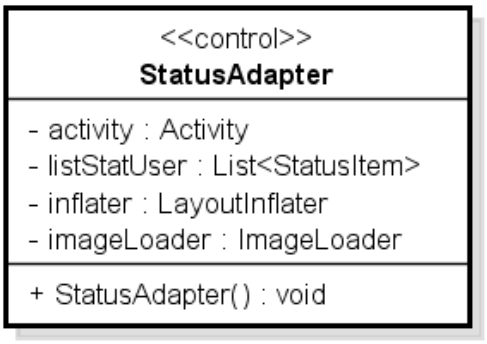
Sementara itu penjelasan dari setiap metode dari kelas NotifkasiAdapter dapat dijelaskan pada Tabel III.113.

Tabel III.113 Deskripsi Metode Kelas NotifkasiAdapter

Nama	Keterangan
NotifikasiAdapter()	Digunakan untuk memasukkan data notifikasi ke dalam kelas NotifikasiAdapter

49. Kelas StatusAdapter

Kelas StatusAdapter merupakan kelas yang berfungsi untuk memasangkan data status yang telah dipetakan oleh kelas *parser* ke dalam ListView status. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusAdapter yang dapat dilihat pada Gambar III.122.



Gambar III.122 Kelas StatusAdapter

Adapun penjelasan dari setiap atribut dari kelas StatusAdapter dapat dijelaskan pada Tabel III.114.

Tabel III.114 Deskripsi Atribut Kelas StatusAdapter

Nama	Tipe	Keterangan
activity	Activity	Merupakan *references* kelas activity yang menggunakan kelas StatusAdapter
listStatUses	List<StatusItem>	Berisi daftar status yang akan ditampilkan
inflater	LayoutInflater	Berfungsi untuk menginstansiasi tata letak *file* XML ke dalam objek status
imageLoader	ImageLoader	Digunakan untuk mengambil foto pengguna Kampus *Online* dari *server*

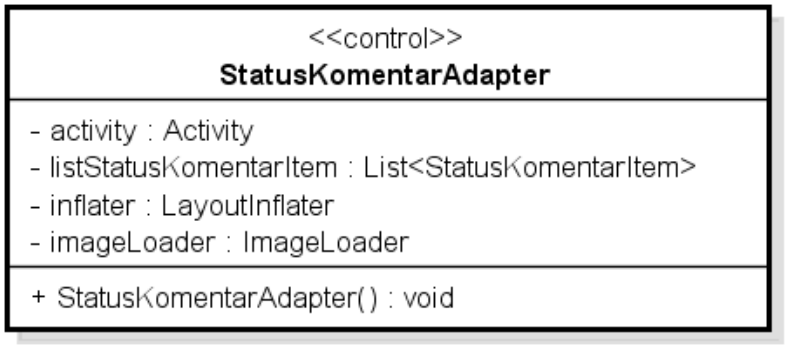
Sementara itu penjelasan dari setiap metode dari kelas StatusAdapter dapat dijelaskan pada Tabel III.115.

Tabel III.115 Deskripsi Metode Kelas StatusAdapter

Nama	Keterangan
StatusAdapter()	Digunakan untuk memasukkan data status ke dalam kelas StatusAdapter

50. Kelas StatusKomentarAdapter

Kelas StatusKomentarAdapter merupakan kelas yang berfungsi untuk memasangkan data komentar status yang telah dipetakan oleh kelas *parser* ke dalam ListView komentar. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusKomentarAdapter yang dapat dilihat pada Gambar III.123.



Gambar III.123 Kelas StatusKomentarAdapter

Adapun penjelasan dari setiap atribut dari kelas StatusKomentarAdapter dapat dijelaskan pada Tabel III.116.

Tabel III.116 Deskripsi Atribut Kelas StatusKomentarAdapter

Nama	Tipe	Keterangan
activity	Activity	Merupakan *references* kelas activity yang menggunakan kelas StatusKomentarAdapter
listStatusKometarItem	List<StatusKomentarItem>	Berisi daftar komentar status yang akan ditampilkan
inflater	LayoutInflater	Berfungsi untuk menginstansiasi tata letak *file* XML ke dalam objek komentar
imageLoader	ImageLoader	Digunakan untuk mengambil foto pengguna Kampus *Online* dari *server*

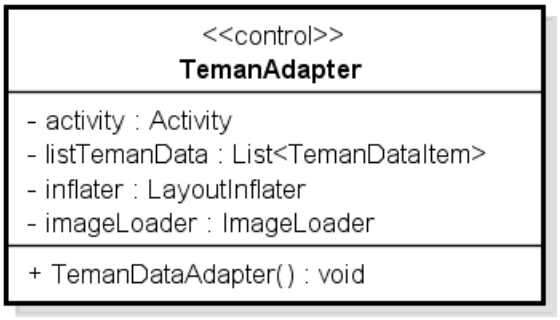
Sementara itu penjelasan dari setiap metode dari kelas StatusKomentarAdapter dapat dijelaskan pada Tabel III.117.

Tabel III.117 Deskripsi Metode Kelas StatusKomentarAdapter

Nama	Keterangan
StatusKomentarAdapter()	Digunakan untuk memasukkan data komentar ke dalam kelas StatusKomentarAdapter

51. Kelas TemanAdapter

Kelas TemanAdapter merupakan kelas yang berfungsi untuk memasangkan data teman yang telah dipetakan oleh kelas *parser* ke dalam ListView teman. Berikut ini atribut dan metode yang dimiliki oleh kelas TemanAdapter yang dapat dilihat pada Gambar III.124.



Gambar III.124 Kelas TemanAdapter

Adapun penjelasan dari setiap atribut dari kelas TemanAdapter dapat dijelaskan pada Tabel III.118.

Tabel III.118 Deskripsi Atribut Kelas TemanAdapter

Nama	Tipe	Keterangan
activity	Activity	Merupakan *references* kelas activity yang menggunakan kelas TemanAdapter
listTemanData	List<TemanDataItem>	Berisi daftar teman yang akan ditampilkan
inflater	LayoutInflater	Berfungsi untuk menginstansiasi tata letak *file* XML ke dalam objek teman
imageLoader	ImageLoader	Digunakan untuk mengambil foto teman Kampus *Online* dari *server*

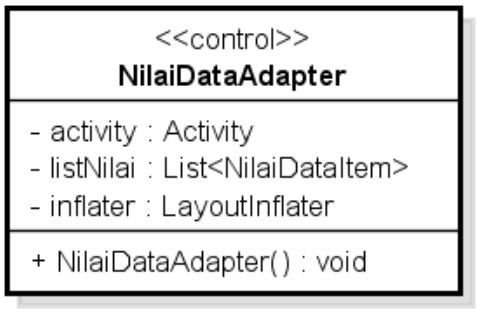
Sementara itu penjelasan dari setiap metode dari kelas TemanAdapter dapat dijelaskan pada Tabel III.119.

Tabel III.119 Deskripsi Metode Kelas TemanAdapter

Nama	Keterangan
TemanDataAdapter()	Digunakan untuk memasukkan data teman ke dalam kelas TemanAdapter

52. Kelas NilaiDataAdapter

Kelas NilaiDataAdapter merupakan kelas yang berfungsi untuk memasangkan data nilai yang telah dipetakan oleh kelas *parser* ke dalam ListView nilai. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiDataAdapter yang dapat dilihat pada Gambar III.125.



Gambar III.125 Kelas NilaiDataAdapter

Adapun penjelasan dari setiap atribut dari kelas NilaiDataAdapter dapat dijelaskan pada Tabel III.120.

Tabel III.120 Deskripsi Atribut Kelas NilaiDataAdapter

Nama	Tipe	Keterangan
activity	Activity	Merupakan *references* kelas activity yang menggunakan kelas NilaiDataAdapter
listNilai	List<NilaiDataItem>	Berisi daftar semester yang telah ditempuh oleh mahasiswa
inflater	LayoutInflater	Berfungsi untuk menginstansiasi tata letak *file* XML ke dalam objek nilai

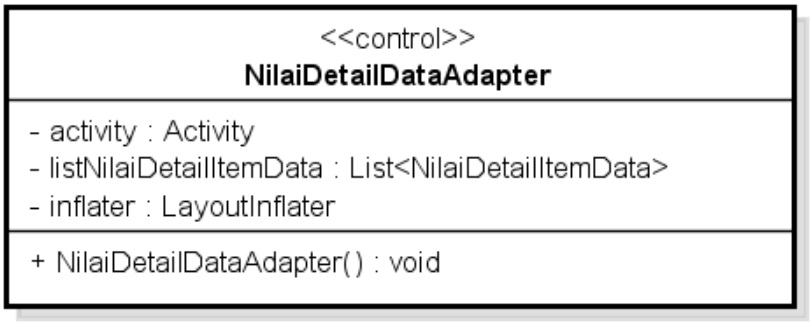
Sementara itu penjelasan dari setiap metode dari kelas NilaiDataAdapter dapat dijelaskan pada Tabel III.121.

Tabel III.121 Deskripsi Metode Kelas NilaiDataAdapter

Nama	Keterangan
NilaiDataAdapter()	Digunakan untuk memasukkan data semester yang telah ditempuh ke dalam kelas NilaiDataAdapter

53. Kelas NilaiDetailDataAdapter

Kelas NilaiDetailDataAdapter merupakan kelas yang berfungsi untuk memasangkan data detail nilai yang telah dipetakan oleh kelas *parser* ke dalam ListView nilai item. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiDetailDataAdapter yang dapat dilihat pada Gambar III.126.



Gambar III.126 Kelas NilaiDetailDataAdapter

Adapun penjelasan dari setiap atribut dari kelas NilaiDetailDataAdapter dapat dijelaskan pada Tabel III.122.

Tabel III.122 Deskripsi Atribut Kelas NilaiDetailDataAdapter

Nama	Tipe	Keterangan
activity	Activity	Merupakan *references* kelas activity yang menggunakan kelas NilaiDetailDataAdapter
listNilaiDetailItemData	List< NilaiDetailItemData>	Berisi daftar semester yang telah ditempuh oleh mahasiswa
inflater	LayoutInflater	Berfungsi untuk menginstansiasi tata letak *file* XML ke dalam objek nilai

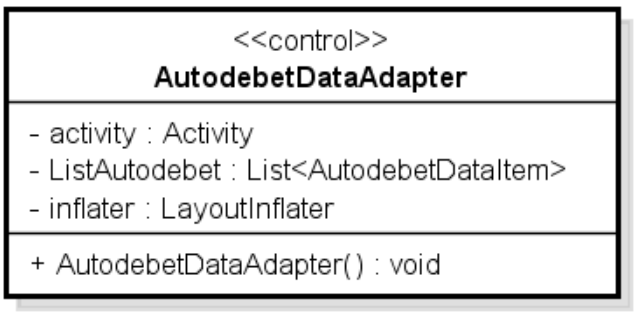
Sementara itu penjelasan dari setiap metode dari kelas NilaiDetailDataAdapter dapat dijelaskan pada Tabel III.123.

Tabel III.123 Deskripsi Metode Kelas NilaiDetailDataAdapter

Nama	Keterangan
NilaiDetailDataAdapter()	Digunakan untuk memasukkan data semester yang telah ditempuh ke dalam kelas NilaiDetailDataAdapter

54. Kelas AutodebetDataAdapter

Kelas AutodebetDataAdapter merupakan kelas yang berfungsi untuk memasangkan data autodebet yang telah dipetakan oleh kelas *parser* ke dalam ListView autodebet. Berikut ini atribut dan metode yang dimiliki oleh kelas AutodebetDataAdapter yang dapat dilihat pada Gambar III.127.



Gambar III.127 Kelas AutodebetDataAdapter

Adapun penjelasan dari setiap atribut dari kelas AutodebetDataAdapter dapat dijelaskan pada Tabel III.124.

Tabel III.124 Deskripsi Atribut Kelas AutodebetDataAdapter

Nama	Tipe	Keterangan
activity	Activity	Merupakan *references* kelas activity yang menggunakan kelas AutodebetDataAdapter
listAutodebet	List<AutodebetDataItem>	Berisi daftar nilai dari semester yang dipilih oleh mahasiswa
inflater	LayoutInflater	Berfungsi untuk menginstansiasi tata letak *file* XML ke dalam objek nilai detail

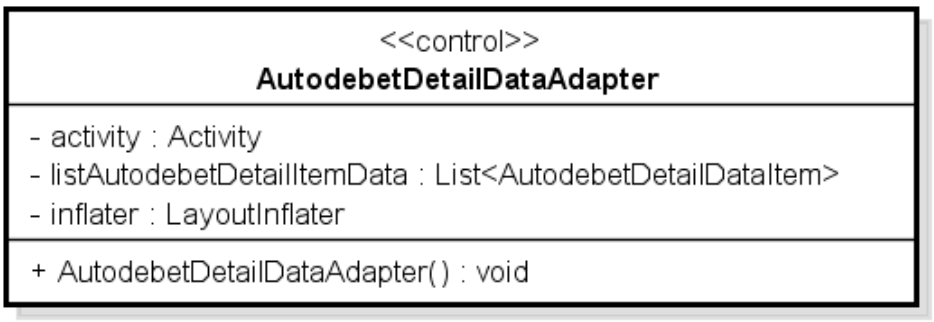
Sementara itu penjelasan dari setiap metode dari kelas AutodebetDataAdapter dapat dijelaskan pada Tabel III.125.

Tabel III.125 Deskripsi Metode Kelas AutodebetDataAdapter

Nama	Keterangan
AutodebetDataAdapter()	Digunakan untuk memasukkan data autodebet ke dalam kelas AutodebetDataAdapter

55. Kelas AutodebetDetailDataAdapter

Kelas AutodebetDetailDataAdapter merupakan kelas yang berfungsi untuk memasangkan data detail autodebet yang telah dipetakan oleh kelas *parser* ke dalam ListView detail autodebet. Berikut ini atribut dan metode yang dimiliki oleh kelas AutodebetDetailDataAdapter yang dapat dilihat pada Gambar III.128.



Gambar III.128 Kelas AutodebetDetailDataAdapter

Adapun penjelasan dari setiap atribut dari kelas AutodebetDetailDataAdapter dapat dijelaskan pada Tabel III.126.

Tabel III.126 Deskripsi Atribut Kelas AutodebetDetailDataAdapter

Nama	Tipe	Keterangan
activity	Activity	Merupakan *references* kelas activity yang menggunakan kelas AutodebetDetailDataAdapter
listAutodebetDetailItemData	List	Berisi daftar nilai dari semester yang dipilih oleh mahasiswa
inflater	LayoutInflater	Berfungsi untuk menginstansiasi tata letak *file* XML ke dalam objek nilai detail

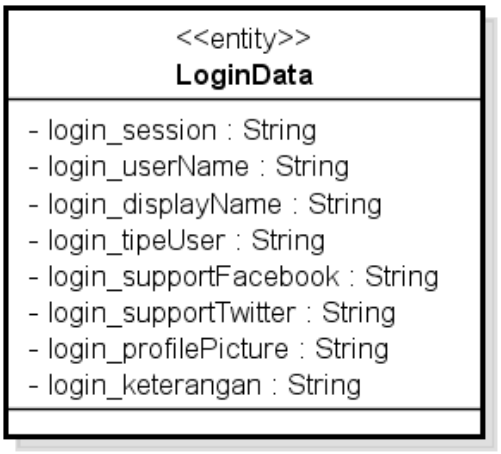
Sementara itu penjelasan dari setiap metode dari kelas AutodebetDetailDataAdapter dapat dijelaskan pada Tabel III.127.

Tabel III.127 Deskripsi Metode Kelas AutodebetDetailDataAdapter

Nama	Keterangan
AutodebetDetailDataAdapter()	Digunakan untuk memasukkan data detail autodebet ke dalam kelas AutodebetDetailDataAdapter

56. Kelas LoginData

Kelas LoginData merupakan kelas yang berfungsi untuk menampung data *login* yang didapatkan dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas LoginData yang dapat dilihat pada Gambar III.129.



Gambar III.129 Kelas LoginData

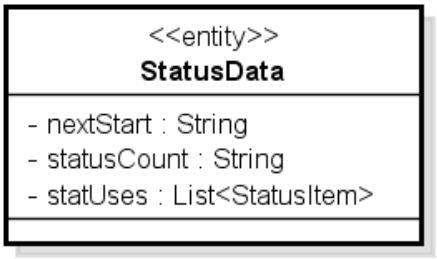
Adapun penjelasan dari setiap atribut dari kelas LoginData dapat dijelaskan pada Tabel III.128.

Tabel III.128 Deskripsi Atribut Kelas LoginData

Nama	Tipe	Keterangan
login_session	String	Berisi *session* pengguna yang *login*
login_userName	String	Berisi *username* dari pengguna yang *login*
login_displayName	String	Berisi nama dari pengguna yang *login*
login_tipeUser	String	Berisi tipe *user* dari pengguna yang *login*
login_supportFacebook	String	Berisi status *support facebook* atau tidak dari pengguna yang *login*
login_supportTwitter	String	Berisi status *support twitter* dari pengguna yang *login*
login_profilePicture	String	Berisi *link* foto profil dari pengguna yang *login*
login_keterangan	String	Berisi keterangan dari pengguna yang *login*

57. Kelas StatusData

Kelas StatusData merupakan kelas yang berfungsi untuk menampung data status yang didapatkan dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusData yang dapat dilihat pada Gambar III.130.



Gambar III.130 Kelas StatusData

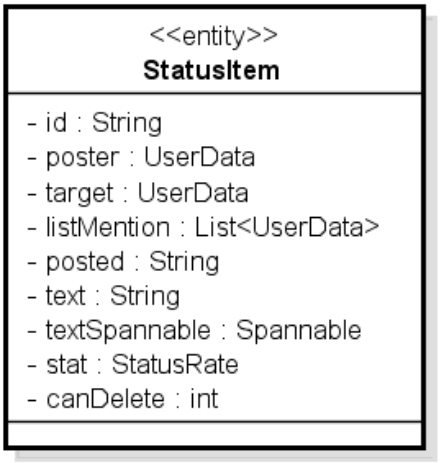
Adapun penjelasan dari setiap atribut dari kelas StatusData dapat dijelaskan pada Tabel III.129.

Tabel III.129 Deskripsi Atribut Kelas StatusData

Nama	Tipe	Keterangan
nextStart	String	Berisi posisi status pada halaman selanjutnya
statusCount	String	Berisi jumlah status yang didapatkan dari *server*
statUses	List	Berisi daftar status

58. Kelas StatusItem

Kelas StatusItem merupakan kelas yang berfungsi untuk menampung data item status yang didapatkan dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusItem yang dapat dilihat pada Gambar III.130.



Gambar III.131 Kelas StatusItem

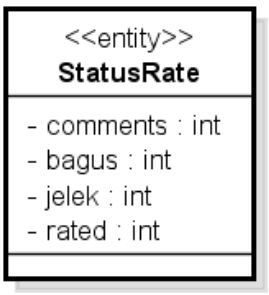
Adapun penjelasan dari setiap atribut dari kelas StatusItem dapat dijelaskan pada Tabel III.130.

Tabel III.130 Deskripsi Atribut Kelas StatusItem

Nama	Tipe	Keterangan
id	String	Berisi id status
poster	UserData	Berisi data pengguna pemilik status
target	UserData	Berisi data target jika status merupakan sapaan
listMention	List	Berisi pengguna-pengguna yang di-*mention* dalam status
posted	String	Berisi waktu status dibuat
text	String	Berisi teks status
textSpannable	Spannable	Berisi teks status berseta *emoticon*
stat	StatusRate	Berisi *rating* dari status
canDelete	Int	Berisi keterangan apakah status bisa dihapus atau tidak

59. Kelas StatusRate

Kelas StatusRate merupakan kelas yang berfungsi untuk menampung data *rating* status yang didapatkan dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusRate yang dapat dilihat pada Gambar III.132.



Gambar III.132 Kelas StatusRate

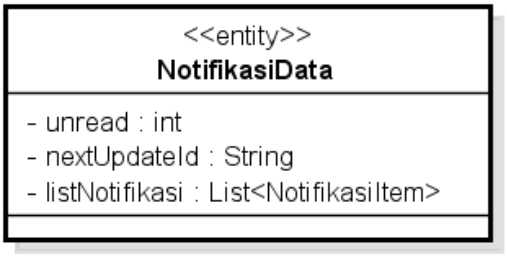
Adapun penjelasan dari setiap atribut dari kelas StatusRate dapat dijelaskan pada Tabel III.131.

Tabel III.131 Deskripsi Atribut Kelas StatusRate

Nama	Tipe	Keterangan
comments	Int	Berisi jumlah komentar dari status
bagus	Int	Berisi jumlah pengguna yang memberi *rating* bagus
jelek	Int	Berisi jumlah pengguna yang memberi *rating* jelek
rated	Int	Berisi jumlah pengguna yang memberi *rating*

60. Kelas NotifikasiData

Kelas NotifikasiData merupakan kelas yang berfungsi untuk menampung data notifikasi yang didapatkan dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas NotifikasiData yang dapat dilihat pada Gambar III.133.



Gambar III.133 Kelas NotifikasiData

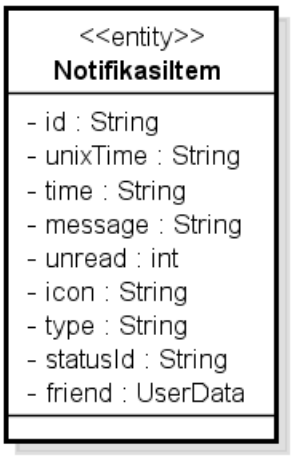
Adapun penjelasan dari setiap atribut dari kelas NotifikasiData dapat dijelaskan pada Tabel III.132.

Tabel III.132 Deskripsi Atribut Kelas NotifikasiData

Nama	Tipe	Keterangan
unread	Int	Berisi jumlah notifikasi yang belum dibaca
nextUpdateId	String	Berisi posisi notifikasi pada halaman berikutnya
listNotifikasi	List	Berisi data notifikasi

61. Kelas NotifikasiItem

Kelas NotifikasiItem merupakan kelas yang berfungsi untuk menampung data item notifikasi yang didapatkan dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas NotifikasiItem yang dapat dilihat pada Gambar III.134.



Gambar III.134 Kelas NotifikasiItem

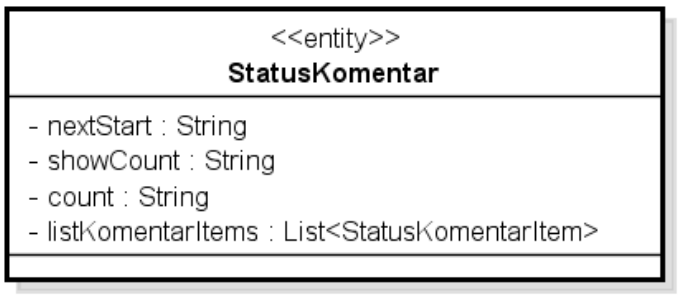
Adapun penjelasan dari setiap atribut dari kelas NotifikasiItem dapat dijelaskan pada Tabel III.133.

Tabel III.133 Deskripsi Atribut Kelas NotifikasiItem

Nama	Tipe	Keterangan
id	String	Berisi id notifikasi
unixTime	String	Berisi waktu dalam format unix
time	String	Berisi waktu dalam string
message	String	Berisi pesan notifikasi
unread	int	Berisi jumlah notifikasi yang belum dibaca
icon	String	Berisi *link icon* notifikasi
type	String	Berisi tipe dari notifikasi
statusId	String	Berisi id status dari notifikasi jika notifikasi bertipe status
friend	UserData	Berisi data pengguna jika notifikasi bertipe *friend*

62. Kelas StatusKomentar

Kelas StatusKomentar merupakan kelas yang berfungsi untuk menampung data komentar dari status yang didapatkan dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusKomentar yang dapat dilihat pada Gambar III.135.



Gambar III.135 Kelas StatusKomentar

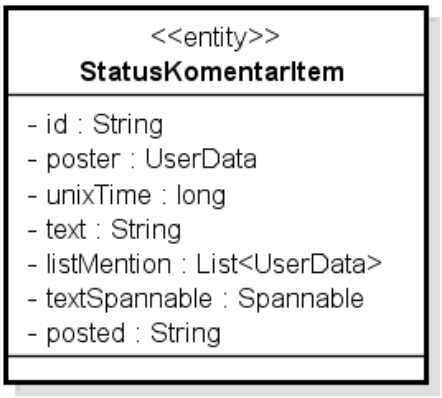
Adapun penjelasan dari setiap atribut dari kelas StatusKomentar dapat dijelaskan pada Tabel III.134.

Tabel III.134 Deskripsi Atribut Kelas StatusKomentar

Nama	Tipe	Keterangan
nextStart	String	Berisi posisi komentar pada halaman selanjutnya
showCount	String	Berisi jumlah komentar yang ditampilkan
count	String	Berisi seluruh jumlah komentar dari status
statusKomentarItems	List	Berisi data komentar yang diambil dari server

63. Kelas StatusKomentarItem

Kelas StatusKomentarItem merupakan kelas yang berfungsi untuk menampung data item komentar dari status yang didapatkan dari *server*. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusKomentarItem yang dapat dilihat pada Gambar III.136.



Gambar III.136 Kelas StatusKomentarItem

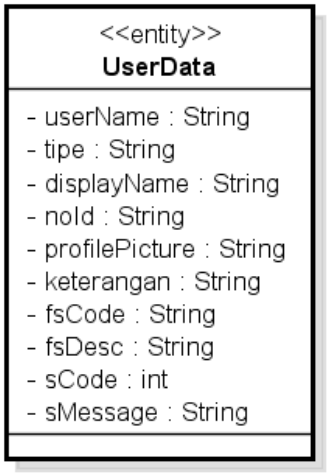
Adapun penjelasan dari setiap atribut dari kelas StatusKomentarItem dapat dijelaskan pada Tabel III.135.

Tabel III.135 Deskripsi Atribut Kelas StatusKomentarItem

Nama	Tipe	Keterangan
id	String	Berisi posisi komentar pada halaman selanjutnya
poster	UserData	Berisi jumlah komentar yang ditampilkan
unixTime	Long	Berisi seluruh jumlah komentar dari status
text	String	Isi komentar dari status
listMention	List	Daftar pengguna yang disinggung (*mention*) dalam komentar
textSpannable	Spannable	Isi komentar yang beserta gambar *emoticon*
posted	String	Waktu dalam format string

64. Kelas UserData

Kelas UserData merupakan kelas yang berfungsi untuk menampung data pengguna Kampus *Online*. Berikut ini atribut dan metode yang dimiliki oleh kelas UserData yang dapat dilihat pada Gambar III.137.



Gambar III.137 Kelas UserData

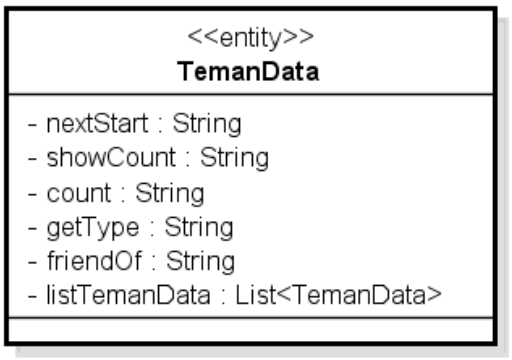
Adapun penjelasan dari setiap atribut dari kelas UserData dapat dijelaskan pada Tabel III.136.

Tabel III.136 Deskripsi Atribut Kelas UserData

Nama	Tipe	Keterangan
userName	String	Berisi username dari pengguna Kampus *Online*
tipe	String	Berisi tipe pengguna Kampus *Online*
displayName	String	Berisi nama pengguna Kampus *Online*
noId	String	Berisi id dari pengguna
profilePicture	String	Berisi *link* foto dari pengguna
keterangan	String	Berisi keterangan pengguna Kampus *Online*
fsCode	String	Berisi kode pertemanan dengan pengguna yang melakukan *login*
fsDesc	String	Berisi deskripsi pertemanan berdasarkan kode
sCode	Int	Berisi status permintaan data ke *server*
sMessage	String	Berisi pesan status permintaan

65. Kelas TemanData

Kelas TemanData kelas yang berfungsi untuk menampung data teman. Berikut ini atribut dan metode yang dimiliki oleh kelas TemanData yang dapat dilihat pada Gambar III.138.



Gambar III.138 Kelas TemanData

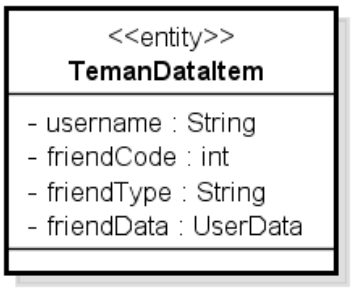
Adapun penjelasan dari setiap atribut dari kelas TemanData dapat dijelaskan pada Tabel III.137.

Tabel III.137 Deskripsi Atribut Kelas TemanData

Nama	Tipe	Keterangan
nextStart	String	Berisi posisi data teman pada halaman berikutnya
showCount	String	Berisi jumlah data teman yang ditampilkan
count	String	Berisi jumlah data teman yang diambil dari *server*
getType	String	Berisi tipe dari teman
friendOf	String	Berisi username dari teman yang ditampilkan
listTemanData	List	Berisi data teman yang diambil dari server

66. Kelas TemanDataItem

Kelas TemanDataItem merupakan kelas yang berfungsi untuk menampung data item teman. Berikut ini atribut dan metode yang dimiliki oleh kelas TemanDataItem yang dapat dilihat pada Gambar III.139.



Gambar III.139 Kelas TemanDataItem

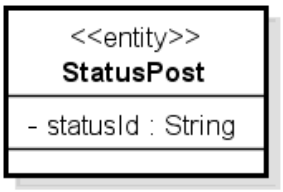
Adapun penjelasan dari setiap atribut dari kelas TemanDataItem dapat dijelaskan pada Tabel III.138.

Tabel III.138 Deskripsi Atribut Kelas TemanDataItem

Nama	Tipe	Keterangan
username	String	Berisi data *username* dari data item teman
friendCode	Int	Berisi kode pertemanan
friendType	String	Berisi deskripsi tipe pertemanan
friendData	UserData	Berisi data lengkap dari teman

67. Kelas StatusPost

Kelas StatusPost merupakan kelas yang berfungsi untuk menampung data hasil penambahan data status. Berikut ini atribut dan metode yang dimiliki oleh kelas StatusPost yang dapat dilihat pada Gambar III.140.



Gambar III.140 Kelas StatusPost

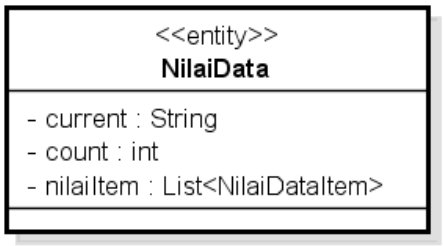
Adapun penjelasan dari setiap atribut dari kelas StatusPost dapat dijelaskan pada Tabel III.139.

Tabel III.139 Deskripsi Atribut Kelas StatusPost

Nama	Tipe	Keterangan
statusId	String	Berisi data id status dari status yang ditambahkan

68. Kelas NilaiData

Kelas NilaiData merupakan kelas yang berfungsi untuk menampung data semester yang telah ditempuh. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiData yang dapat dilihat pada Gambar III.141.



Gambar III.141 Kelas NilaiData

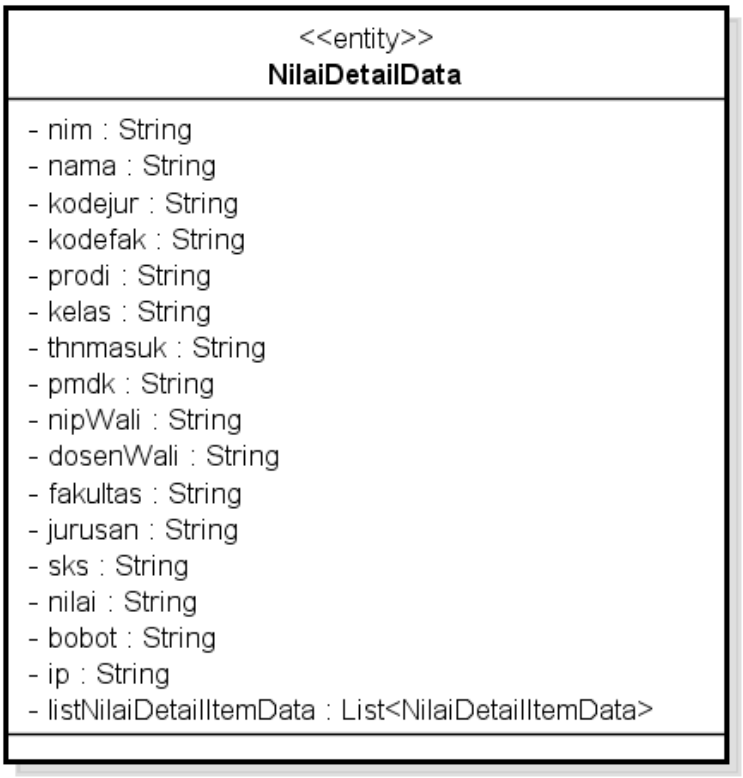
Adapun penjelasan dari setiap atribut dari kelas NilaiData dapat dijelaskan pada Tabel III.140.

Tabel III.140 Deskripsi Atribut Kelas NilaiData

Nama	Tipe	Keterangan
current	String	Berisi data semester yang sedang berjalan
count	Int	Berisi jumlah semester yang telah ditempuh oleh mahasiswa
nilaiItem	List	Berisi daftar semester yang telah ditempuh

69. Kelas NilaiDetailData

Kelas NilaiDetailData merupakan kelas yang berfungsi untuk menampung data detail nilai dari semester yang dipilih. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiDetailData yang dapat dilihat pada Gambar III.142.



Gambar III.142 Kelas NilaiDetailData

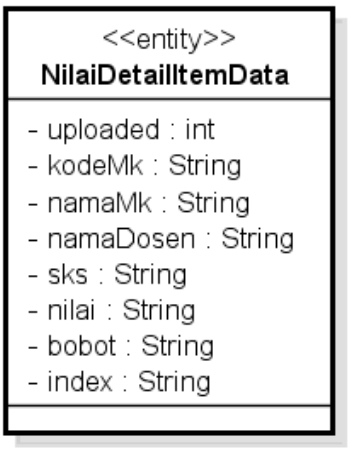
Adapun penjelasan dari setiap atribut dari kelas NilaiDetailData dapat dijelaskan pada Tabel III.141.

Tabel III.141 Deskripsi Atribut Kelas NilaiDetailData

Nama	Tipe	Keterangan
nim	String	Berisi NIM mahasiswa
nama	String	Berisi nama lengkap mahasiswa
kodejur	String	Berisi kode jurusan mahasiswa
kodefak	String	Berisi kode fakultas mahasiswa
prodi	String	Berisi program studi yang diambil
kelas	String	Berisi kelas dari mahasiswa
thnmasuk	String	Berisi data tahun masuk mahasiswa
pmdk	String	Berisi status apakah mahasiswa tersebut masuk melalui jalur PMDK atau bukan
nipWali	String	Berisi NIP dari dosen wali mahasiswa
dosenWali	String	Berisi nama dosen wali mahasiswa
fakultas	String	Berisi nama fakultas mahasiswa
jurusan	String	Berisi nama jurusan mahasiswa
sks	String	Berisi jumlah sks yang diambil pada semester yang dipilih
nilai	String	Berisi jumlah nilai yang didapat dari semua mata kuliah yang diambil dari semester yang dipilih
bobot	String	Berisi bobot nilai yang didapat hasil perkalian antara sks dengan nilai
ip	String	Berisi indeks prestasi yang diperoleh dari semester yang dipilih
listNilaiDetailItemData	List	Berisi daftar semester yang telah ditempuh oleh mahasiswa

70. Kelas NilaiDetailItemData

Kelas NilaiDetailItemData merupakan kelas yang berfungsi untuk menampung data item dari detail nilai berdasarkan semester yang dipilih. Berikut ini atribut dan metode yang dimiliki oleh kelas NilaiDetailItemData yang dapat dilihat pada Gambar III.143.



Gambar III.143 Kelas NilaiDetailItemData

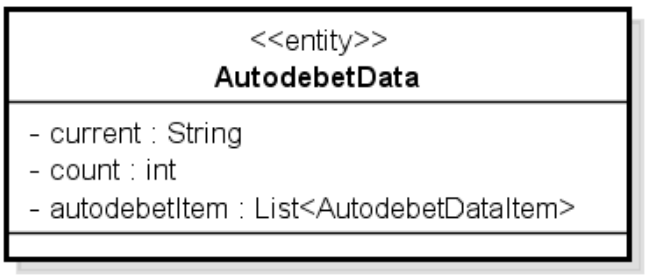
Adapun penjelasan dari setiap atribut dari kelas NilaiDetailItemData dapat dijelaskan pada Tabel III.142.

Tabel III.142 Deskripsi Atribut Kelas NilaiDetailItemData

Nama	Tipe	Keterangan
uploaded	int	Berisi status *upload* dari data nilai
kodeMk	String	Berisi kode mata kuliah
namaMk	String	Berisi nama mata kuliah
namaDosen	String	Berisi nama dosen pengajar mata kuliah yang bersangkutan
sks	String	Berisi jumlah sks mata kuliah
nilai	String	Berisi data nilai dari mata kuliah
bobot	String	Berisi bobot dari mata yang dipilih
index	String	Berisi indeks mata kuliah

71. Kelas AutodebetData

Kelas AutodebetData merupakan kelas yang berfungsi untuk menampung data tahun akademik yang telah ditempuh. Berikut ini atribut dan metode yang dimiliki oleh kelas AutodebetData yang dapat dilihat pada Gambar III.144.



Gambar III.144 Kelas AutodebetData

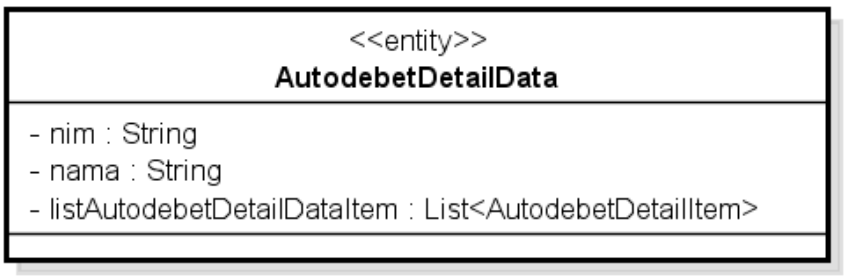
Adapun penjelasan dari setiap atribut dari kelas AutodebetData dapat dijelaskan pada Tabel III.143.

Tabel III.143 Deskripsi Atribut Kelas AutodebetData

Nama	Tipe	Keterangan
current	String	Berisi data tahun akademik yang sedang berjalan
count	Int	Berisi data tahun akademik yang telah ditempuh oleh mahasiswa yang *login*
autodebetItem	List	Berisi daftar tahun akademik yang telah ditempuh

72. Kelas AutodebetDetailData

Kelas AutodebetDetailData merupakan kelas yang berfungsi untuk menampung data detail autodebet berdasarkan tahun akademik yang dipilih. Berikut ini atribut dan metode yang dimiliki oleh kelas AutodebetDetailData yang dapat dilihat pada Gambar III.145.



Gambar III.145 Kelas AutodebetData

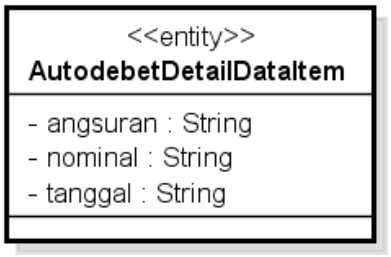
Adapun penjelasan dari setiap atribut dari kelas AutodebetDetailData dapat dijelaskan pada Tabel III.143.

Tabel III.144 Deskripsi Atribut Kelas AutodebetDetailData

Nama	Tipe	Keterangan
nim	String	Berisi NIM mahasiswa
nama	String	Berisi nama mahasiswa
listAutodebetDetailDataItem	List	Berisi daftar angsuran berdasarkan tahun akademik yang dipilih

73. Kelas AutodebetDetailDataItem

Kelas AutodebetDetailDataItem merupakan kelas yang berfungsi untuk menampung data detail autodebet berdasarkan tahun akademik yang dipilih. Berikut ini atribut dan metode yang dimiliki oleh kelas AutodebetDetailDataItem yang dapat dilihat pada Gambar III.146.



Gambar III.146 Kelas AutodebetDetailDataItem

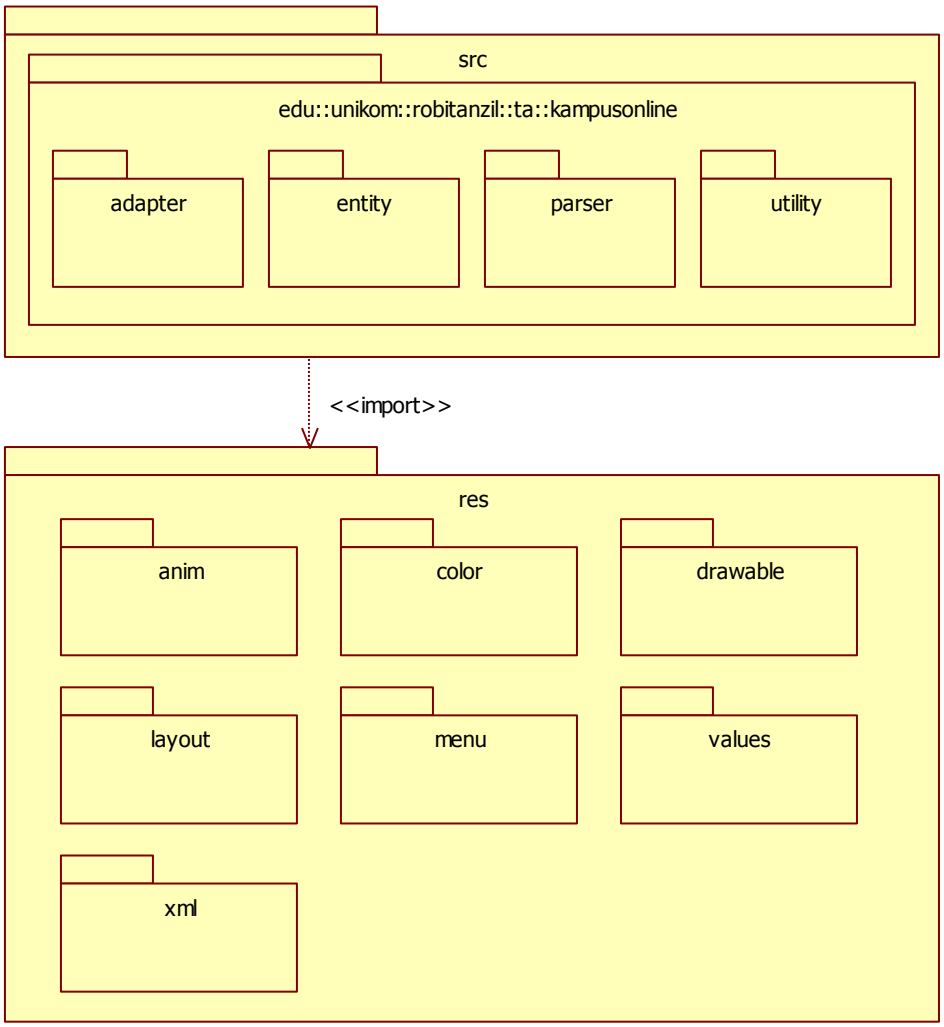
Adapun penjelasan dari setiap atribut dari kelas AutodebetDetailDataItem dapat dijelaskan pada Tabel III.145.

Tabel III.145 Deskripsi Atribut Kelas AutodebetDetailDataItem

Nama	Tipe	Keterangan
angsuran	String	Berisi keterangan angsuran dari autodebet yang telah dibayar
nominal	String	Berisi nominal biaya dari autodebet
tanggal	String	Berisi tanggal penarikan autodebet

3. 1. 9. 6. ***Package Diagram***

Package diagram menggambarkan pengorganisasian kelas-kelas yang terdapat pada aplikasi *mobile* Kampus *Online* dalam paket-paket tertentu. Adapun *package diagram* aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.147.



Gambar III.147 *Package Diagram* Aplikasi *Mobile* Kampus *Online*

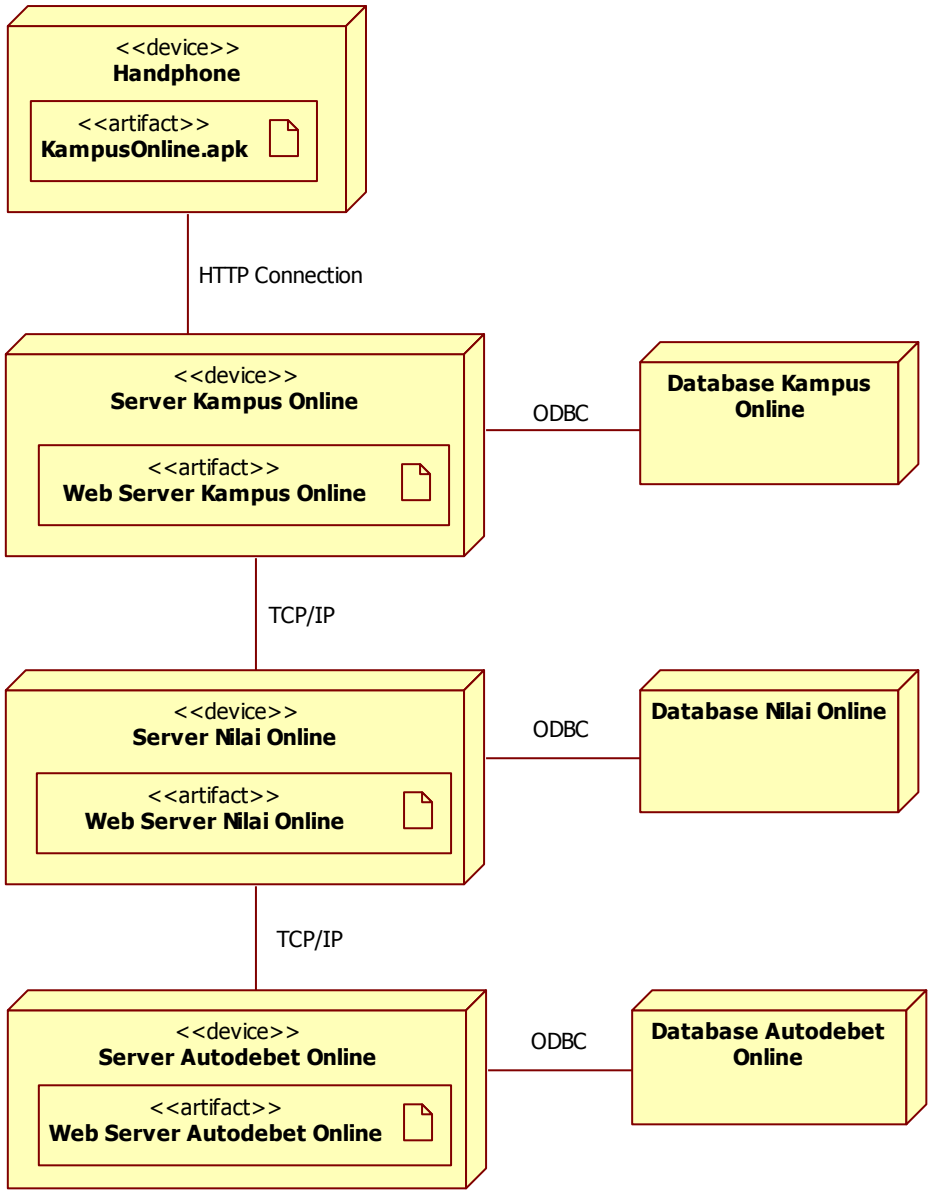
Package diagram aplikasi *mobile* Kampus *Online* (Gambar III.147) terdiri dua paket utama, yaitu paket src dan paket res. Paket src merupakan paket yang berisi kumpulan kelas-kelas kontrol dan *entity* dari aplikasi *mobile* Kampus *Online*. Sedangkan paket res merupakan paket yang berisi *external resources* dari aplikasi *mobile* Kampus *Online*. Berikut ini penjelasan dari masing-masing paket tersebut:

1. Paket src merupakan paket yang berisi paket untuk kelas-kelas kontrol dan *entity*, yaitu paket adapter, paket entity, paket parser, dan paket utility. Adapun penjelasan dari masing-masing paket tersebut adalah sebagai berikut:
    a. Paket adapter, merupakan paket yang berisi kelas-kelas kontrol yang berfungsi untuk memetakkan data respon dari *server* ke dalam tampilan antarmuka aplikasi *mobile* Kampus *Online*.
    b. Paket entity, merupakan paket yang berisi kelas-kelas yang berfungsi sebagai penampung data sementara dari *server* hasil pemetaan dari kelas parser.
    c. Paket parser, merupakan paket yang berisi kelas-kelas yang berfungsi untuk memetakkan data respon dari *server* ke dalam kelas-kelas di dalam paket entity.
    d. Paket utility, merupakan paket yang berisi kelas-kelas utilitas seperti kelas APIKampus yang berisi kumpulan metode dalam API Kampus *Online*, kelas APIConnections yang berfungsi untuk mengunduh data dari *server*, dan kelas-kelas utilitas lainnya.

2. Paket res merupakan paket yang berisi sumber daya aplikasi (*external resources*) yang terdiri dari paket anim, paket color, paket drawable, dan paket layout, paket menu, paket value, dan paket xml. Adapun penjelasan dari masing-masing paket tersebut adalah sebagai berikut:
    a. Paket anim, merupakan paket yang berisi kelas-kelas yang berfungsi sebagai kontrol animasi pergantian halaman dari satu halaman ke halaman lainnya.
    a. Paket color, merupakan paket yang berisi kelas-kelas yang digunakan sebagai kontrol warna dari komponen antarmuka, seperti *text area*, *edit text*, dll.
    b. Paket drawable, merupakan paket yang berisi *file-file* gambar dan kelas-kelas yang berfungsi sebagai kontrol warna gradasi dari komponen antarmuka seperti komponen *button*, *edi ttext*, dll.
    c. Paket layout, merupakan paket yang berisi kelas-kelas yang digunakan untuk rancangan antarmuka (*interface*).
    d. Paket menu, merupakan paket yang berisi kelas-kelas yang digunakan untuk rancangan menu pada setiap halaman.
    e. Paket values, merupakan paket yang berisi kelas-kelas yang digunakan untuk menampung nilai statis berupa *array* atau *string*.
    f. Paket xml, merupakan paket yang berisi kelas yang berisi rancangan antarmuka dari halaman *setting* aplikasi *mobile* Kampus *Online*.

3. 1. 9. 7. ***Deployment Diagram***

Deployment diagram menggambarkan bagaimana infrastruktur sistem dari aplikasi *mobile* Kampus *Online* yang terdiri dari *handphone client*, *server*, basis data, dan hubungan antara komponen-komponen tersebut. Adapun *deployment diagram* aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.148.



Gambar III.148 *Deployment Diagram* Aplikasi *Mobile* Kampus *Online*

Deployment diagram aplikasi *mobile* Kampus *Online* (Gambar III.148) terdiri tiga tipe *node*, yaitu *client* yang diwakili dengan *handphone*, *server* yang terdiri dari *server* Kampus *Online*, *server* Nilai *Online*, dan *server* Autodebet *Online*, serta basis data yang terdiri dari basis data Kampus *Online*, basis data Nilai *Online*, dan basis data Autodebet *Online*. Berikut ini penjelasan masing-masing dari tiga tipe *node* tersebut:

1. Node *client*, merupakan perangkat (*handphone*) tempat aplikasi *mobile* Kampus *Online* berada. Node *client* berkomunikasi dengan server menggunakan jalur internet (HTTP Connections).
2. Node server, merupakan tempat web server dari masing-masing fasilitas yang melayani segala bentuk permintaan client seperti mengirim sapaan, mengecek nilai online, dan mengecek autodebet online. Node server terdiri dari server Kampus Online, server Nilai Online, dan server Autodebet Online. Node Server. Ketiga server tersebut saling terhubung dan berkomunikasi menggunakan jalur TCP/IP (Transmission Control Protocol/Internet Protocol).
3. Node basis data, merupakan tempat penyimpanan data dari masing-masing fasilitas yang ada. Node basis data terdiri dari basis data Kampus Online, basis data Nilai Online, dan basis data Autodebet Online. Ketiga basis data tersebut terhubung dengan server masing-masing dengan menggunakan jalur basis data ODBC (Open Database Connectivity).

3. 2 Perancangan

Perancangan sistem merupakan penggambaran, perencanaan, dan pembuatan dari beberapa elemen sistem yang terpisah ke dalam suatu kesatuan model yang utuh berdasarkan analisis sistem yang telah dilakukan sebelumnya. Tahapan perancangan pembangunan aplikasi *mobile* Kampus *Online* ini meliputi perancangan struktur *file*, perancangan struktur menu, perancangan kerangka kerja, perancangan antarmuka, jaringan semantik, dan perancangan prosedural.

3. 2. 1. Struktur *File*

Struktur *file* menggambarkan atribut dari *file-file* yang digunakan sebagai penampung data sementara hasil dari permintaan data terhadap *server*. Berikut ini *file-file* yang digunakan sebagai penampung data sementara dalam aplikasi *mobile* Kampus *Online* seperti terlihat pada Tabel III.146 sampai dengan Tabel III.163.

1. LoginData.java

File LoginData.java merupakan *file* yang digunakan untuk menampung data *login* ketika pengguna *login* ke dalam aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* LoginData.java terlihat pada Tabel III.146.

Tabel III.146 Struktur *File* LoginData.java

Nama Atribut	Tipe	Ukuran
login_session	String	-
login_userName	String	-
login_displayName	String	-
login_tipeUser	String	-
login_supportFacebook	String	-
login_supportTwitter	String	-
login_profilePicture	String	-
login_keterangan	String	-

2. StatusData.java

File StatusData.java merupakan *file* yang digunakan untuk menampung data status dalam aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* StatusData.java terlihat pada Tabel III.147.

Tabel III.147 Struktur *File* StatusData.java

Nama Atribut	Tipe	Ukuran
nextStart	String	-
statusCount	String	-
statUses	List	-

3. StatusItem.java

File StatusItem.java merupakan *file* yang digunakan untuk menampung data item status dalam aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* StatusItem.java terlihat pada Tabel III.148.

Tabel III.148 Struktur *File* StatusItem.java

Nama Atribut	Tipe	Ukuran
id	String	-
poster	UserData	-
target	UserData	-
listMention	List	-
posted	String	-
text	String	-
textSpannable	String	-
stat	String	-
canDelete	int	$(-2^{31}-1) - 2^{31}$

4. StatusRate.java

File StatusRate.java merupakan *file* yang digunakan untuk menampung data *rating* status dalam aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* StatusRate.java terlihat pada Tabel III.149.

Tabel III.149 Struktur *File* StatusRate.java

Nama Atribut	Tipe	Ukuran
comments	int	$(-2^{31}-1) - 2^{31}$
bagus	int	$(-2^{31}-1) - 2^{31}$
jelek	int	$(-2^{31}-1) - 2^{31}$
rated	int	$(-2^{31}-1) - 2^{31}$

5. NotifikasiData.java

File NotifikasiData.java merupakan *file* yang digunakan untuk menampung data notifikasi dalam aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* NotifikasiData.java terlihat pada Tabel III.150.

Tabel III.150 Struktur *File* NotifikasiData.java

Nama Atribut	Tipe	Ukuran
unread	int	$(-2^{31}-1) - 2^{31}$
nextUpdateId	String	-
listNotifikasi	List	-

6. NotifikasiItem.java

File NotifikasiItem.java merupakan *file* yang digunakan untuk menampung data item notifikasi dalam aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* NotifikasiItem.java terlihat pada Tabel III.151.

Tabel III.151 Struktur *File* NotifikasiItem.java

Nama Atribut	Tipe	Ukuran
id	String	-
unixTime	String	-
time	String ta	-
message	String	-
unread	int	$(-2^{31}-1) - 2^{31}$
icon	String	-
type	String	-
statusId	String	-
friend	UserData	-

7. StatusKomentar.java

File StatusKomentar.java merupakan *file* yang digunakan untuk menampung data komentar suatu status dalam aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* StatusKomentar.java terlihat pada Tabel III.152.

Tabel III.152 Struktur *File* StatusKomentar.java

Nama Atribut	Tipe	Ukuran
nextStart	String	-
showCount	String	-
count	String	-
statusKomentarItems	List	-

8. StatusKomentarItem.java

File StatusKomentarItem.java merupakan *file* yang digunakan untuk menampung data item komentar suatu status dalam aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* StatusKomentarItem.java terlihat pada Tabel III.153.

Tabel III.153 Struktur *File* StatusKomentarItem.java

Nama Atribut	Tipe	Ukuran
id	String	-
poster	UserData	-
unixTime	long	$(-2^{63}-1) - 2^{63}$
listMention	List	-
posted	String	-
text	String	-
textSpannable	String	-

9. UserData.java

File UserData.java merupakan *file* yang digunakan untuk menampung data item pengguna aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* UserData.java terlihat pada Tabel III.154.

Tabel III.154 Struktur *File* UserData.java

Nama Atribut	Tipe	Ukuran
userName	String	-
tipe	String	-
displayName	String	-
noId	String	-
profilePicture	String	-
keterangan	String	-
fsCode	String	-
fsDesc	String	-
sCode	int	$(-2^{31}-1) - 2^{31}$
sMessage	String	-

10. TemanData.java

File TemanData.java merupakan *file* yang digunakan untuk menampung data teman pada aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* TemanData.java terlihat pada Tabel III.155.

Tabel III.155 Struktur *File* TemanData.java

Nama Atribut	Tipe	Ukuran
nextStart	String	-
showCount	String	-
count	String	-
getType	String	-
friendOf	String	-

11. TemanDataItem.java

File TemanDataItem.java merupakan *file* yang digunakan untuk menampung data item teman pada aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* TemanDataItem.java terlihat pada Tabel III.156.

Tabel III.156 Struktur *File* TemanDataItem.java

Nama Atribut	Tipe	Ukuran
username	String	-
friendCode	int	$(-2^{31}-1) - 2^{31}$
friendType	String	-
friendData	UserData	-

12. StatusPost.java

File StatusPost.java merupakan *file* yang digunakan untuk menampung respon data ketika pengguna melakukan *update status* pada aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* StatusPost.java terlihat pada Tabel III.157.

Tabel III.157 Struktur *File* StatusPost.java

Nama Atribut	Tipe	Ukuran
statusId	String	-

13. NilaiData.java

File NilaiData.java merupakan *file* yang digunakan untuk menampung data nilai pada aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* NilaiData.java terlihat pada Tabel III.158.

Tabel III.158 Struktur *File* NilaiData.java

Nama Atribut	Tipe	Ukuran
current	String	-
count	int	$(-2^{31}-1) - 2^{31}$
nilaiItem	List	-

14. NilaiDetailData.java

File NilaiDetailData.java merupakan *file* yang digunakan untuk menampung data detail nilai pada aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* NilaiDetailData.java terlihat pada Tabel III.159.

Tabel III.159 Struktur *File* NilaiDetailData.java

Nama Atribut	Tipe	Ukuran
nim	String	-
nama	String	-
kodejur	String	-
kodefak	String	-
prodi	String	-
kelas	String	-
thnmasuk	String	-
pmdk	String	-
nipWali	String	-
dosenWali	String	-
fakultas	String	-
jurusan	String	-

15. NilaiDataItem.java

File NilaiDataItem.java merupakan *file* yang digunakan untuk menampung data item nilai pada aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* NilaiDataItem.java terlihat pada Tabel III.160.

Tabel III.160 Struktur *File* NilaiDataItem.java

Nama Atribut	Tipe	Ukuran
tahunakad	String	-
tahun	String	-
semester	String	-

16. AutodebetData.java

File AutodebetData.java merupakan *file* yang digunakan untuk menampung data autodebet pada aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* AutodebetData.java terlihat pada Tabel III.161.

Tabel III.161 Struktur *File* AutodebetData.java

Nama Atribut	Tipe	Ukuran
current	String	-
count	int	$(-2^{31}-1) - 2^{31}$
autodebetItem	List	-

17. AutodebetDataItem.java

File AutodebetDataItem.java merupakan *file* yang digunakan untuk menampung data item autodebet pada aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* AutodebetDataItem.java terlihat pada Tabel III.162.

Tabel III.162 Struktur *File* AutodebetDataItem.java

Nama Atribut	Tipe	Ukuran
tahunakad	String	-
tahun	String	-

18. AutodebetDetailData.java

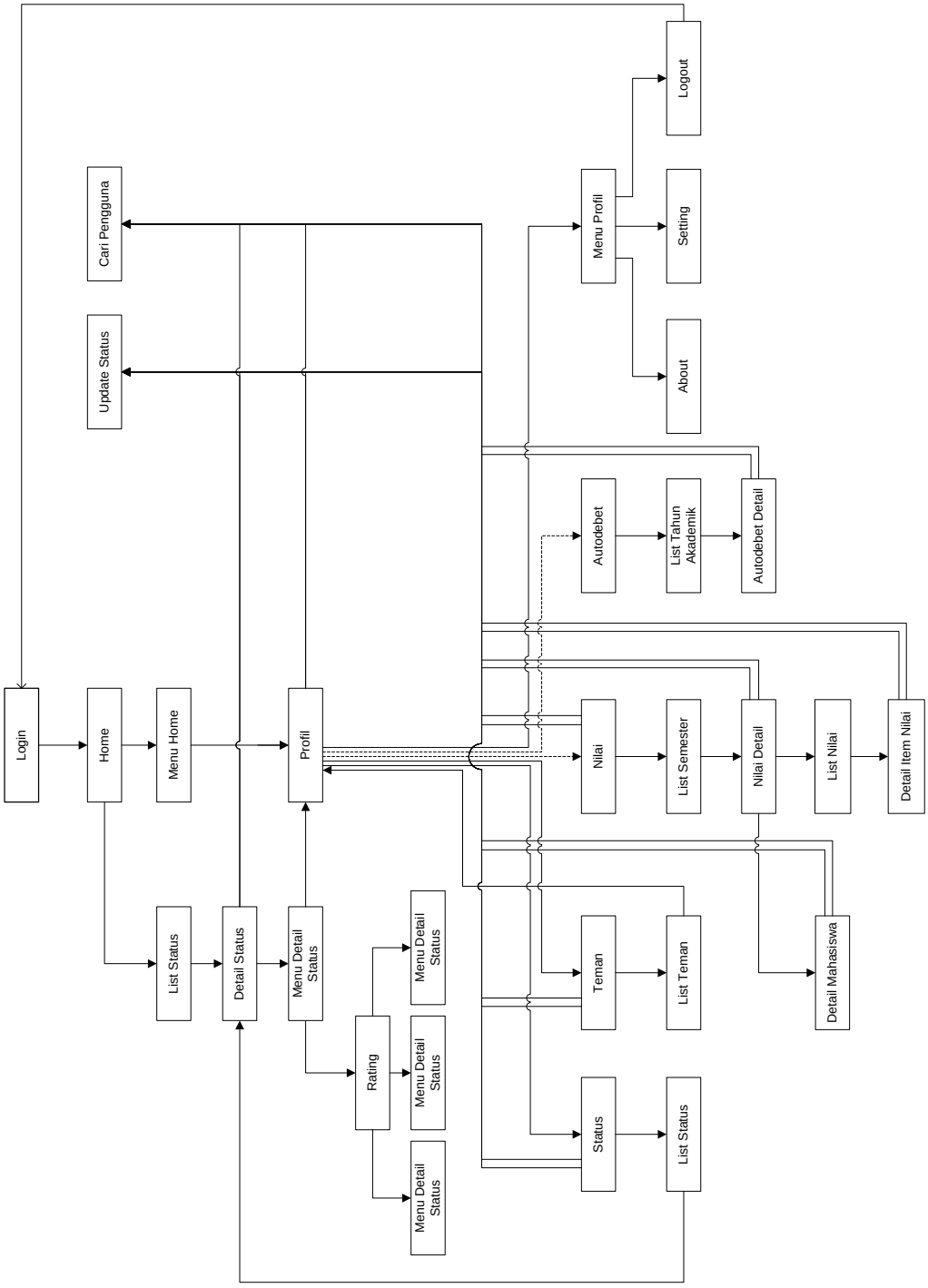
File AutodebetDetailData.java merupakan *file* yang digunakan untuk menampung data detail autodebet pada aplikasi *mobile* Kampus *Online*. Adapun struktur *file* dari *file* AutodebetDetailData.java terlihat pada Tabel III.163.

Tabel III.163 Struktur *File* AutodebetDetailData.java

Nama Atribut	Tipe	Ukuran
Nim	String	-
Nama	String	-
autodebetDataItem	AutodebetDataItem	-
listAutodebetDetailDataItem	List	-

3. 2. 2. Struktur Menu

Perancangan struktur menu merupakan gambaran jalur pemakaian aplikasi sehingga aplikasi yang dibangun mudah dipahami dan mudah digunakan. Perancangan struktur menu dari aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.149.



Gambar III.149 Struktur Menu Aplikasi *Mobile* Kampus *Online*

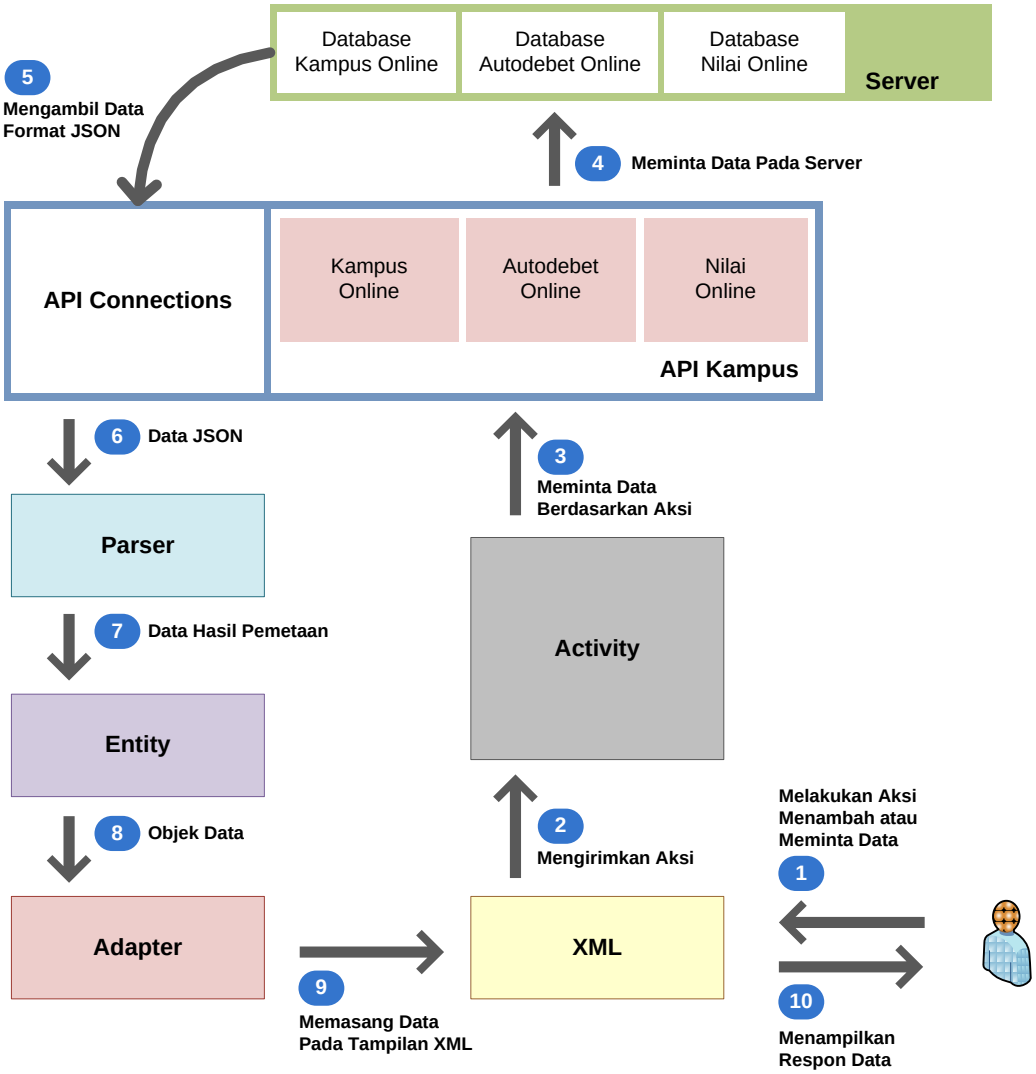
3. 2. 3. Perancangan Kerangka Kerja

Perancangan kerangka kerja menggambarkan alur kerja dari masing-masing komponen yang terdapat pada aplikasi *mobile* Kampus *Online*. Adapun komponen yang terdapat pada aplikasi *mobile* Kampus *Online*, yaitu XML, Activity, API Kampus, Server, API Connections, Parser, Entity, dan Adapter. Berikut penjelasan tentang fungsi dari masing-masing komponen tersebut:

1. XML, merupakan komponen yang berfungsi sebagai media masukan data dari pengguna dan sebagai tampilan (*interface*) yang menampilkan respon data dari *server* hasil permintaan data dari pengguna.
2. Activity, merupakan komponen yang berfungsi untuk mengatur permintaan dari pengguna melalui komponen XML.
3. API Kampus, merupakan komponen yang berisi metode-metode untuk melakukan permintaan data terhadap *server*.
4. Server, merupakan komponen yang berfungsi untuk melayani setiap permintaan yang dilakukan oleh aplikasi *mobile* Kampus *Online* melalui metode yang terdapat pada API Kampus.
5. API Connections, merupakan komponen yang berfungsi untuk mengambil respon data dari *server* hasil permintaan data dari metode-metode yang terdapat pada API Kampus.
6. Parser, merupakan komponen yang berfungsi untuk memetakkan respon data dari *server* yang telah diambil oleh API Connections ke dalam bentuk objek (Entity).

7. Entity, merupakan komponen penampung respon data sementara hasil permintaan dari aplikasi *mobile* Kampus *Online* sebelum akhirnya ditampilkan ke pengguna.
8. Adapter, merupakan komponen yang memetakkan data yang terdapat pada komponen Entity ke dalam bentuk XML.

Berikut ini gambaran alur kerja dari setiap komponen aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.150.



Gambar III.150 Perancangan Kerangka Kerja Aplikasi *Mobile* Kampus *Online*

Perancangan kerangka kerja aplikasi *mobile* Kampus *Online* (Gambar III.150) terdiri dari 10 tahapan dari mulai aksi yang dilakukan oleh pengguna sampai dengan tampilnya respon data dari *server*. Adapun penjelasan setiap tahapan dari kerangka kerja aplikasi *mobile* Kampus *Online* adalah sebagai berikut:

1. Pengguna aplikasi *mobile* Kampus *Online* melakukan aksi baik itu berupa penambahan data (status, sapaan, *rating*) maupun permintaan data (profil pengguna, komentar, dll). Aksi dari pengguna akan diterima oleh komponen XML yang berfungsi sebagai media untuk mengambil aksi dari pengguna.
2. Hasil aksi dari pengguna yang ditangkap oleh XML kemudian akan disalurkan pada komponen Activity yang berfungsi sebagai *controller* setiap aksi dari pengguna sebelum akhirnya melakukan permintaan data ke *server*.
3. Permintaan data dilakukan oleh komponen Activity berdasarkan tipe permintaan data dari pengguna. Adapun tipe dari permintaan data tersebut terdiri dari tiga tipe, yaitu permintaan data Kampus *Online*, permintaaan data Nilai *Online*, dan permintaan data Autodebet *Online*.
4. API Kampus akan melakukan permintaan data pada *server* yang merupakan hasil permintaan data dari komponen Activity dengan menggunakan metode-metode yang terdapat pada API Kampus *Online*.
5. Setelah API Kampus melakukan permintaan data pada server dengan menggunakan metode tertentu selanjutnya respon data hasil permintaan akan diunduh dari *server* oleh API Connections dengan menggunakan format JSON.

6. Format data JSON hasil respon data dari *server* akan dikirimkan pada komponen Parser yang selanjutnya akan dilakukan pemetaan data format JSON ke dalam objek-objek tertentu sehingga mudah dibaca oleh program.
7. Data hasil pemetaan oleh komponen Parser dikirimkan pada komponen Entity sebagai penampung data sementara dari *server*.
8. Data respon dari *server* yang telah ditampung oleh komponen Entity selanjutnya dikirimkan pada komponen Adapter dalam bentuk objek data.
9. Komponen Adapter bertugas untuk memetakkan setiap baris data yang diperoleh dari komponen Entity pada komponen XML.
10. Komponen XML akan menampilkan data-data yang telah dipetakkan oleh komponen Adapter ke dalam tampilan (*interface*) pengguna.

3. 2. 4. Perancangan Antarmuka

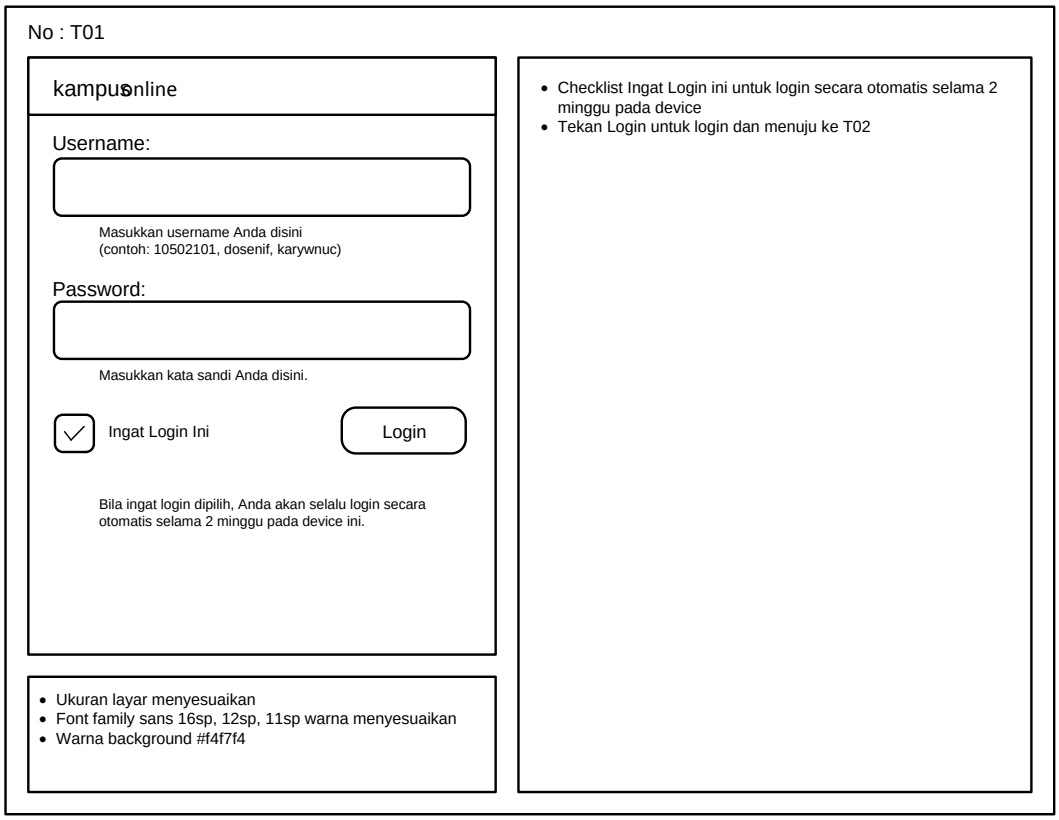
Perancangan antarmuka mendeskripsikan rencana tampilan dari setiap *form* yang akan digunakan pada aplikasi *mobile* Kampus *Online*. Perancangan antarmuka pada aplikasi *mobile* Kampus *Online* terdiri dari perancangan *form*, perancangan pesan, perancangan menu, dan perancangan dialog.

3. 2. 4. 1. **Perancangan *Form***

Perancangan *form* mendeskripsikan rencana tampilan dari setiap *form* yang akan digunakan pada tampilan aplikasi *mobile* Kampus *Online*. Berikut ini gambaran dari perancangan antarmuka aplikasi *mobile* Kampus *Online* seperti terlihat pada Gambar III.51 sampai dengan Gambar III.171.

1. Perancangan *Form Login*

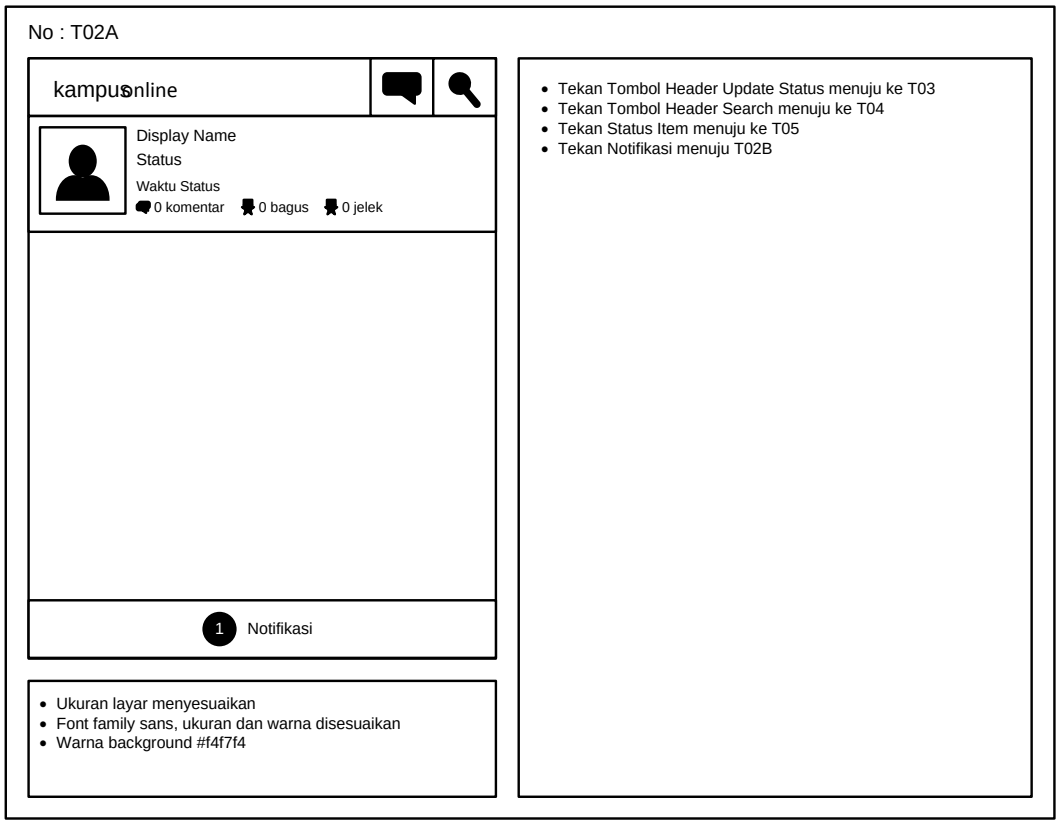
Gambaran umum dari perancangan antarmuka *form login* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.151.



Gambar III.151 Perancangan *Form Login*

2. Perancangan *Form Home*

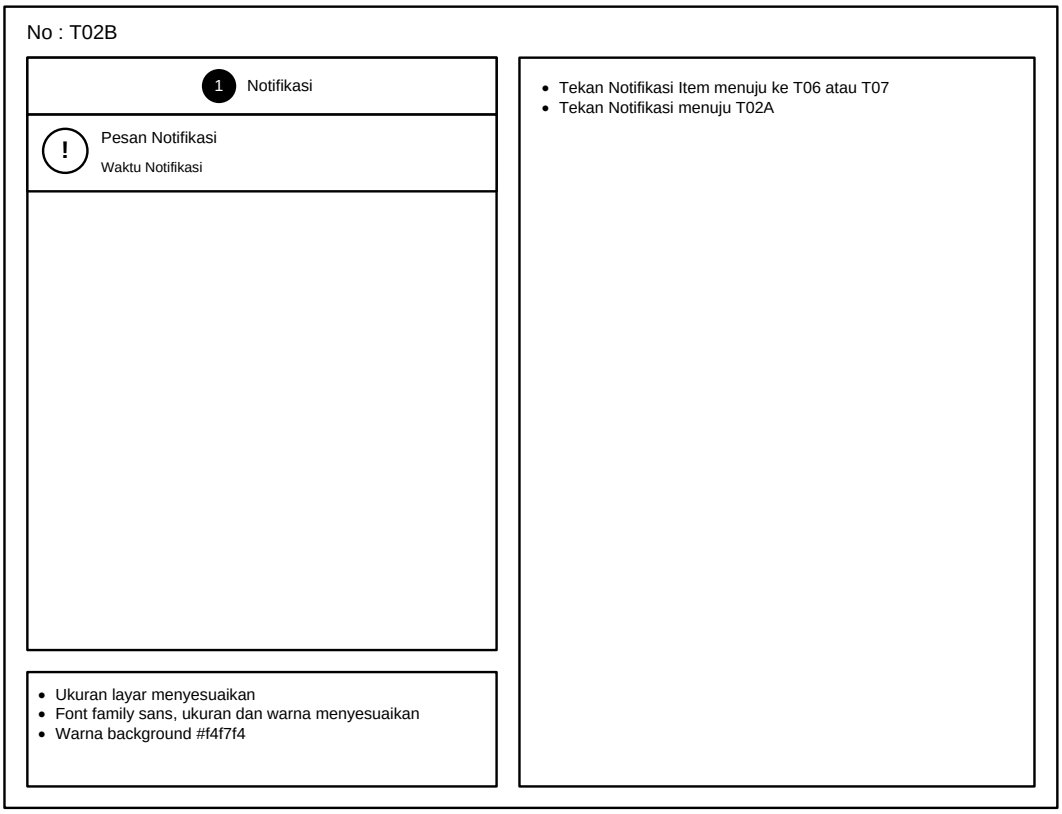
Gambaran umum dari perancangan antarmuka *form home* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.152.



Gambar III.152 Perancangan *Form Home*

3. Perancangan *Form* Notifikasi

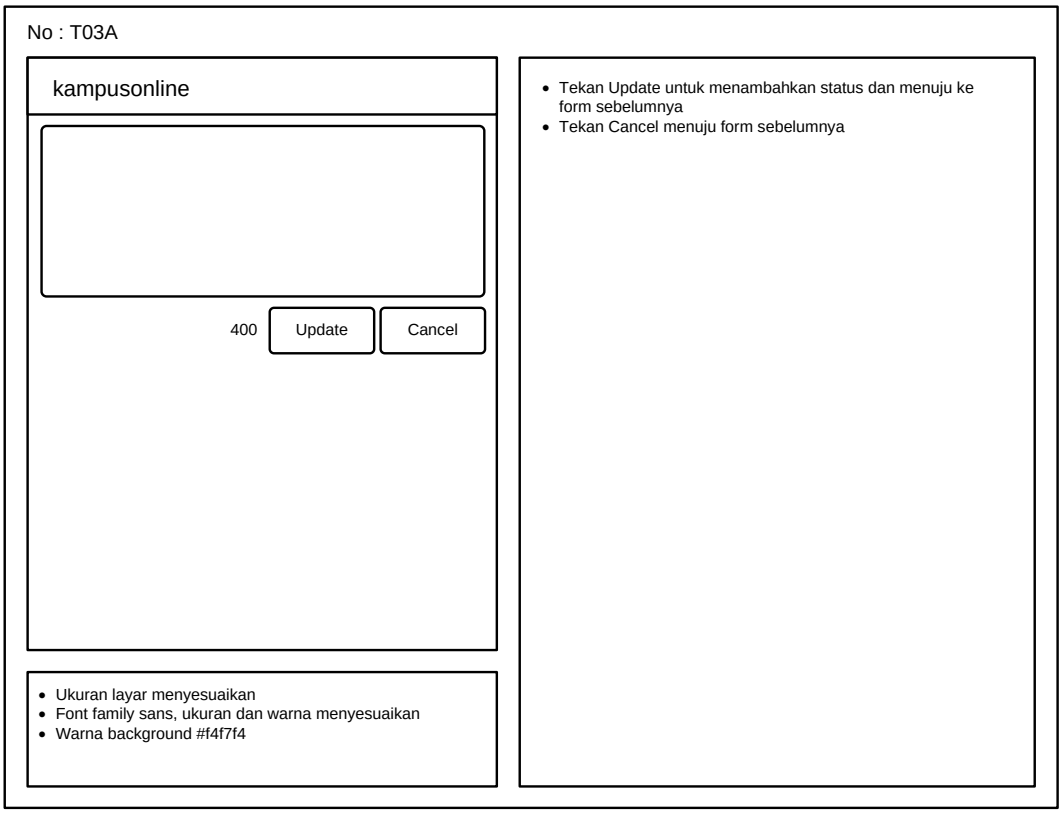
Gambaran umum dari perancangan antarmuka *form* notifikasi pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.153.



Gambar III.153 Perancangan *Form* Notifikasi

4. Perancangan *Form Update Status*

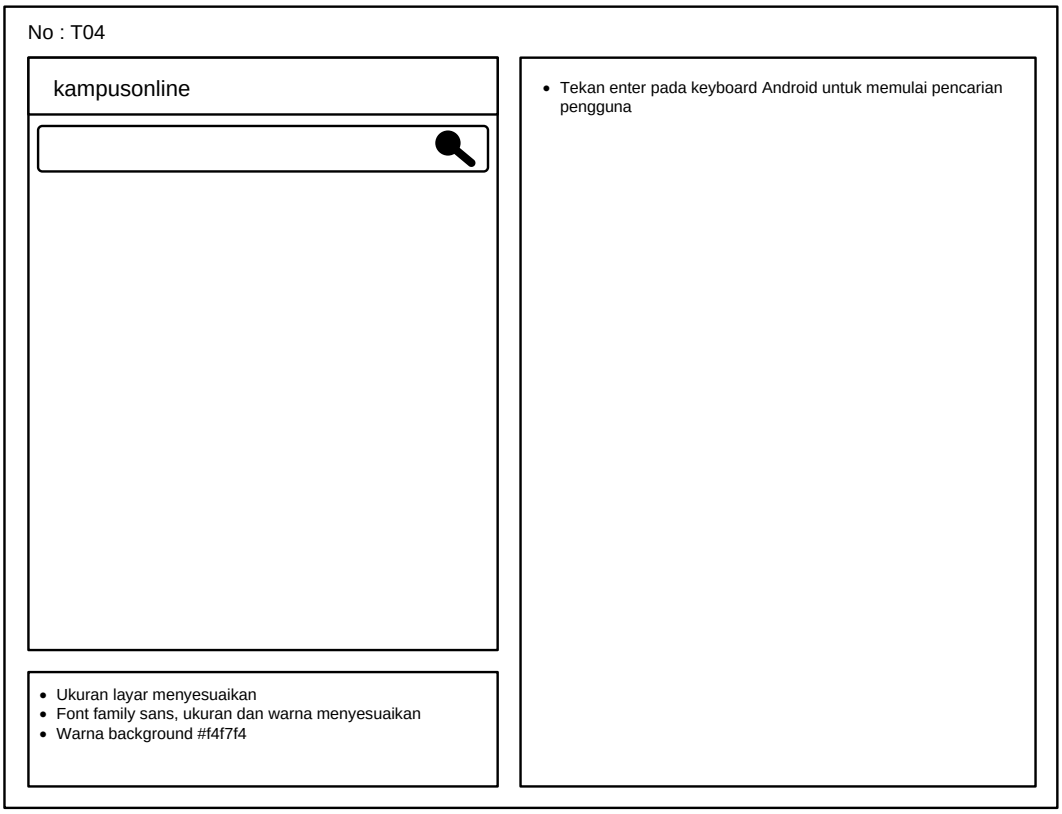
Gambaran umum dari perancangan antarmuka *form update status* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.154.



Gambar III.154 Perancangan *Form Update Status*

5. Perancangan *Form* Cari Pengguna

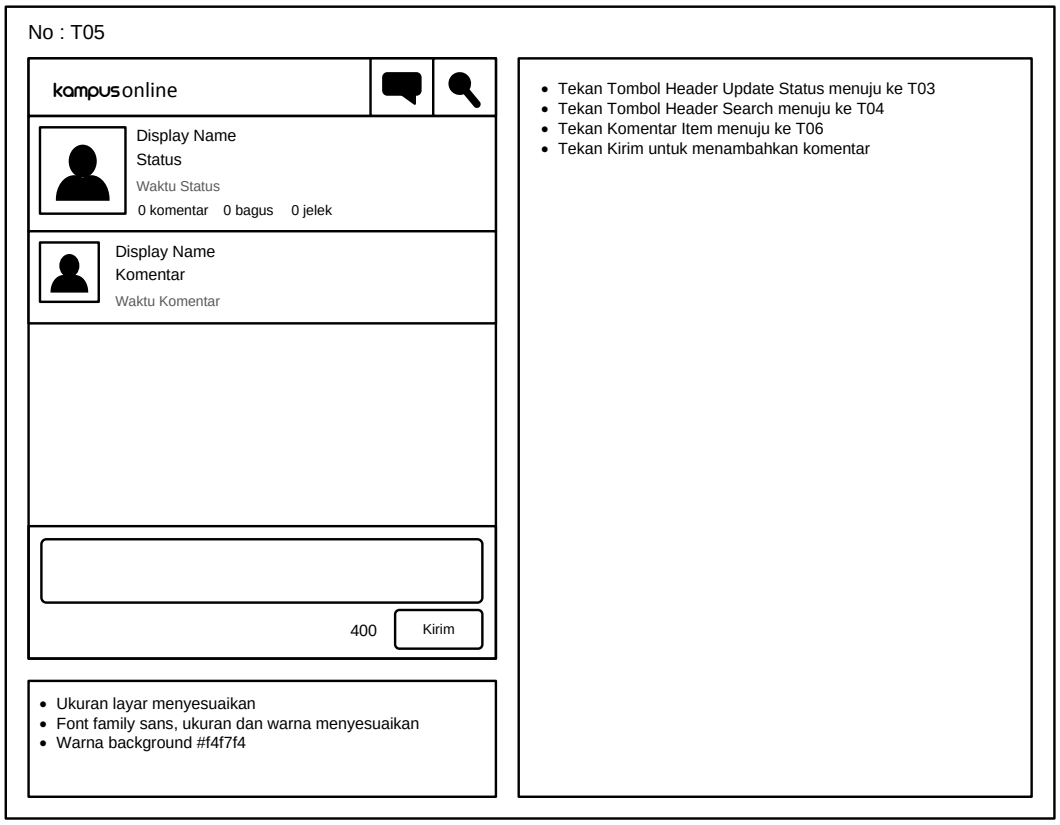
Gambaran umum dari perancangan antarmuka *form* cari pengguna pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.155.



Gambar III.155 Perancangan *Form* Cari Pengguna

6. Perancangan *Form* Detail Status

Gambaran umum dari perancangan antarmuka *form* detail status pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.156.



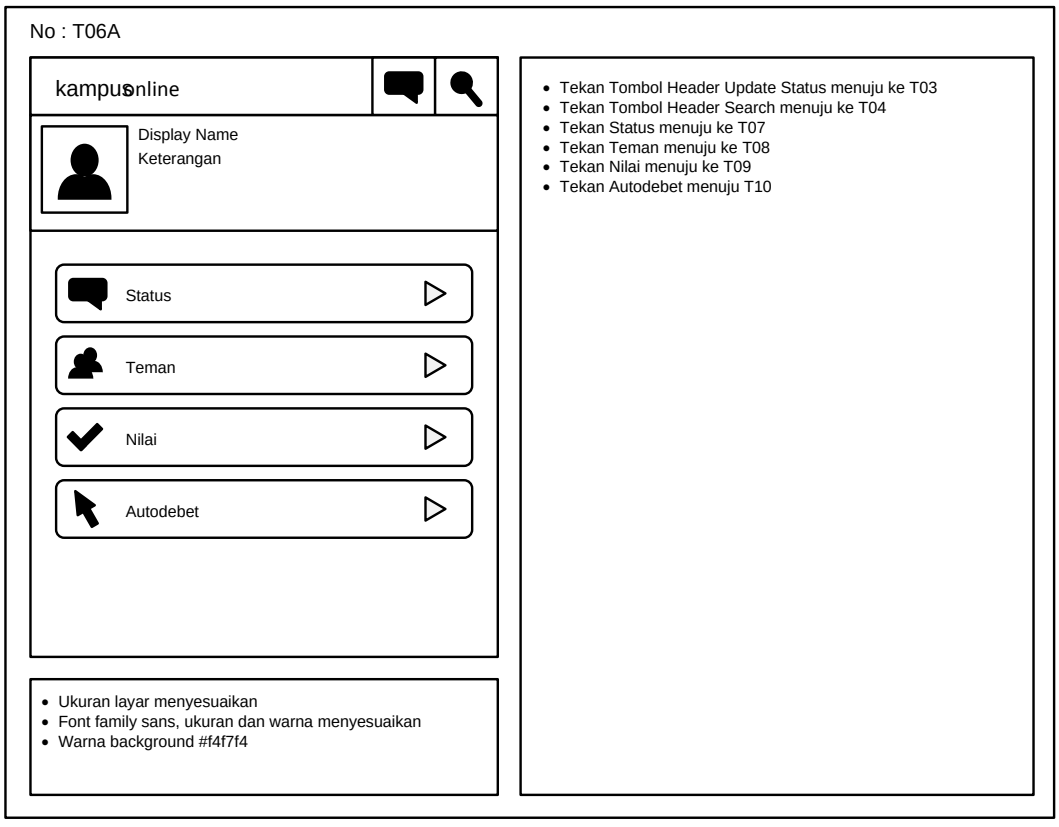
Gambar III.156 Perancangan *Form* Detail Status

7. Perancangan *Form* Profil

Perancangan antarmuka *form* profil berisi data pribadi pengguna aplikasi *mobile* Kampus *Online*. *Form* profil pengguna aplikasi *mobile* Kampus *Online* dibagi menjadi dua tipe profil, yaitu:

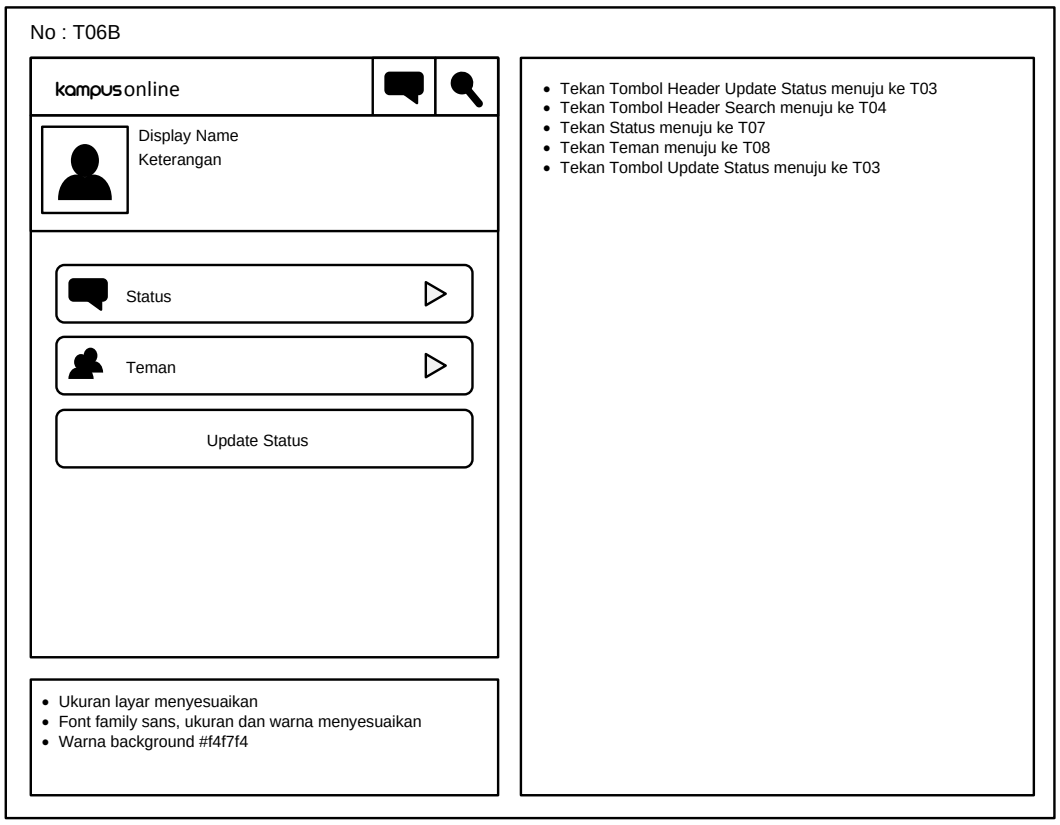
a. *Form* profil pengguna yang login di aplikasi *mobile* Kampus *Online*.

Bagi pengguna yang login di aplikasi *mobile* Kampus *Online* juga memiliki tipe halaman profil yang terbagi menjadi tiga tipe, yaitu tipe mahasiswa dan karyawan. Gambaran umum dari perancangan antarmuka *form* profil mahasiswa pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.157.



Gambar III.157 Perancangan *Form* Profil Mahasiswa

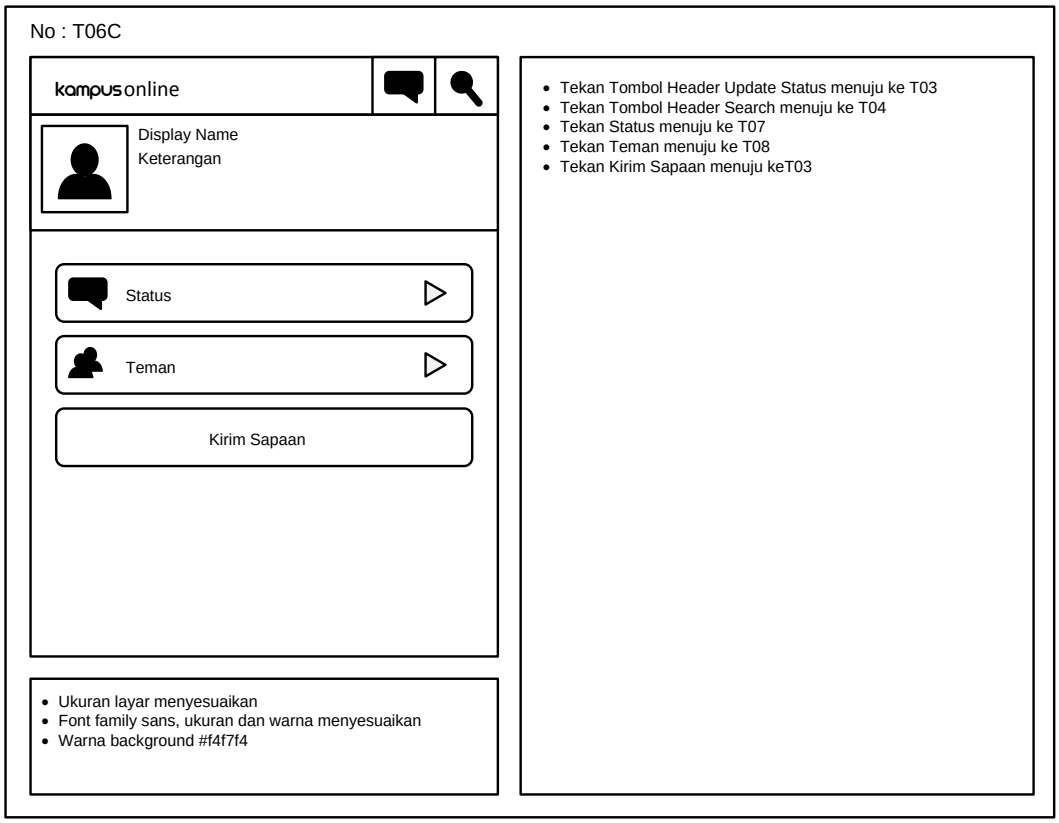
Bagi pengguna yang *login* ke aplikasi *mobile* Kampus *Online* dengan status karyawan tidak akan memiliki data nilai ataupun autodebet. Berikut gambaran umum dari perancangan antarmuka *form* profil karyawan pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.158.



Gambar III.158 Perancangan *Form* Profil Karyawan

b. *Form* profil teman dari pengguna yang *login*.

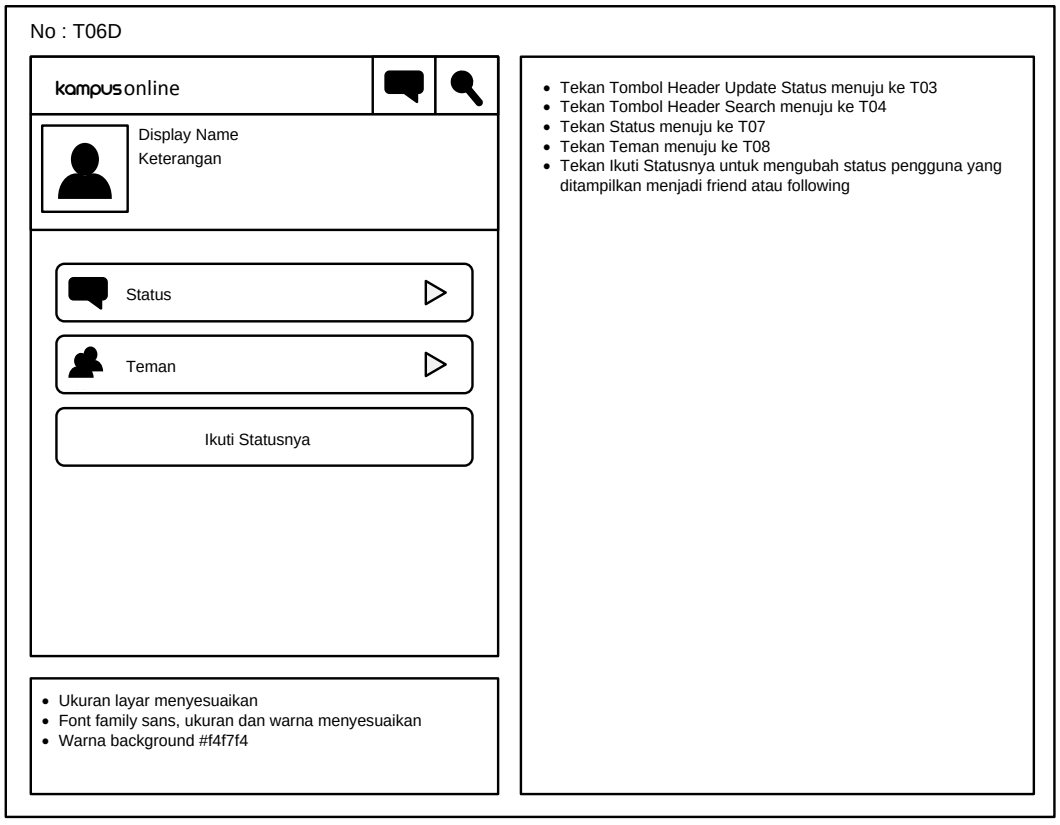
Form profil teman dari pengguna yang login tidak memiliki fasilitas untuk melihat nilai maupun autodebet. Pengguna yang *login* hanya bisa melihat status teman dan daftar pertemanan pengguna tersebut. Adapun gambaran umum dari perancangan antarmuka *form* profil teman pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.159.



Gambar III.159 Perancangan *Form* Profil Teman

c. *Form* profil bukan teman dari pengguna yang *login*.

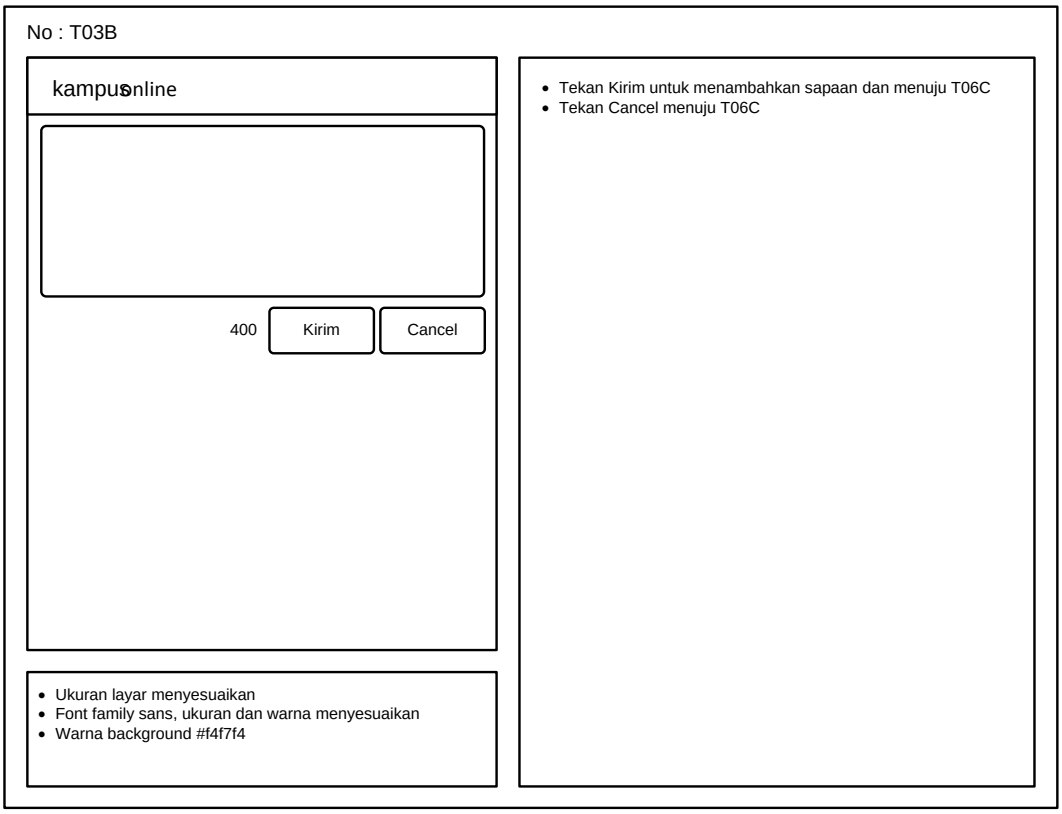
Form profil bukan teman merupakan *form* profil pengguna Kampus *Online* tetapi statusnya belum menjadi *friend* atau *following*. Sehingga pengguna Kampus *Online* yang *login* hanya bisa melihat status, melihat daftar pertemanan, dan mengikuti statusnya. Adapun gambaran umum dari perancangan antarmuka *form* profil bukan teman pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.160.



Gambar III.160 Perancangan *Form* Profil Bukan Teman

8. Perancangan *Form* Kirim Sapaan

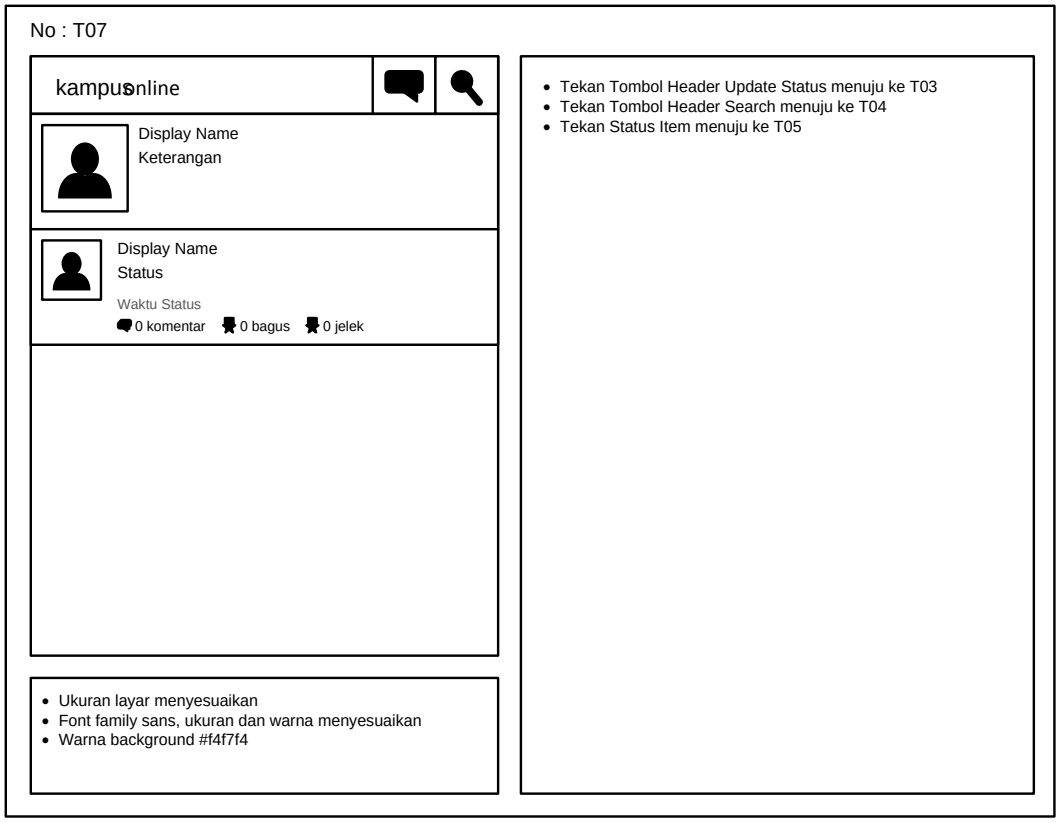
Gambaran umum dari perancangan antarmuka *form* kirim sapaan pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.161.



Gambar III.161 Perancangan *Form* Kirim Sapaan

9. Perancangan *Form* Status Profil

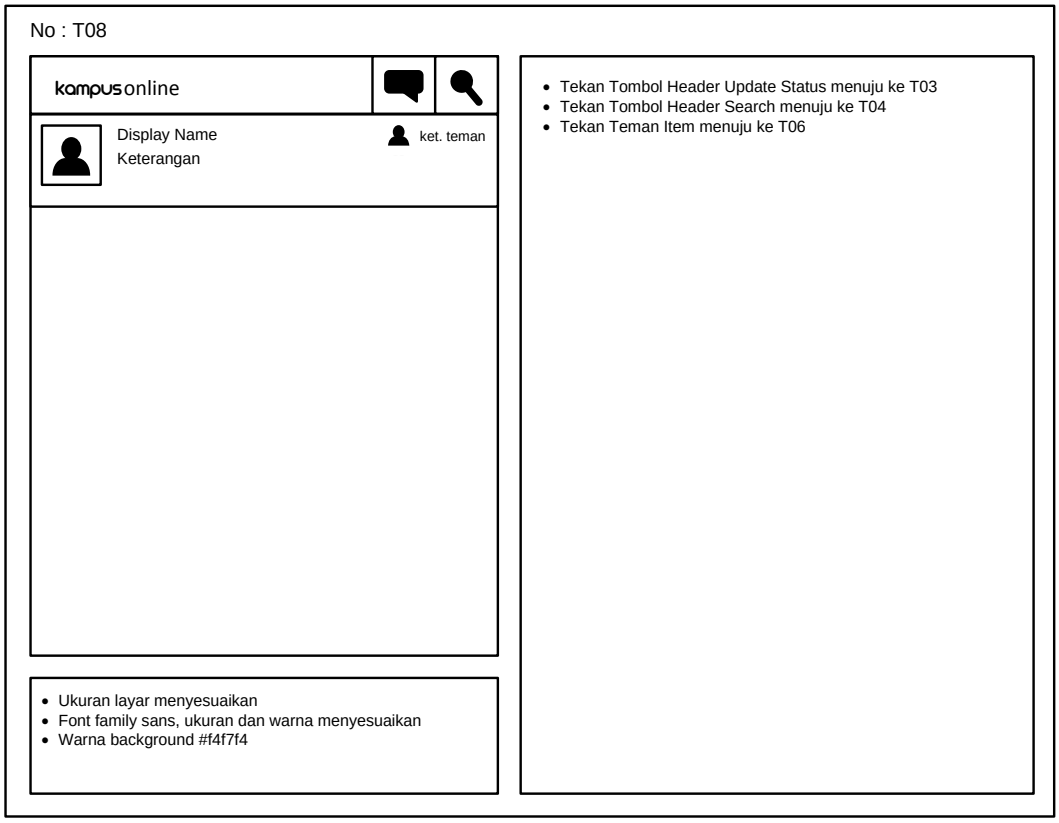
Gambaran umum dari perancangan antarmuka *form* status profil pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.162.



Gambar III.162 Perancangan *Form* Status Profil

10. Perancangan *Form* Teman

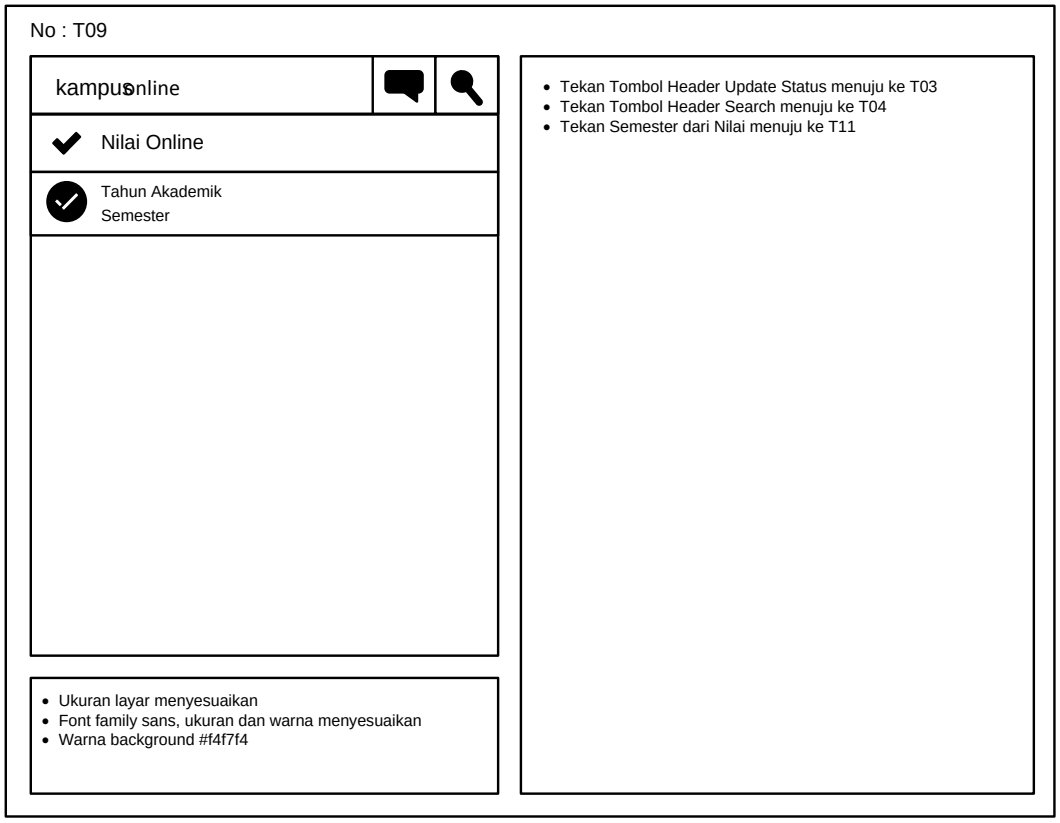
Gambaran umum dari perancangan antarmuka *form* teman pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.163.



Gambar III.163 Perancangan *Form* Teman

11. Perancangan *Form* Nilai

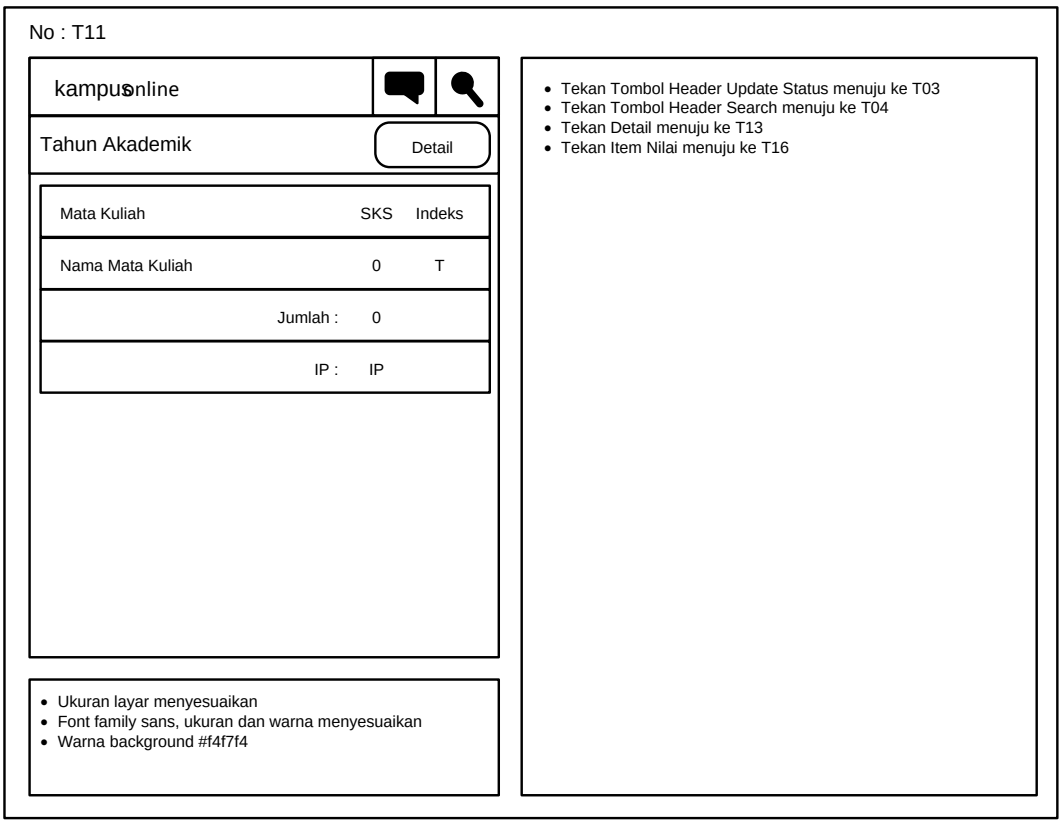
Gambaran umum dari perancangan antarmuka *form* nilai pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.164.



Gambar III.164 Perancangan *Form* Nilai

12. Perancangan *Form* Detail Nilai

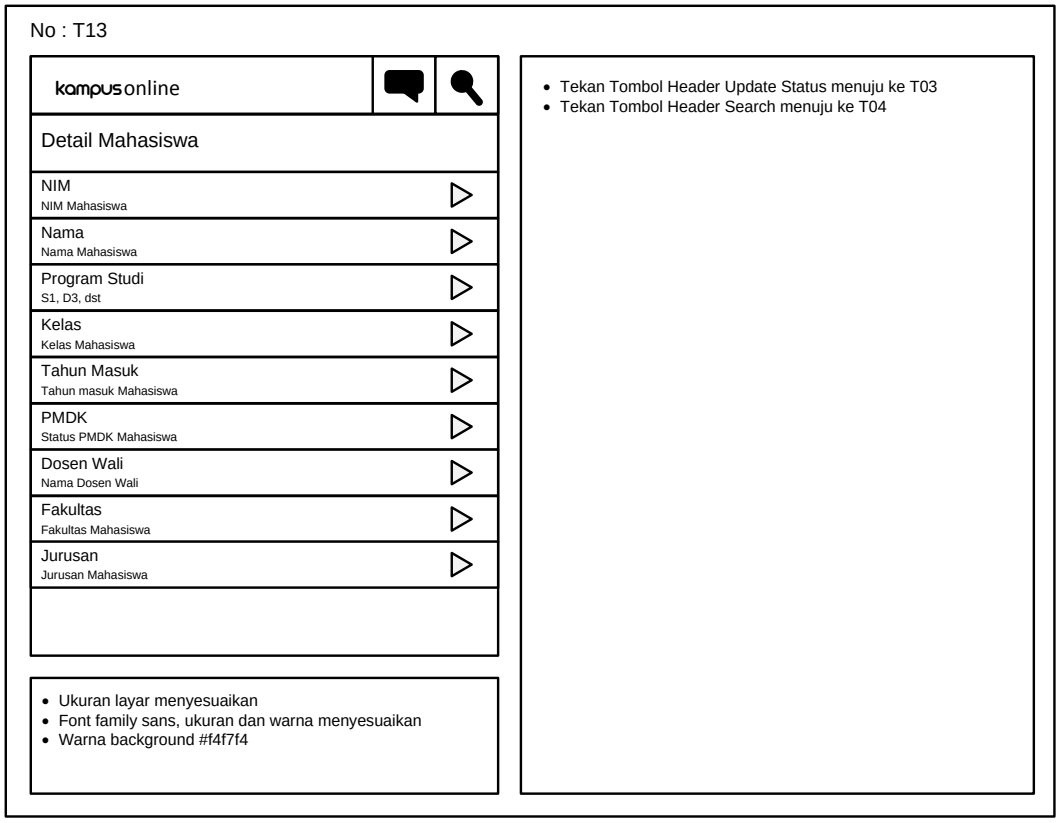
Gambaran umum dari perancangan antarmuka *form* detail nilai setiap semesternya pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.165.



Gambar III.165 Perancangan *Form* Detail Nilai

13. Perancangan *Form* Detail Mahasiswa

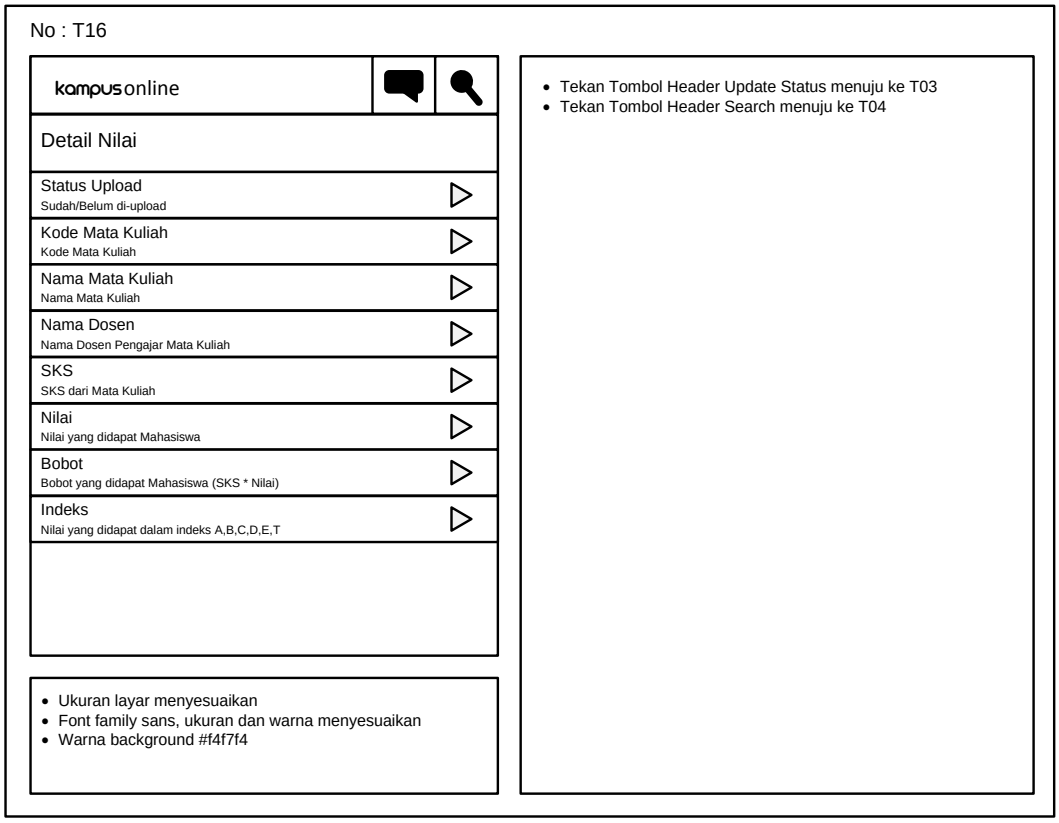
Gambaran umum dari perancangan antarmuka *form* detail mahasiswa pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.166.



Gambar III.166 Perancangan *Form* Detail Mahasiswa

14. Perancangan *Form* Detail Nilai Item

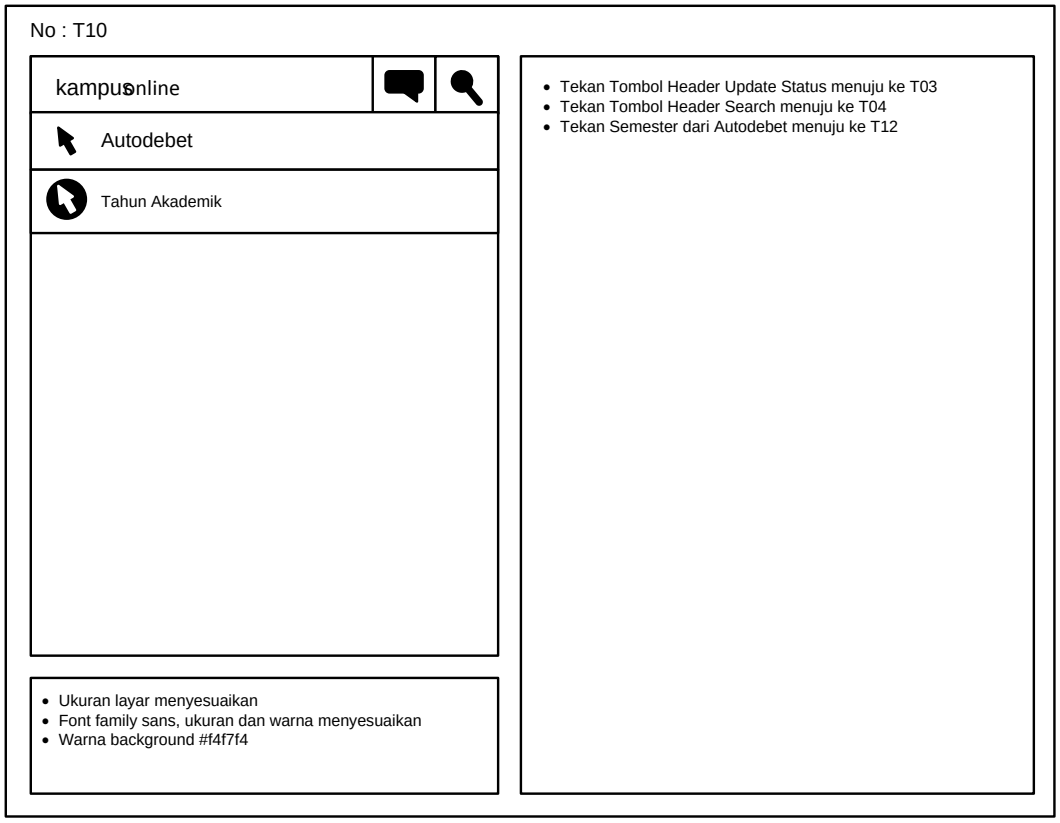
Gambaran umum dari perancangan antarmuka *form* detail nilai item setiap semester pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.167.



Gambar III.167 Perancangan *Form* Detail Nilai Item

15. Perancangan *Form* Autodebet

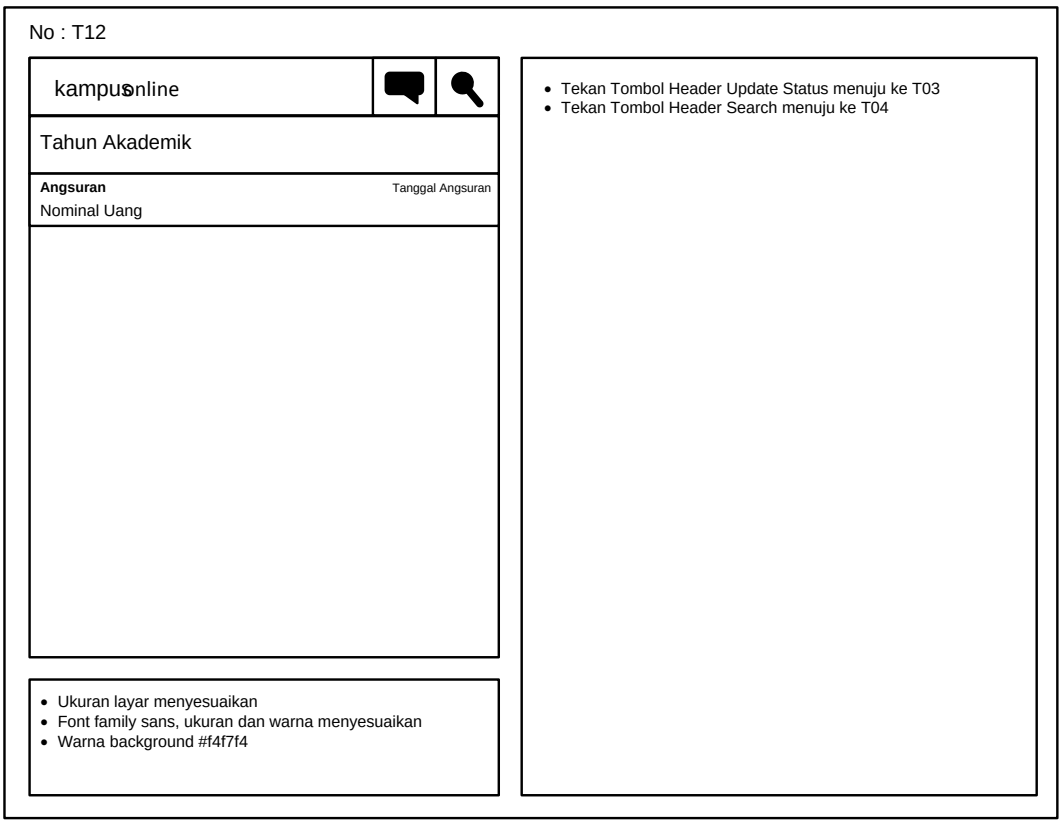
Gambaran umum dari perancangan antarmuka *form* autodebet pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.168.



Gambar III.168 Perancangan *Form* Autodebet

16. Perancangan *Form* Detail Autodebet

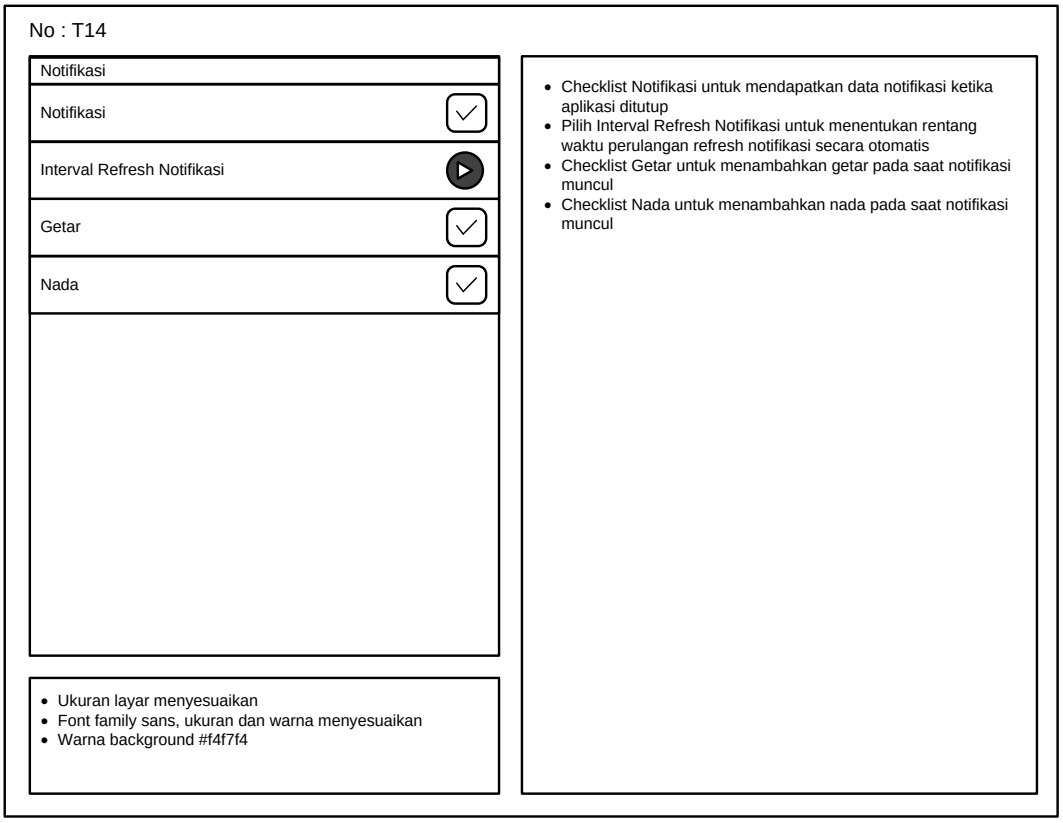
Gambaran umum dari perancangan antarmuka *form* detail autodebet setiap semesternya pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.169.



Gambar III.169 Perancangan *Form* Detail Autodebet

17. Perancangan Form Setting

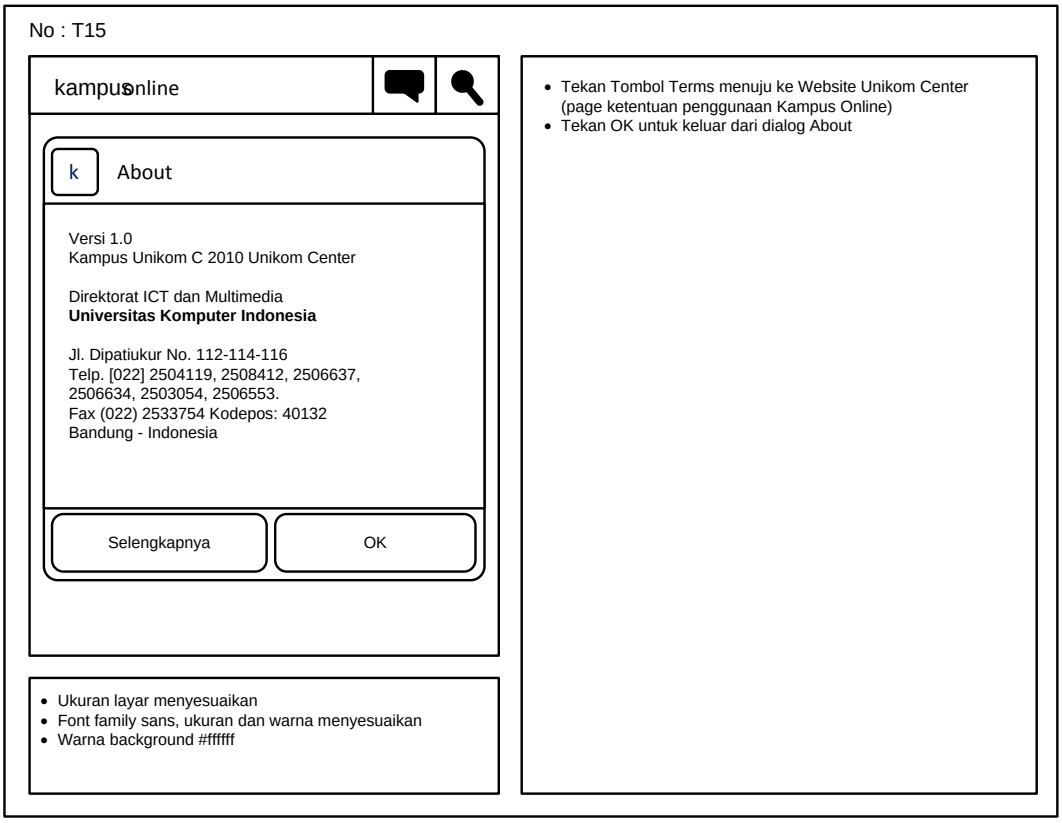
Gambaran umum dari perancangan antarmuka *form setting* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.170.



Gambar III.170 Perancangan *Form Setting*

18. Perancangan *Form About*

Gambaran umum dari perancangan antarmuka *form about* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.171.



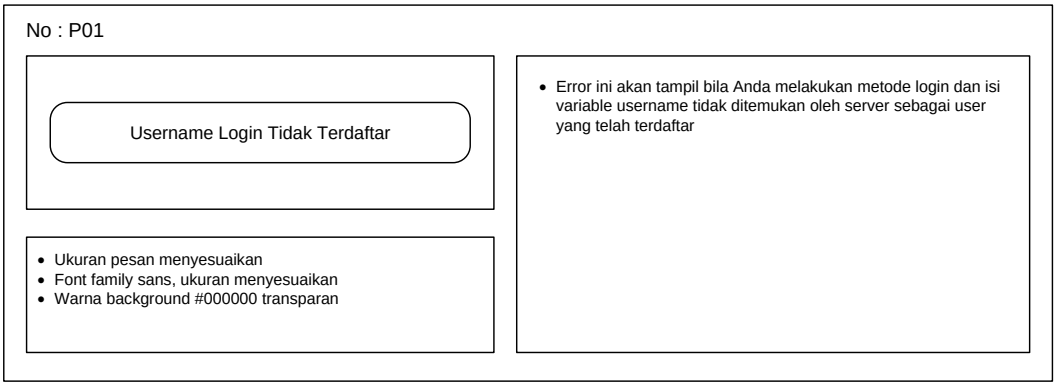
Gambar III.171 Perancangan *Form About*

3. 2. 4. 2. **Perancangan Pesan**

Perancangan pesan mendeskripsikan rencana tampilan dari setiap pesan yang digunakan pada aplikasi *mobile* Kampus *Online*. Berikut ini gambaran dari perancangan pesan pada aplikasi *mobile* Kampus *Online* seperti terlihat pada Gambar III.172 sampai dengan Gambar III.197.

1. Perancangan Pesan Kesalahan *Username Login*

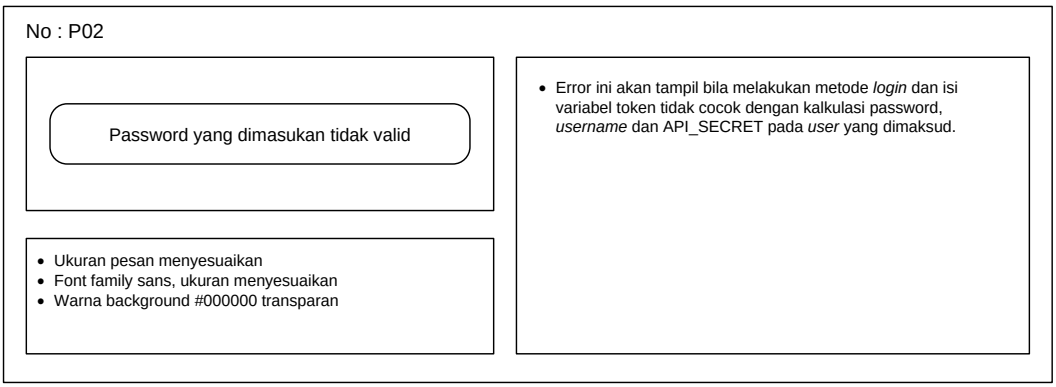
Pesan kesalahan *username login* berisi informasi *username login* yang dimasukkan tidak terdaftar pada Kampus *Online*. Pesan tersebut muncul pada saat pengguna melakukan proses *login*. Adapun gambaran umum dari perancangan pesan kesalahan *username login* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.172.



Gambar III.172 Perancangan Pesan Kesalahan *Username Login*

2. Perancangan Pesan Kesalahan *Password Login*

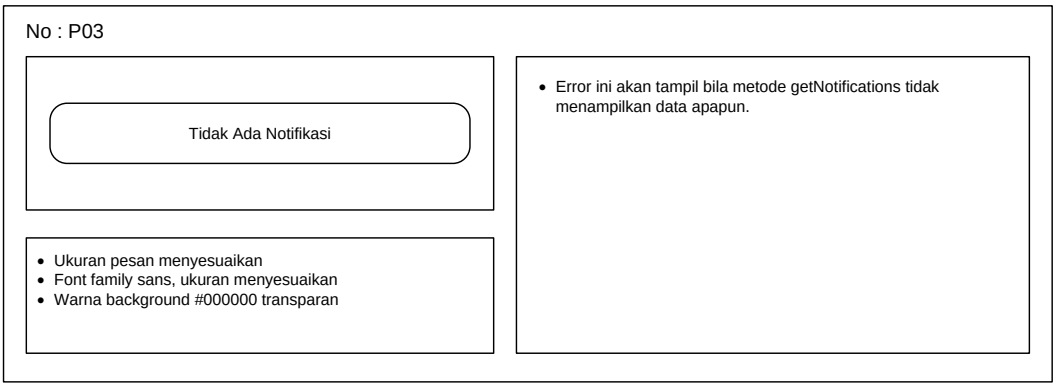
Pesan kesalahan *password login* berisi informasi data *password* dan *username login* yang dimasukkan tidak cocok. Pesan tersebut muncul pada saat pengguna melakukan proses *login*. Gambaran umum dari perancangan pesan kesalahan *password login* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.173.



Gambar III.173 Perancangan Pesan Kesalahan *Password Login*

3. Perancangan Pesan Kesalahan Notifikasi

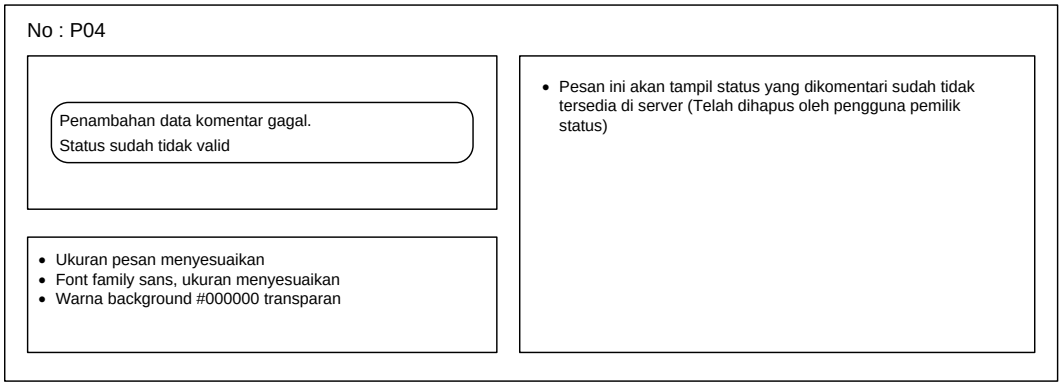
Pesan kesalahan notifikasi berisi informasi bahwa pengguna yang melakukan *login* tidak memiliki data notifikasi. Pesan tersebut muncul pada saat pengguna meminta data notifikasi. Gambaran umum dari perancangan pesan kesalahan notifikasi pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.174.



Gambar III.174 Perancangan Pesan Kesalahan Notifikasi

4. Perancangan Pesan Kesalahan Status

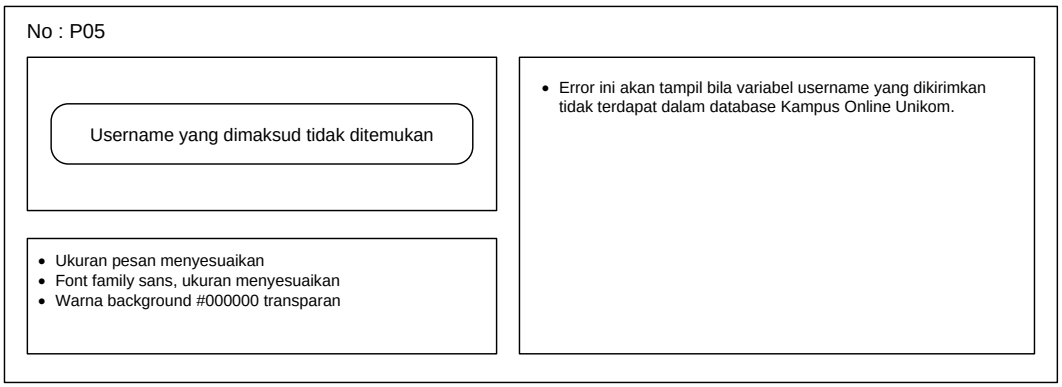
Pesan kesalahan status berisi informasi bahwa status yang dilihat oleh pengguna sudah tidak terdapat pada server (telah dihapus oleh pemilik status). Pesan tersebut muncul ketika pengguna memberi komentar dan memberikan *rating*. Gambaran umum dari perancangan pesan kesalahan status pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.175.



Gambar III.175 Perancangan Pesan Kesalahan Status

5. Perancangan Pesan Kesalahan *Username*

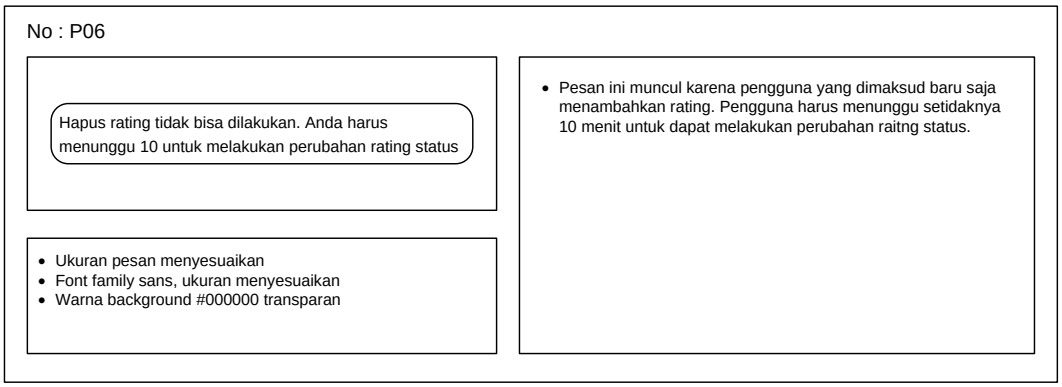
Pesan kesalahan *username* berisi informasi bahwa *username* tidak terdapat pada server. Pesan tersebut muncul ketika pengguna melakukan pencarian pada fasilitas pencarian pengguna Kampus *Online*. Gambaran umum dari perancangan pesan kesalahan *username* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.176.



Gambar III.176 Perancangan Pesan Kesalahan *Username*

6. Perancangan Pesan Kesalahan Hapus *Rating*

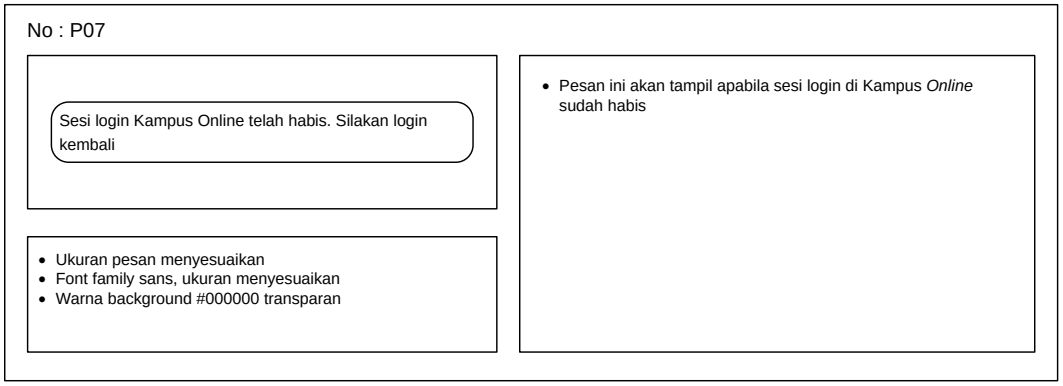
Pesan kesalahan hapus *rating* berisi informasi pengguna terlalu cepat melakukan perubahan data *rating* pada status.. Gambaran umum dari perancangan pesan kesalahan hapus *rating* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.176.



Gambar III.177 Perancangan Pesan Kesalahan Aktivitas Ilegal

7. Perancangan Pesan Kesalahan *Session* Tidak Valid

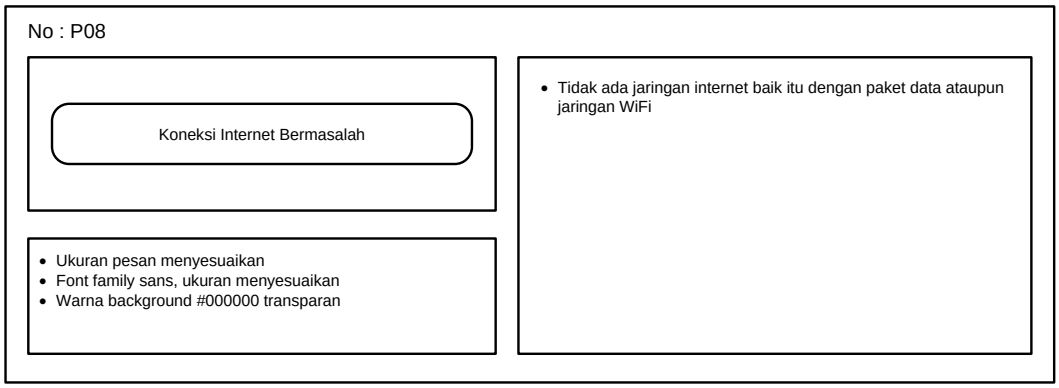
Pesan kesalahan *session* tidak valid berisi informasi bahwa waktu penggunaan Kampus *Online* telah habis. Pesan tersebut muncul ketika pengguna waktu pengguna habis, yaitu dua jam untuk *login* tanpa ingat login dan dua minggu ketika login dengan ingat login. Gambaran umum dari perancangan pesan kesalahan *session* tidak valid pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.178.



Gambar III.178 Perancangan Pesan Kesalahan *Session* Tidak Valid

8. Perancangan Pesan Kesalahan Koneksi Internet

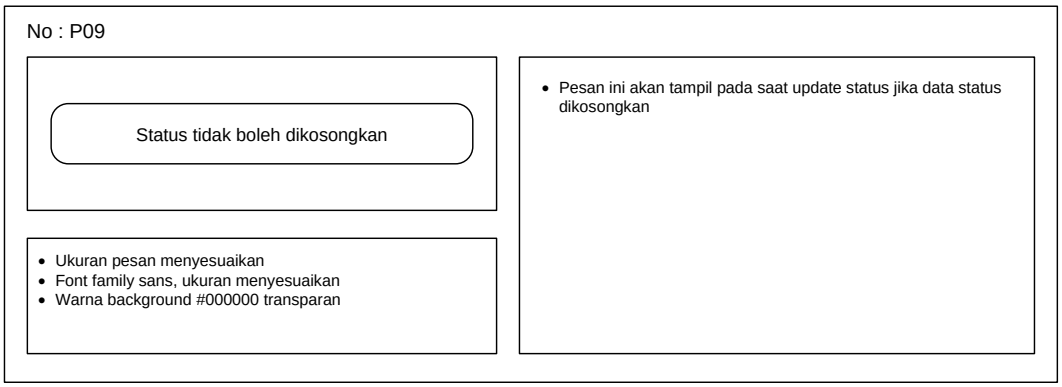
Pesan kesalahan koneksi internet berisi informasi bahwa tidak ada jaringan internet untuk terhubung ke server Kampus *Online* baik itu dengan paket data maupun dengan jaringan wireless. Gambaran umum dari perancangan pesan kesalahan koneksi internet pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.179.



Gambar III.179 Perancangan Pesan Kesalahan Koneksi Internet

9. Perancangan Pesan Kesalahan *Update Status*

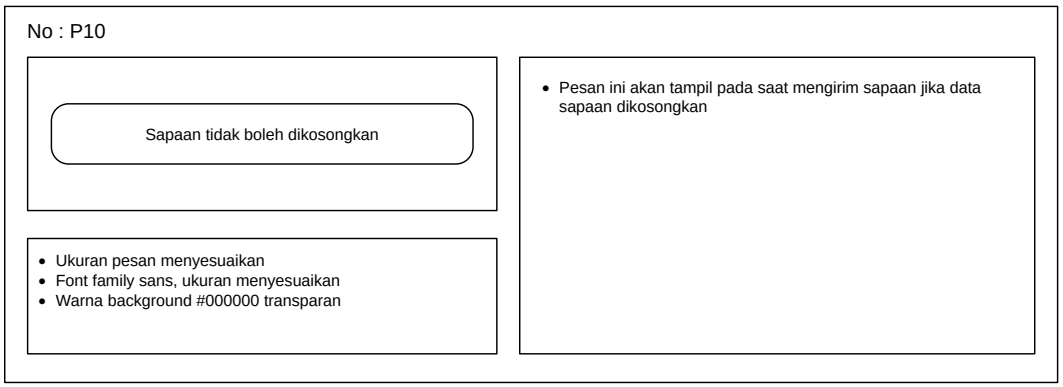
Pesan kesalahan *update status* berisi informasi data status pada proses *update status* masih kosong. Gambaran umum dari perancangan pesan kesalahan *update status* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.180.



Gambar III.180 Perancangan Pesan Kesalahan *Update Status*

10. Perancangan Pesan Kesalahan Kirim Sapaan

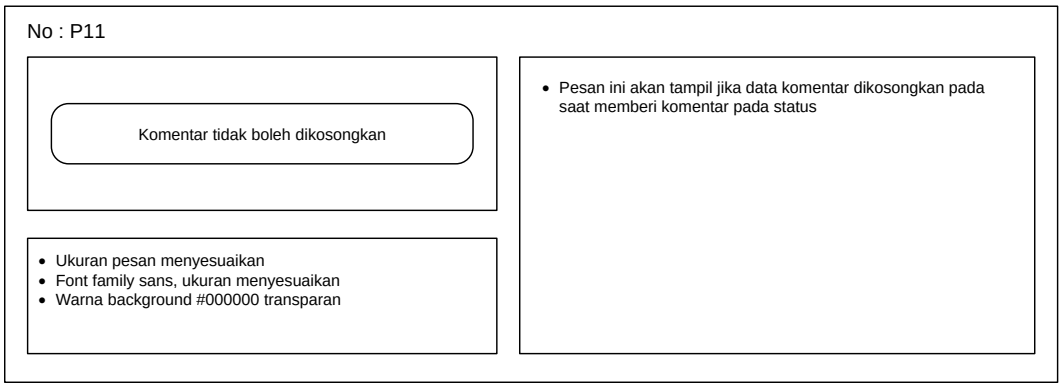
Pesan kesalahan kirim sapaan berisi informasi data sapaan pada proses mengirim sapaan masih kosong. Gambaran umum dari perancangan pesan kesalahan kirim sapaan pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.181.



Gambar III.181 Perancangan Pesan Kesalahan Kirim Sapaan

11. Perancangan Pesan Kesalahan Komentar

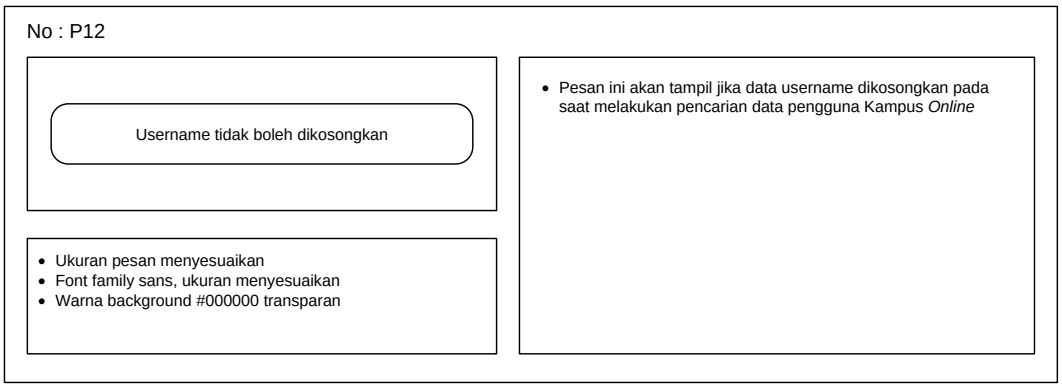
Pesan kesalahan komentar berisi informasi data komentar pada proses penambahan komentar masih kosong. Gambaran umum dari perancangan pesan kesalahan komentar pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.182.



Gambar III.182 Perancangan Pesan Kesalahan Komentar

12. Perancangan Pesan Kesalahan Pencarian

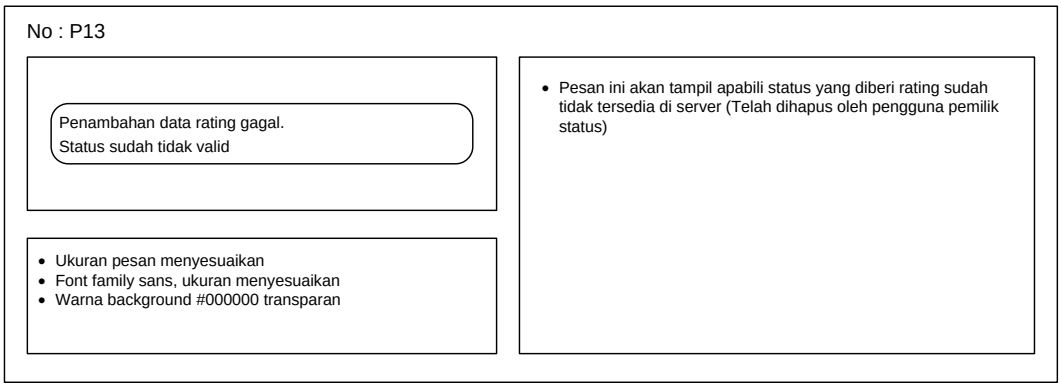
Pesan kesalahan pencarian berisi informasi data *username* pengguna pada proses pencarian masih kosong. Gambaran umum dari perancangan pesan kesalahan pencarian pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.183.



Gambar III.183 Perancangan Pesan Kesalahan Pencarian

13. Perancangan Pesan Kesalahan Status Tidak Valid

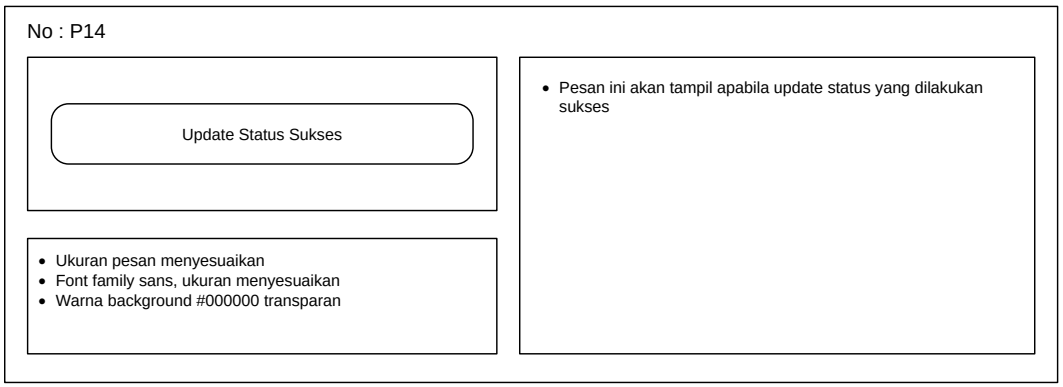
Pesan kesalahan status tidak valid berisi informasi data status yang pengguna pada proses pencarian masih kosong. Gambaran umum dari perancangan pesan kesalahan pencarian pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.184.



Gambar III.184 Perancangan Pesan Kesalahan Status Tidak Valid

14. Perancangan Pesan *Update Status* Sukses

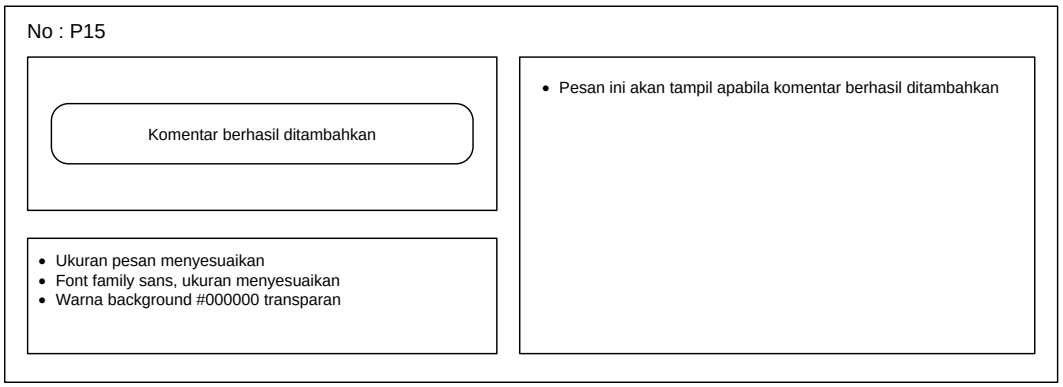
Pesan *update status* sukses berisi informasi *update status* yang dilakukan pengguna telah berhasil. Gambaran umum dari perancangan pesan *update status* sukses pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.185.



Gambar III.185 Perancangan Pesan *Update Status* Sukses

15. Perancangan Pesan Komentar Sukses

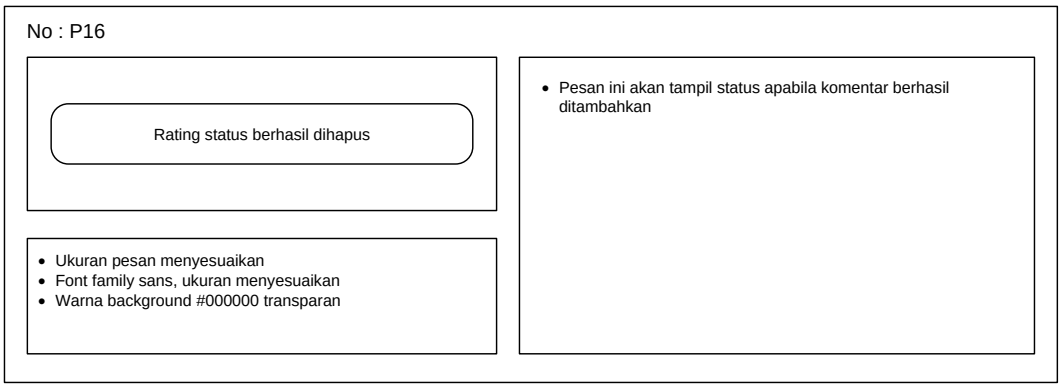
Pesan komentar sukses berisi informasi penambahan data komentar yang dilakukan pengguna telah berhasil. Gambaran umum dari perancangan pesan komentar sukses pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.186.



Gambar III.186 Perancangan Pesan Komentar Sukses

16. Perancangan Pesan Hapus *Rating*

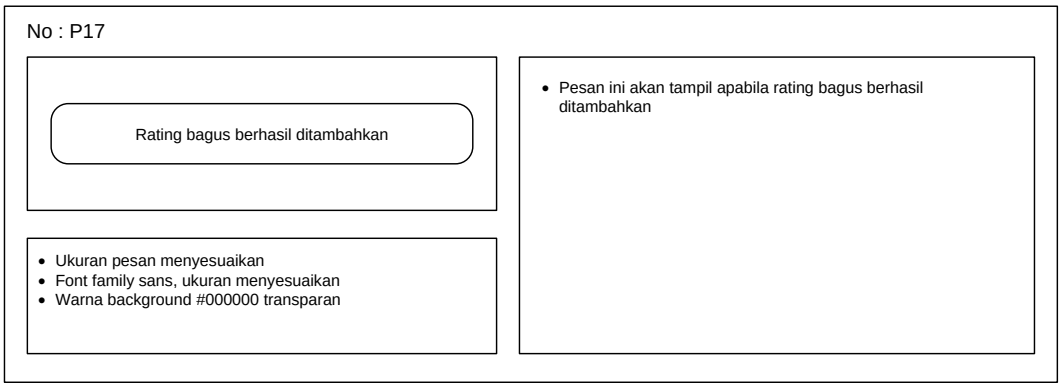
Pesan hapus *rating* berisi informasi penghapusan data *rating* yang sebelumnya ditambahkan telah berhasil dihapus. Gambaran umum dari perancangan pesan hapus *raitng* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.187.



Gambar III.187 Perancangan Pesan Hapus *Rating*

17. Perancangan Pesan Tambah *Rating* Bagus

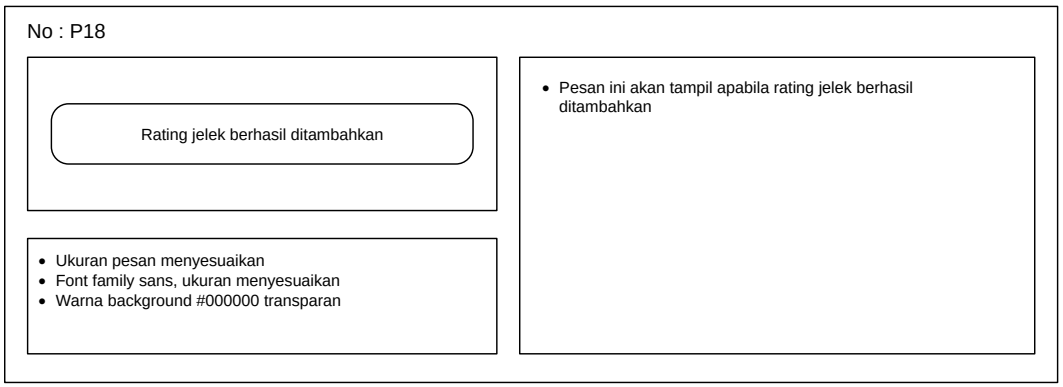
Pesan tambah *rating* bagus berisi informasi penambahan data *rating* bagus pada status tertentu berhasil dilakukan. Gambaran umum dari perancangan pesan tambah *raitng* bagus pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.188.



Gambar III.188 Perancangan Pesan Tambah *Rating* Bagus

18. Perancangan Pesan Tambah *Rating* Jelek

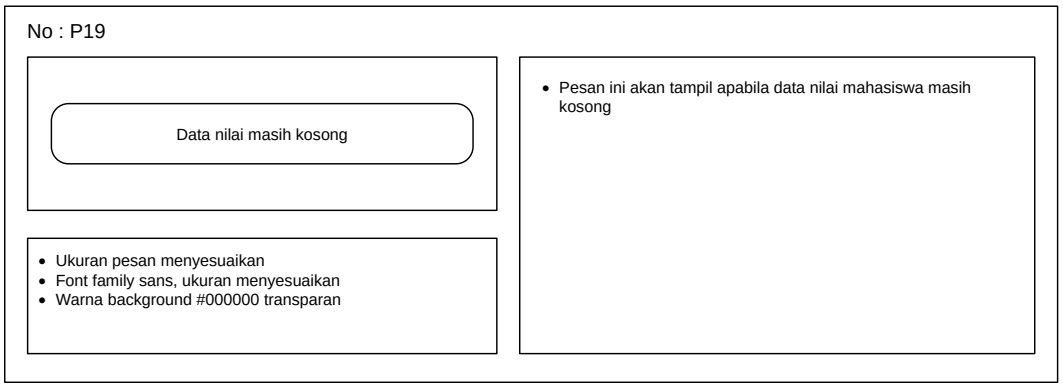
Pesan tambah *rating* jelek berisi informasi penambahan data *rating* jelek pada status tertentu berhasil dilakukan. Gambaran umum dari perancangan pesan tambah *raitng* jelek pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.189.



Gambar III.189 Perancangan Pesan Tambah *Rating* Jelek

19. Perancangan Pesan Nilai Kosong

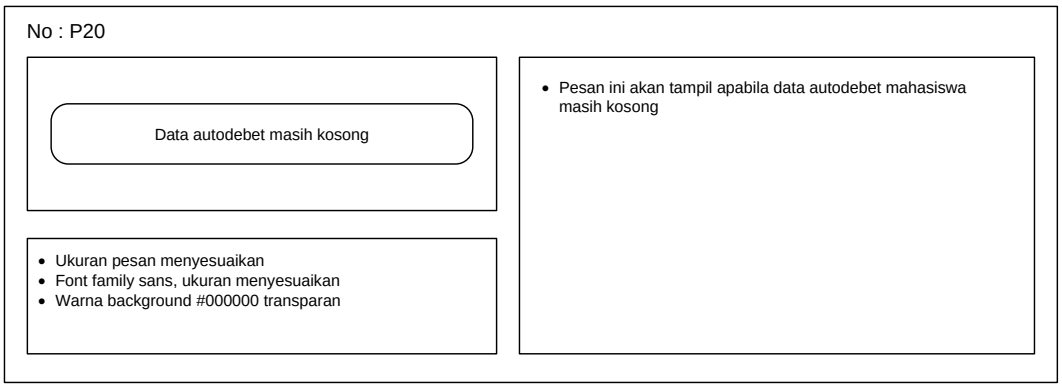
Pesan nilai kosong berisi informasi data nilai mahasiswa masih kosong. Hal ini terjadi pada mahasiswa semester pertama yang belum mendapatkan nilai. Gambaran umum dari perancangan pesan nilai kosong pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.190.



Gambar III.190 Perancangan Pesan Nilai Kosong

20. Perancangan Pesan Autodebet Kosong

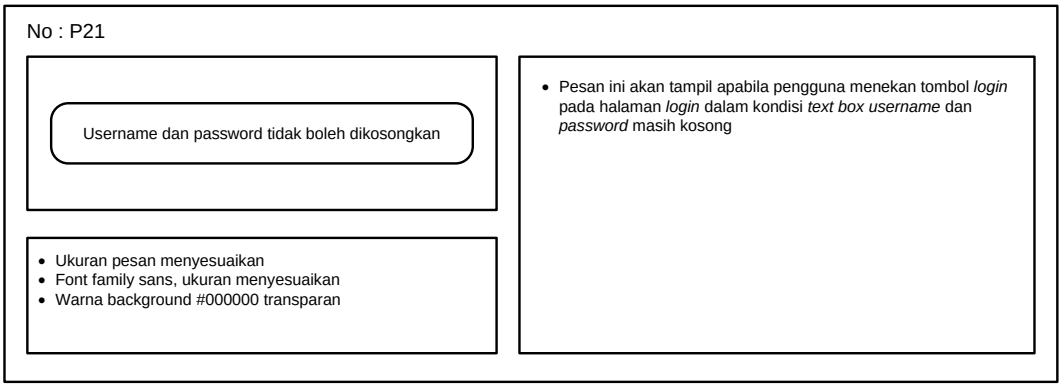
Pesan autodebet kosong berisi informasi data autodebet mahasiswa masih kosong. Gambaran umum dari perancangan pesan autodebet kosong pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.191.



Gambar III.191 Perancangan Pesan Autodebet Kosong

21. Perancangan Pesan *Username* dan *Password Login* Kosong

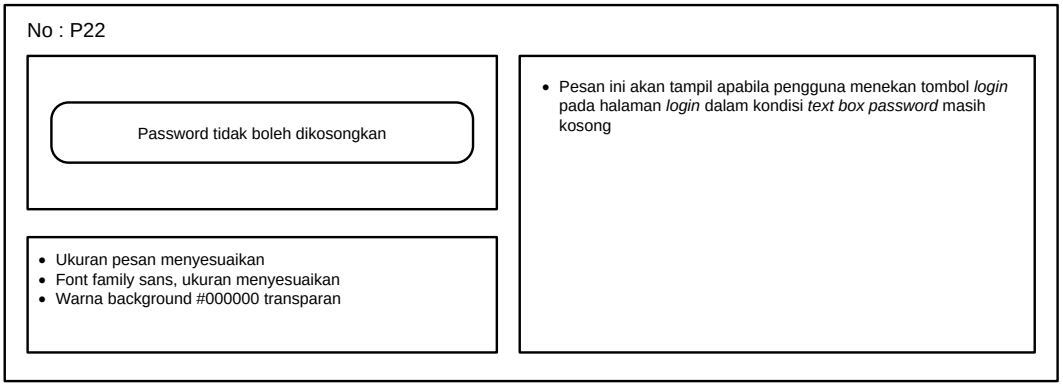
Pesan kesalahan *username* dan *password login* kosong berisi informasi *text box* isian *username* dan *password login* masih kosong. Pesan tersebut muncul pada saat pengguna menekan tombol *login*. Adapun gambaran umum dari perancangan pesan *username* dan *password login* kosong pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.192.



Gambar III.192 Perancangan Pesan *Username Login* Kosong

22. Perancangan Pesan *Password Login* Kosong

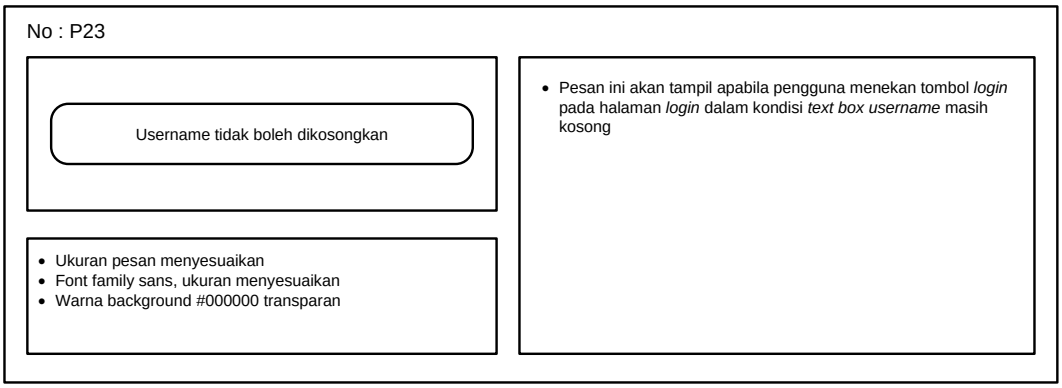
Pesan kesalahan *password login* kosong berisi informasi *text box* isian *password login* masih kosong. Pesan tersebut muncul pada saat pengguna menekan tombol *login*. Adapun gambaran umum dari perancangan pesan *password login* kosong pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.193.



Gambar III.193 Perancangan Pesan *Password Login* Kosong

23. Perancangan Pesan *Username Login* Kosong

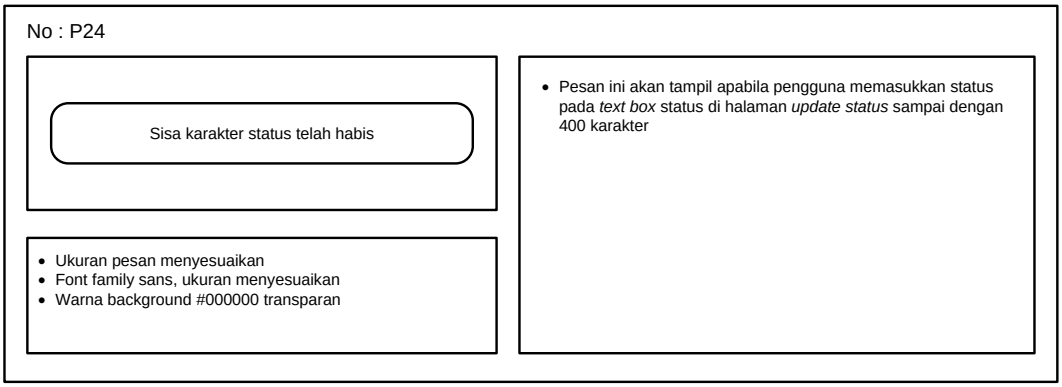
Pesan kesalahan *username login* kosong berisi informasi *text box* isian *username login* masih kosong. Pesan tersebut muncul pada saat pengguna menekan tombol *login*. Adapun gambaran umum dari perancangan pesan *username login* kosong pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.194.



Gambar III.194 Perancangan Pesan *Username Login* Kosong

24. Perancangan Pesan Karakter Status Habis

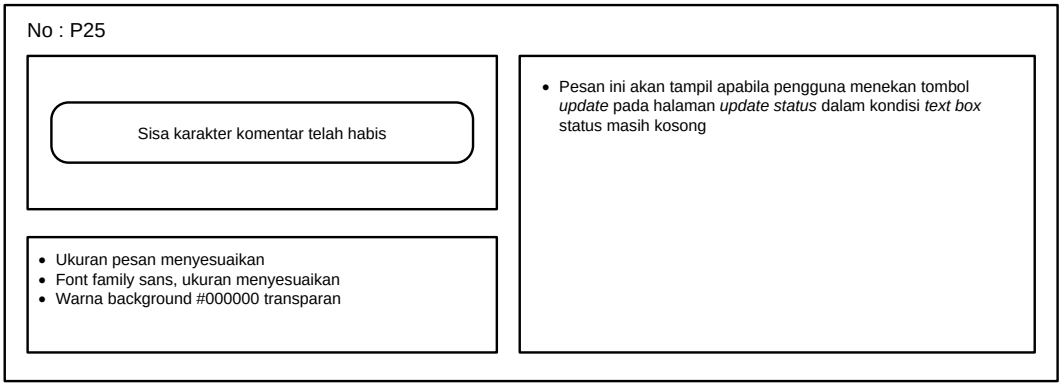
Pesan karakter status habis berisi informasi sisa karakter pada *text box* status telah sampai pada maksimum karakter, yaitu 400 karakter. Pesan tersebut muncul pada saat karakter status berjumlah 400 karakter. Adapun gambaran umum dari perancangan pesan karakter status habis pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.195.



Gambar III.195 Perancangan Pesan Karakter Status Habis

25. Perancangan Pesan Karakter Komentar Habis

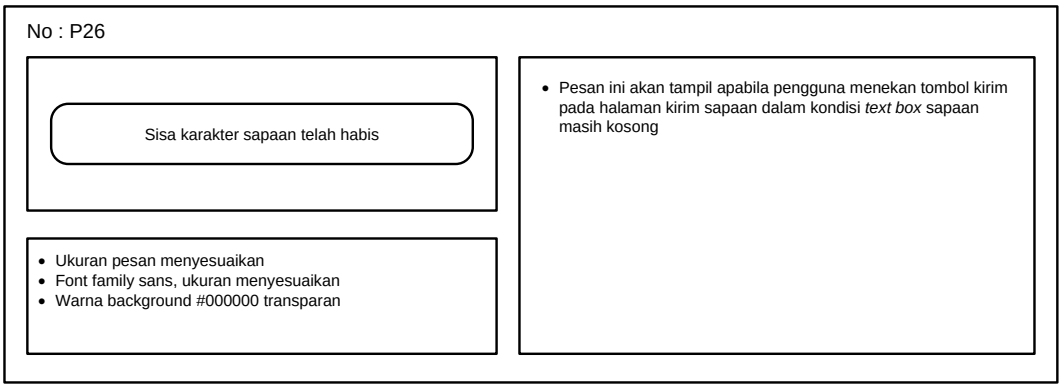
Pesan karakter komentar habis berisi informasi sisa karakter pada *text box* komentar telah sampai pada maksimum karakter, yaitu 400 karakter. Pesan tersebut muncul pada saat karakter komentar berjumlah 400 karakter. Adapun gambaran umum dari perancangan pesan karakter komentar habis pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.196.



Gambar III.196 Perancangan Pesan Karakter Komentar Habis

26. Perancangan Pesan Karakter Sapaan Habis

Pesan karakter sapaan habis berisi informasi sisa karakter pada *text box* sapaan telah sampai pada maksimum karakter, yaitu 400 karakter. Pesan tersebut muncul pada saat karakter sapaan berjumlah 400 karakter. Adapun gambaran umum dari perancangan pesan karakter sapaan habis pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.197.



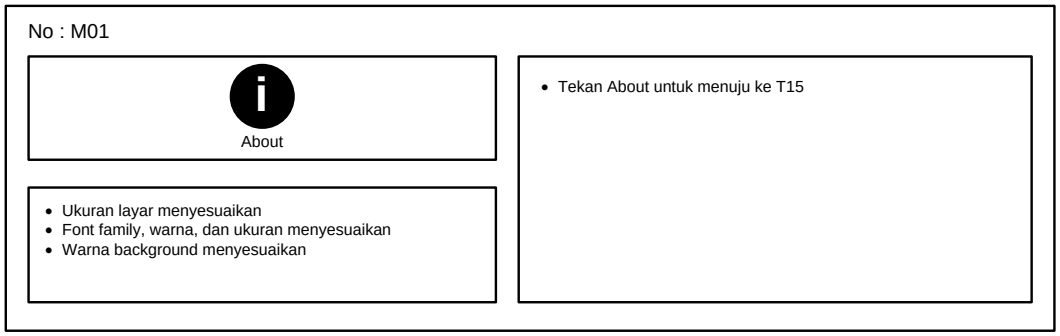
Gambar III.197 Perancangan Pesan Karakter Sapaan Habis

3. 2. 4. 3. **Perancangan Menu**

Perancangan menu mendeskripsikan rencana tampilan dari setiap menu yang digunakan pada aplikasi *mobile* Kampus *Online*. Menu yang dimaksud adalah menu yang ditekan oleh pengguna pada *handset* Android. Berikut ini gambaran dari perancangan menu pada aplikasi *mobile* Kampus *Online* seperti terlihat pada Gambar III.198 sampai dengan Gambar III.202.

1. Perancangan Menu *Login*

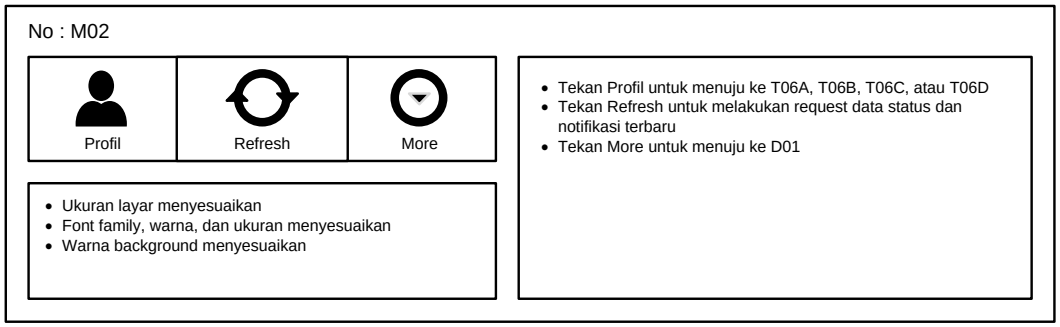
Pesan menu *login* merupakan menu yang muncul ketika pengguna menekan tombol menu pada halaman *login* aplikasi *mobile* Kampus *Online*. Adapun gambaran umum dari perancangan menu *login* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.198.



Gambar III.198 Perancangan Menu *Login*

2. Perancangan Menu *Home*

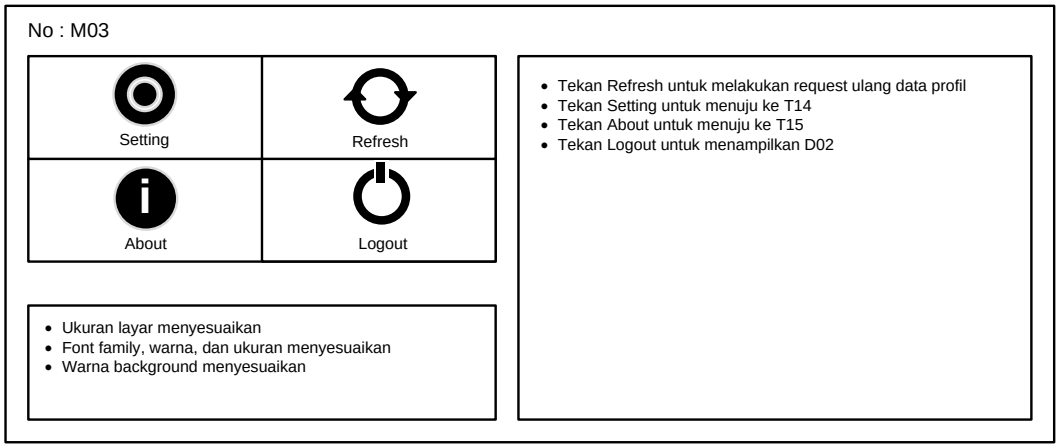
Pesan menu *home* merupakan menu yang muncul ketika pengguna menekan tombol menu pada halaman *home* aplikasi *mobile* Kampus *Online*. Adapun gambaran umum dari perancangan menu *home* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.199.



Gambar III.199 Perancangan Menu *Home*

3. Perancangan Menu Profil

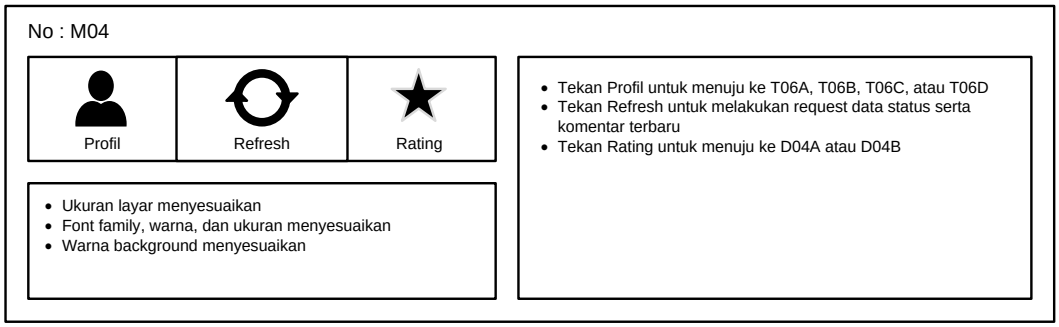
Pesan menu profil merupakan menu yang muncul ketika pengguna menekan tombol menu pada halaman profil aplikasi *mobile* Kampus *Online*. Adapun gambaran umum dari perancangan menu profil pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.200.



Gambar III.200 Perancangan Menu Profil

4. Perancangan Menu Detail Status

Pesan menu detail status merupakan menu yang muncul ketika pengguna menekan tombol menu pada halaman detail status aplikasi *mobile* Kampus *Online*. Adapun gambaran umum dari perancangan menu detail status pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.201.

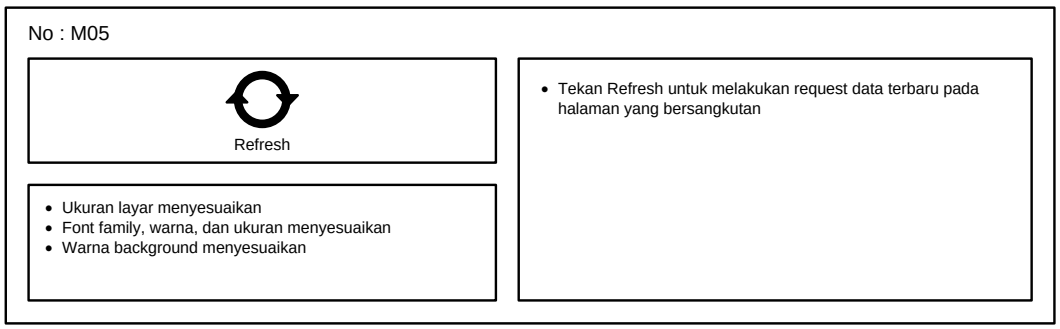


Gambar III.201 Perancangan Menu Detail Status

5. Perancangan Menu *Refresh*

Pesan menu *refresh* merupakan menu yang muncul ketika pengguna menekan tombol menu pada halaman setiap halaman kecuali halaman *login*, *home*, profil, *setting*, dan detail status aplikasi *mobile* Kampus *Online*. Adapun

gambaran umum dari perancangan menu *refresh* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.202.



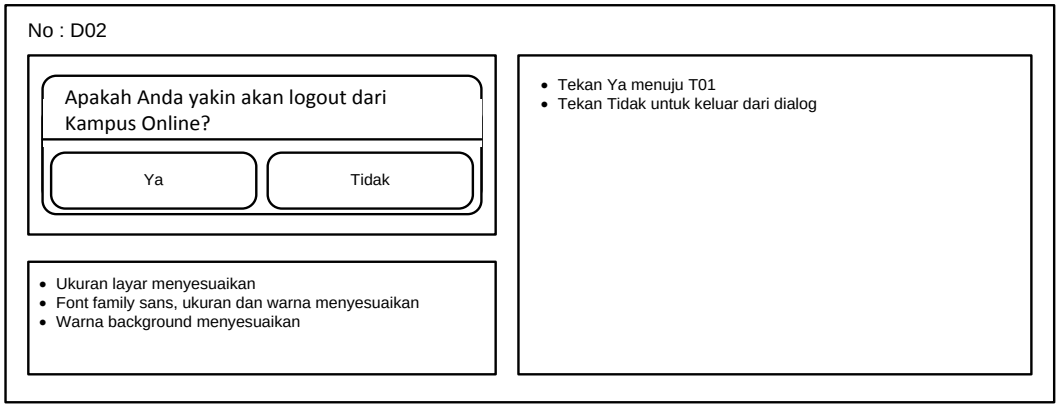
Gambar III.202 Perancangan Menu *Refresh*

3. 2. 4. 4. **Perancangan Dialog**

Perancangan dialog mendeskripsikan rencana tampilan dari setiap dialog yang digunakan pada aplikasi *mobile* Kampus *Online*. Berikut ini gambaran dari perancangan dialog pada aplikasi *mobile* Kampus *Online* seperti terlihat pada Gambar III.203 sampai dengan Gambar III.207.

1. Perancangan Dialog *Logout*

Pesan dialog *logout* merupakan dialog yang muncul ketika pengguna menekan menu *logout* pada halaman profil dan item menu *logout* pada dialog *more* di halaman *home* aplikasi *mobile* Kampus *Online*. Adapun gambaran umum dari perancangan menu *logout* pada aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.203.

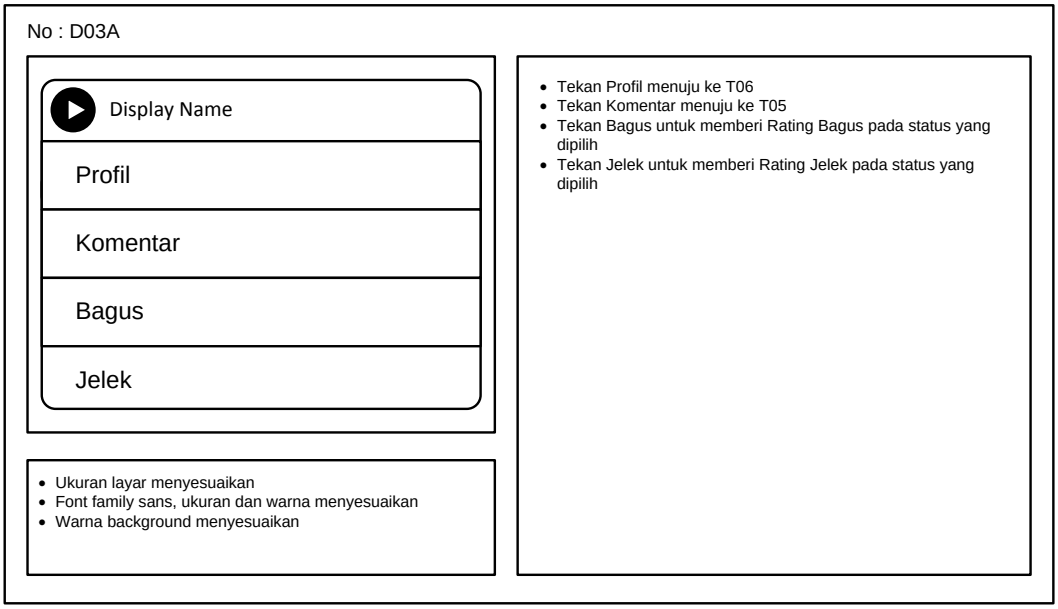


Gambar III.203 Perancangan Dialog *Logout*

2. Perancangan Dialog Status

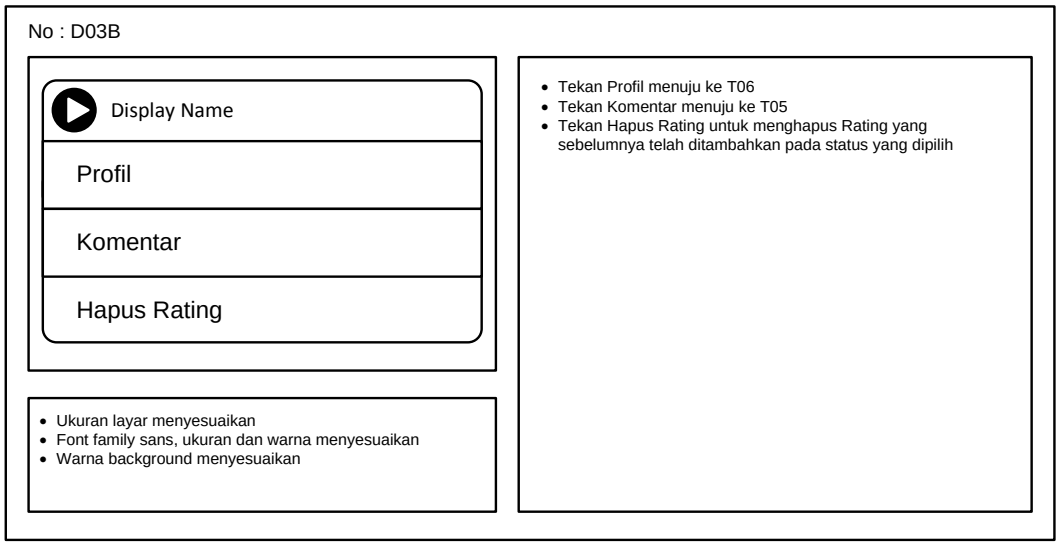
Pesan dialog status merupakan dialog yang muncul ketika pengguna menekan secara lama item status di halaman *home* aplikasi *mobile* Kampus *Online*. Dialog status memiliki dua tipe dialog, yaitu:

a. Dialog yang berfungsi untuk melihat profil pengguna dari status tersebut, menambahkan komentar pada status, dan menambahkan *rating* bagus atau jelek. Berikut gambaran umum dari perancangan dialog status tipe pertama pada aplikasi *mobile* Kampus *Online* yang dapat dilihat pada Gambar III.204.



Gambar III.204 Perancangan Dialog Status (Tipe 1)

b. Dialog yang berfungsi untuk melihat profil pengguna dari status tersebut, menambahkan komentar pada status, dan menghapus *rating* yang sebelumnya telah ditambahkan. Berikut gambaran umum dari perancangan dialog status tipe kedua pada aplikasi *mobile* Kampus *Online* yang dapat dilihat pada Gambar III.205.

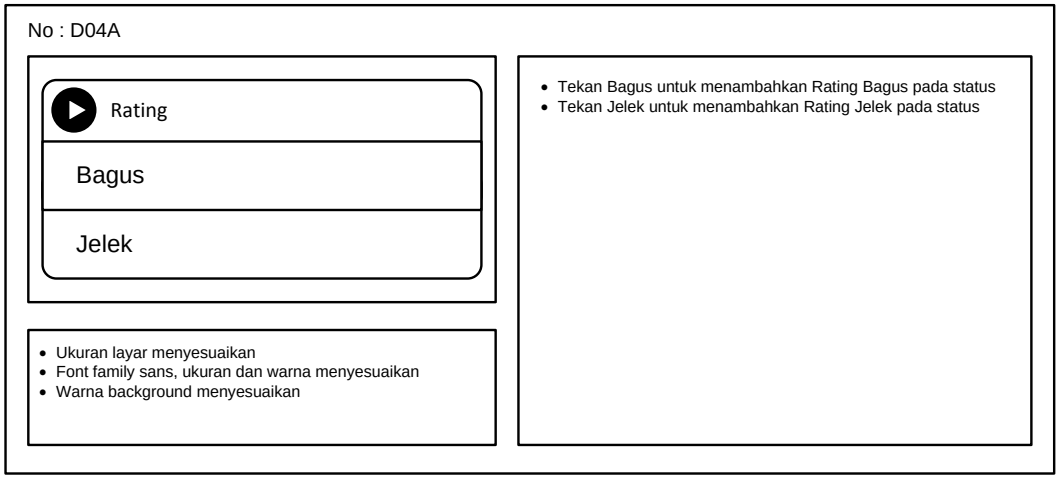


Gambar III.205 Perancangan Dialog Status (Tipe 2)

3. Perancangan Dialog Detail Status

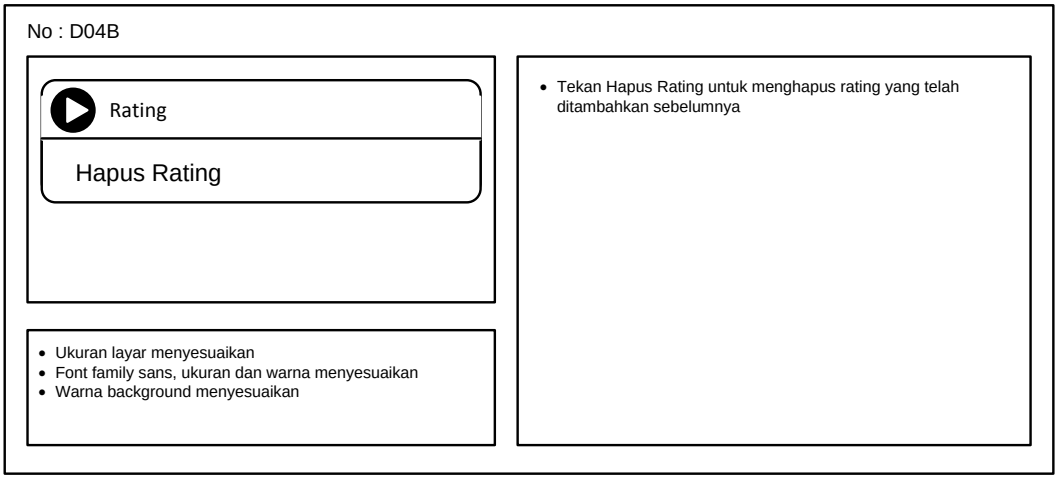
Pesan dialog detail status merupakan dialog yang muncul ketika pengguna menekan menu item rating di halaman detail status aplikasi *mobile* Kampus *Online*. Dialog detail status memiliki dua tipe dialog, yaitu:

a. Dialog yang berfungsi untuk menambah data *rating* pada status di halaman detail status. Berikut gambaran umum dari perancangan dialog detail status tipe pertama pada aplikasi *mobile* Kampus *Online* yang dapat dilihat pada Gambar III.206.



Gambar III.206 Perancangan Dialog Detail Status (Tipe 1)

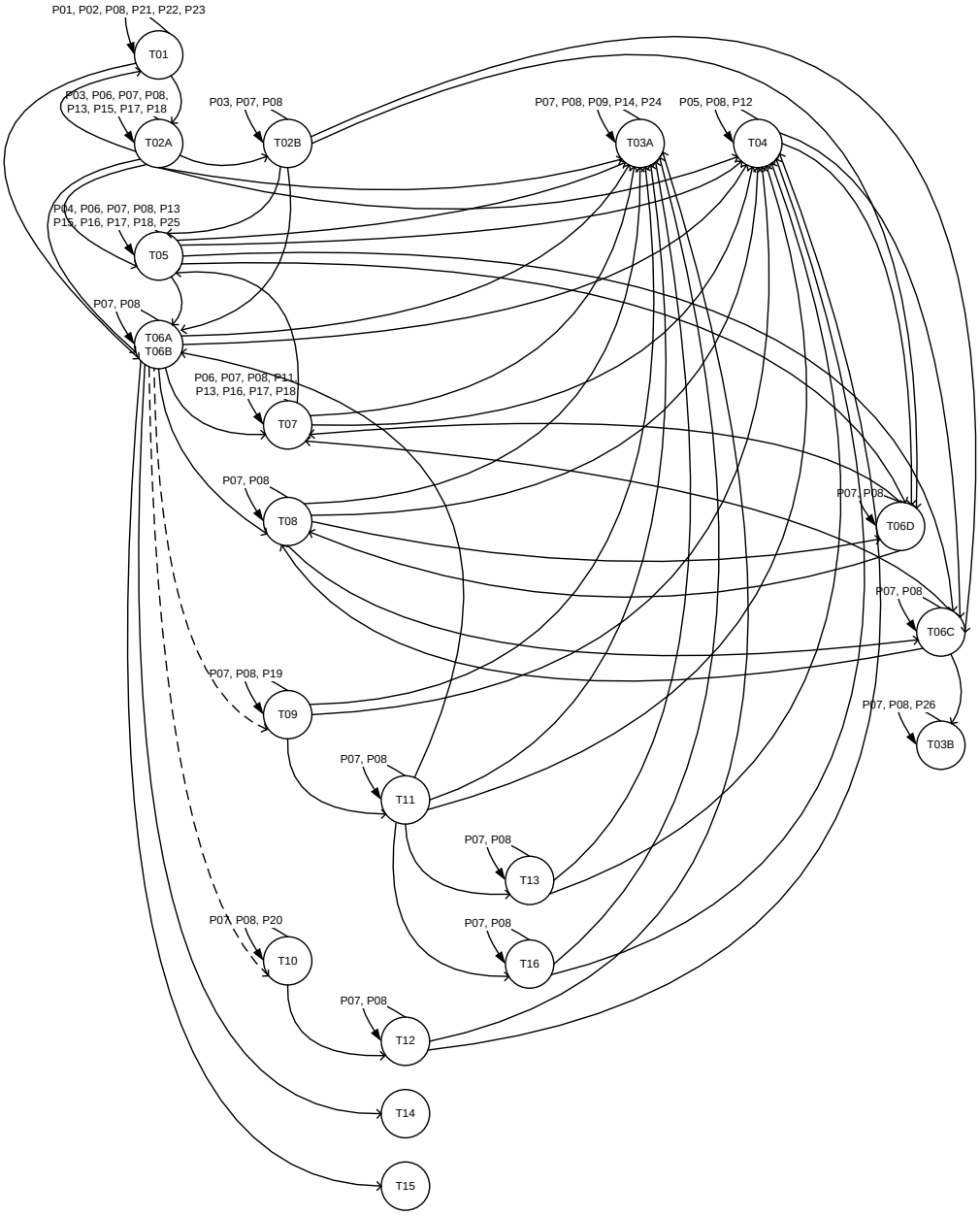
b. Dialog yang berfungsi untuk menghapus data *rating* pada status di halaman detail status. Berikut gambaran umum dari perancangan dialog detail status tipe kedua pada aplikasi *mobile* Kampus *Online* yang dapat dilihat pada Gambar III.207.



Gambar III.207 Perancangan Dialog Detail Status (Tipe 2)

3. 2. 5. Jaringan Semantik

Jaringan Semantik merupakan gambaran pengetahuan grafis yang menunjukkan hubungan antar objek (lingkaran) dan garis yang menggambarkan formasi antarobjek tersebut. Adapun jaringan semantik untuk aplikasi *mobile* Kampus *Online* dapat dilihat pada Gambar III.208.



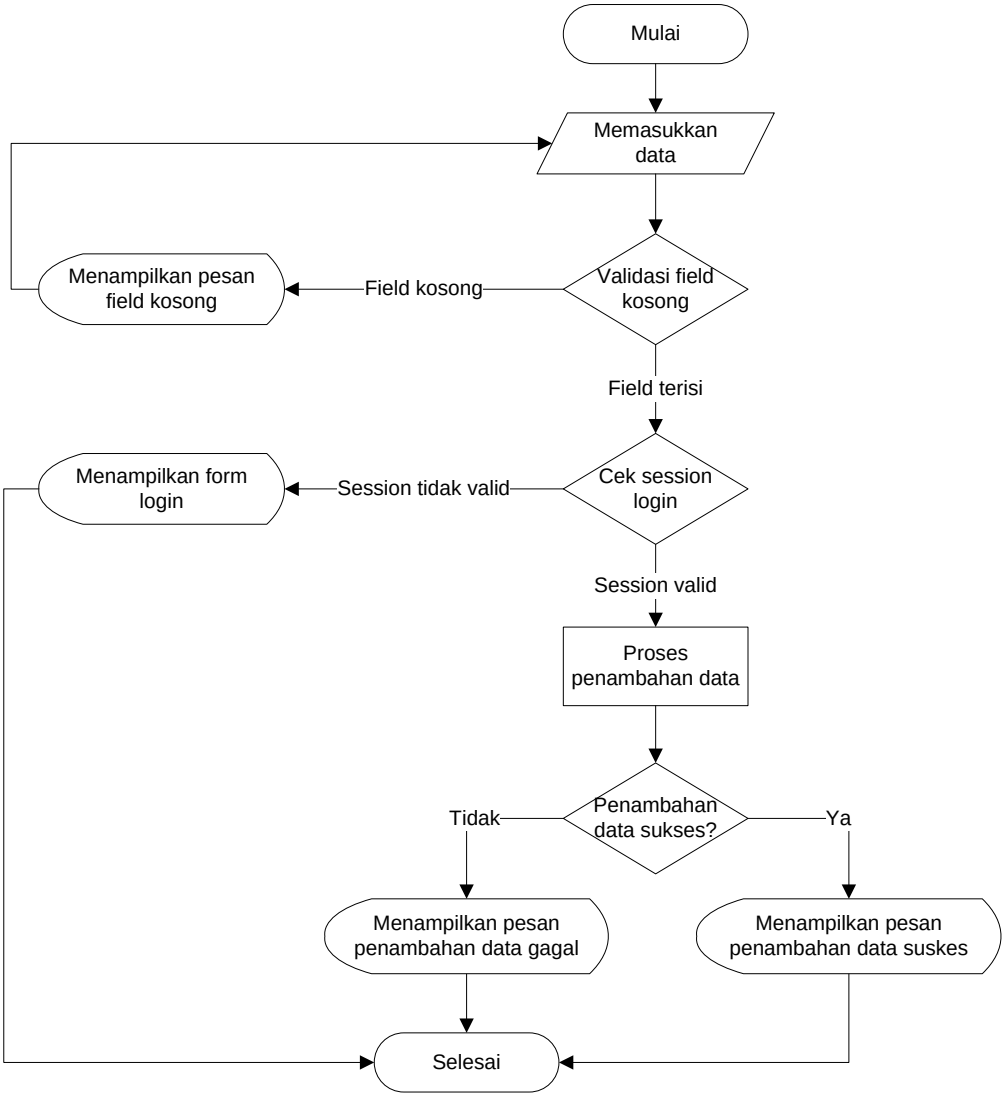
Gambar III.208 Jaringan Semantik Aplikasi *Mobile* Kampus *Online*

3. 2. 6. Perancangan Prosedural

Perancangan prosedural menggambarkan algoritma dari prosedur-prosedur yang terdapat pada perangkat lunak yang dibangun. Adapun perancangan prosedural pada aplikasi *mobile* Kampus *Online* digambarkan dengan menggunakan *flowchart* yang terdiri dari *flowchart* penambahan data, *flowchart* perubahan data, *flowchart* pencarian data, dan *flowchart* penghapusan data. Berikut ini penjelasan dari masing-masing *flowchart* perancangan prosedural aplikasi *mobile* Kampus *Online* yang terlihat pada Gambar III.209 sampai dengan Gambar III.212.

1. *Flowchart* Penambahan Data

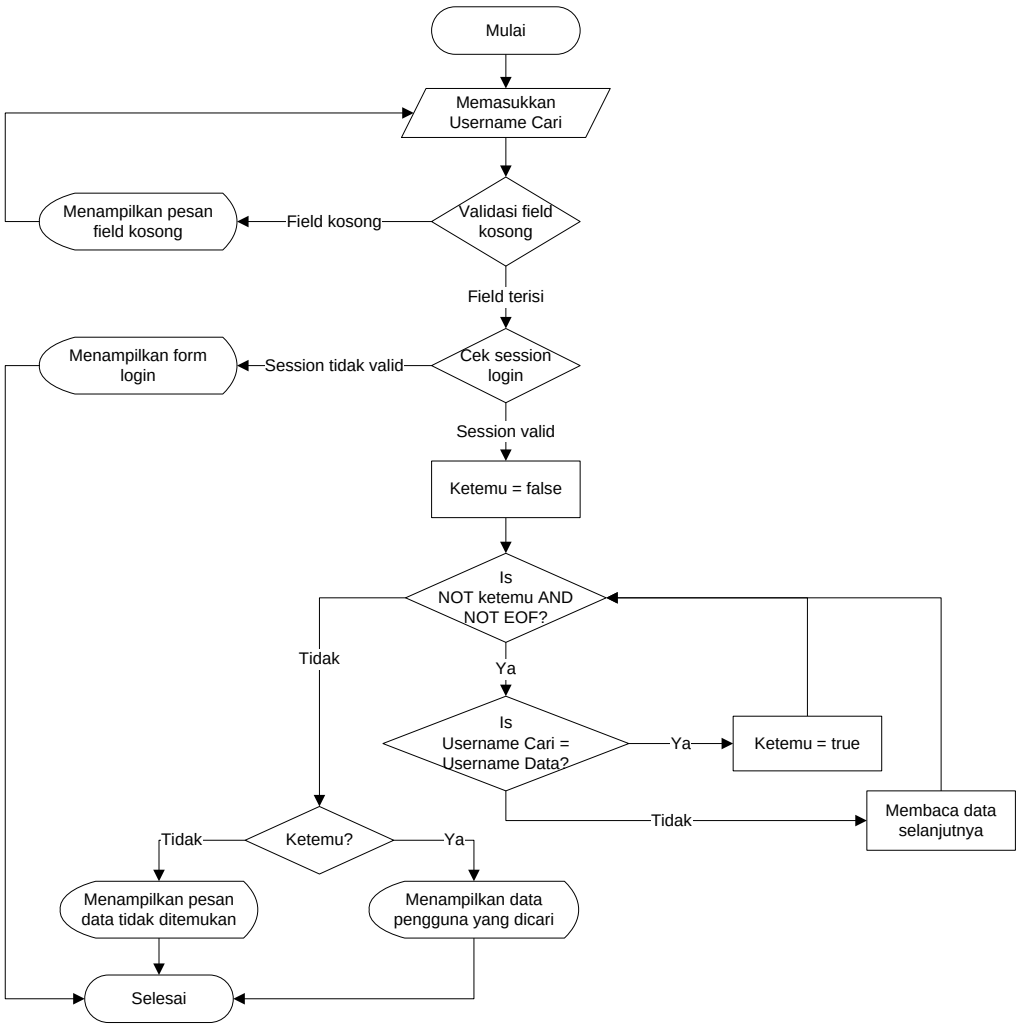
Flowchart penambahan data menggambarkan langkah-langkah yang dilakukan oleh sistem dalam melakukan penambahan data pada aplikasi *mobile* Kampus *Online*. Adapun proses-proses penambahan data pada aplikasi *mobile* Kampus *Online* yang diwakili oleh *flowchart* penambahan data, yaitu pada proses penambahan data status, data sapaan, dan data *rating*. Berikut ini gambaran *flowchart* dari proses penambahan data yang dapat dilihat pada Gambar III.209.



Gambar III.209 *Flowchart* Penambahan Data

2. *Flowchart* Pencarian Data

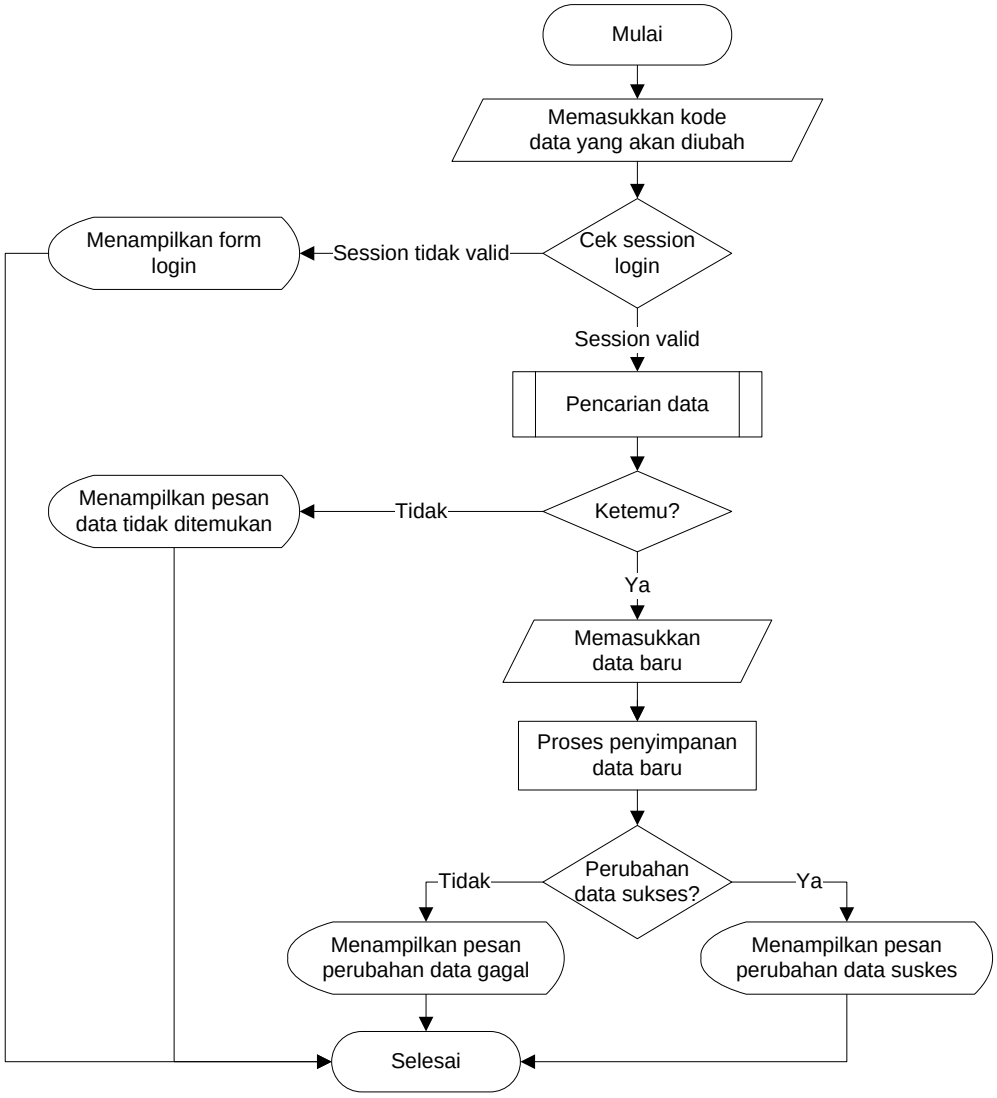
Flowchart pencarian data menggambarkan langkah-langkah yang dilakukan oleh sistem dalam melakukan pencarian data pada aplikasi *mobile* Kampus *Online*. Adapun proses pencarian data pada aplikasi *mobile* Kampus *Online* yang diwakili oleh *flowchart* pencarian data, yaitu pada proses pencarian data pengguna. Berikut ini gambaran *flowchart* dari proses pencarian data yang dapat dilihat pada Gambar III.210.



Gambar III.210 *Flowchart* Pencarian Data

3. *Flowchart* Perubahan Data

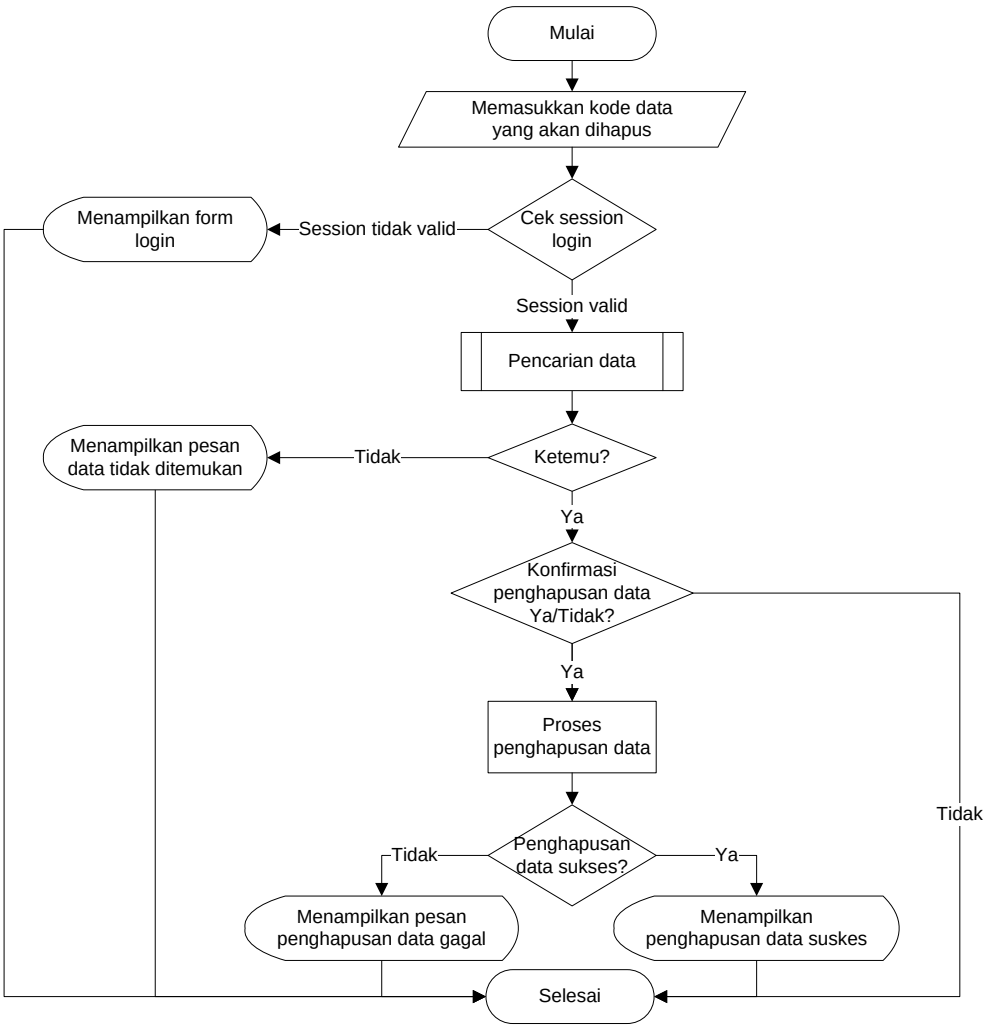
Flowchart perubahan data menggambarkan langkah-langkah yang dilakukan oleh sistem dalam melakukan perubahan data pada aplikasi *mobile* Kampus *Online*. Adapun proses perubahan data pada aplikasi *mobile* Kampus *Online* yang diwakili oleh *flowchart* perubahan data, yaitu pada proses perubahan status notifikasi dari belum dibaca menjadi telah dibaca. Berikut ini gambaran *flowchart* dari proses perubahan data yang dapat dilihat pada Gambar III.211.



Gambar III.211 *Flowchart* Perubahan Data

4. *Flowchart* Penghapusan Data

Flowchart penghapusan data menggambarkan langkah-langkah yang dilakukan oleh sistem dalam melakukan penghapusan data pada aplikasi *mobile* Kampus *Online*. Adapun proses penghapusan data pada aplikasi *mobile* Kampus *Online* yang diwakili oleh *flowchart* penghapusan data, yaitu pada proses penghapusan data *rating*. Berikut ini gambaran *flowchart* dari proses penghapusan data seperti terlihat pada Gambar III.212.



Gambar III.212 *Flowchart* Penghapusan Data