BAB III
ANALISIS DAN PERANCANGAN SISTEM

Analisis dan perancangan sistem memerlukan tahapan yang sistematis untuk mendapatkan aplikasi yang baik dan bersesuaian dengan kegunaan dan tujuannya. Tahap awal dari analisis adalah menganalisis kebutuhan-kebutuhan sistem mulai dari kebutuhan pengguna, kebutuhan non fungsional, dan kebutuhan fungsional. Sedangkan untuk tahap perancangan aplikasi yaitu perancangan *Database,* dan perancangan antarmuka.

III.1 Analisis Sistem

Analisis sistem (*system analysis*) dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasikan permasalah-permasalahan, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikan. Analisis dapat juga diartikan memahami sistem pemikiran yang kompleks dengan memecahnya ke dalam unsur-unsur yang lebih sederhana sehingga hubungan antar unsur-unsur itu menjadi jelas.

III.1.1 Analisis Masalah

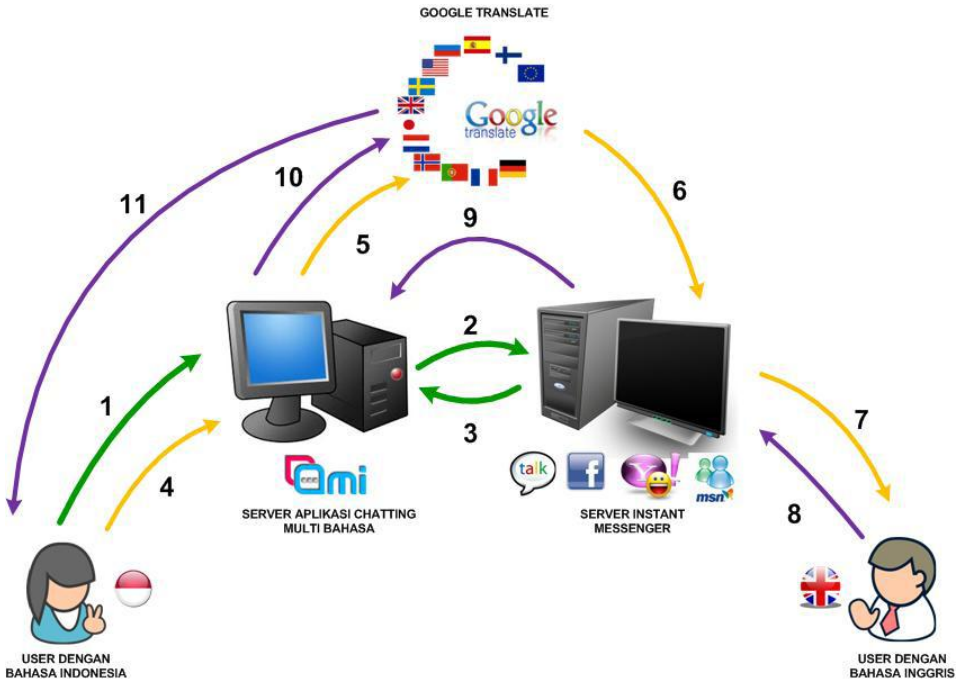
Sulitnya para pengguna aplikasi *chatting* saat berhadapan dengan orang asing dan menggunakan bahasa yang tidak mereka kuasai masih menjadi kendala. Mereka harus menerjemahkan dulu setiap pesan yang diterima maupun pesan yang akan dikirim ke dalam bahasa yang digunakan oleh lawan bicara menggunakan kamus ataupun aplikasi penerjemah. Hal ini tentu sangat merepotkan dan menghambat dalam berkomunikasi, terlebih lagi jika lawan bicara lebih dari satu, seperti *chat conference* dan menggunakan bahasa yang berbeda antara satu dengan yang lainnya. Oleh karena itu dibutuhkan suatu sistem yang dapat mengatasi kendala tersebut.

Pada penelitian sebelumnya telah dibangun sebuah aplikasi *chatting* yang dilengkapi dengan fasilitas penerjemah berbasis *web*, dengan aplikasi ini, pengguna dapat berinteraksi dengan pengguna lain menggunakan bahasa yang berbeda, misalnya pengguna dengan bahasa Indonesia berinteraksi dengan pengguna lain yang menggunakan bahasa Inggris. Pesan teks yang dikirimkan dalam bahasa Indonesia akan diterjemahkan secara otomatis ke dalam Bahasa Inggris dan sebaliknya[15]. Aplikasi ini berbasis *web*, dimana memiliki keterbatasan dalam pengaksesannya karena harus di akses dari komputer yang terhubung ke internet. Pengguna juga harus men-*setting* terlebih dahulu bahasa yang digunakan oleh lawan bicara setiap *login*, sehingga akan menyulitkan pengguna, terlebih jika pengguna belum mengetahui bahasa apa yang di gunakan oleh lawan bicara. Penggunaan aplikasi ini hanya sebatas pada *private chatting*, sehingga pengguna akan merasa kesulitan jika akan membuat suatu *chat conference*, dan pada aplikasi ini juga belum tersedia fitur riwayat percakapan yang merupakan fitur dasar dari sebuah aplikasi *chatting*.

Berdasarkan uraian yang dikemukakan, maka terdapat masalah bagaimana membangun aplikasi *chatting* dengan menggunakan penerjemah bahasa secara otomatis berbasis *mobile*.

III.1.1.1 Analisis Prosedur yang Sedang Berjalan

Berdasarkan hasil penelitian, terdapat beberapa prosedur yang sedang berjalan pada aplikasi *chatting* multi bahasa sebagai berikut:



Gambar III.1 Alur Sistem Yang Sedang Berjalan

Adapun penjelasan dari gambaran umum di atas sebagai berikut :

1. Pengguna dengan bahasa Indonesia melakukan *login* sesuai dengan *instant messenger* yang dipilih dan data *login* tersebut (*username* dan *password*) divalidasi oleh *server* aplikasi *chatting* multi bahasa (AMI).
2. Data *login* yang telah divalidasi oleh *server* AMI akan diotentifikasi oleh *server instant messenger*.
3. Ketika data *login* berhasil diotentifikasi oleh *server instant messenger* maka *server instant messenger* akan memberikan data-data yang berkaitan dengan *user*, diantaranya daftar teman dan status teman. Setelah dinyatakan berhasil *login*, maka *server* AMI akan menampilkan form *chatting*.
4. Pengguna dengan bahasa Indonesia mengirimkan pesan dalam bahasa Indonesia dan menetapkan bahasa Inggris (sesuai dengan bahasa yang digunakan lawan bicara) untuk menerjemahkan pesan yang akan dikirim.

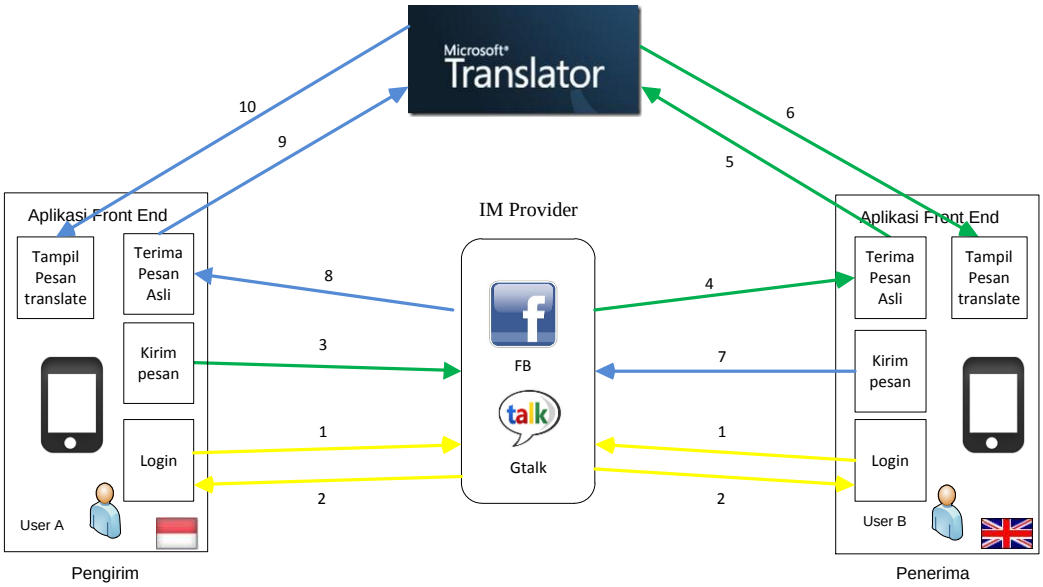
5. Pesan dalam bahasa Indonesia diterjemahkan ke dalam bahasa Inggris oleh *Google Translate*.
6. Pesan yang telah diterjemahkan dikirimkan ke *server instant messenger*.
7. Pesan dalam bahasa Inggris tersebut dikirimkan oleh *server instant messenger* ke pihak lawan bicara.
8. Pihak lawan bicara mengirimkan pesan dalam bahasa Inggris yang kemudian diterima oleh *server instant messenger*.
9. Pesan dalam bahasa Inggris tersebut dikirimkan ke *server* AMI.
10. Oleh *server* AMI, pesan tersebut dikirimkan ke *Google Translate* untuk diterjemahkan kembali ke dalam bahasa Indonesia.
11. Setelah diterjemahkan, pesan dalam bahasa Indonesia tersebut akan dikirimkan ke pengguna.

III.1.2 Analisis Sistem Yang Akan Dibangun

III.1.2.1 Gambaran Umum Sistem yang Akan Dibangun

Sistem yang akan dibangun adalah suatu aplikasi *mobile* menggunakan *platform* Android. Sistem ini terdiri dari dua komponen utama, yaitu aplikasi *frontend* yang berjalan pada pihak *client* dan aplikasi *backend* yang berjalan pada *server*. Pada sistem ini, aplikasi *frontend* merupakan komponen yang aktif, dimana *user* dapat mengirim dan menerima pesan sedangkan aplikasi *backend* bertindak sebagai penyimpanan data pengguna.

Aplikasi *frontend* berfungsi sebagai *client* yang menghubungkan ke *server* yang dioperasikan oleh vendor jaringan IM (*Instant Messenger*), seperti *Google Talk* dan *Facebook chat*. Aplikasi *frontend* ini juga berfungsi untuk menghubungkan ke *server Microsoft Translator* untuk menerjemahkan pesan secara otomatis. Aplikasi *backend* merupakan aplikasi yang berfungsi untuk memonitoring sistem yang sedang berjalan dan menghapus data pengguna. Gambar III.2 menunjukan gambaran umum tentang sistem yang akan dibangun.



Gambar III.2 Alur Kerja Sistem Yang Akan Dibangun

Berikut skenario alur kerja sistem yang akan dibangun :

1. *User* A dan *User* B adalah pengguna aplikasi. Pertama *User* A dan *User* B *login* terlebih dahulu menggunakan salah satu akun IM yang dimiliki dengan cara menginputkan *email* dan *password* untuk akun *Gmail* dan *username* dan *password* untuk akun *facebook.*
2. Ketika data *login* berhasil diotentikasi oleh *server* IM, maka *sever* IM akan mengirimkan data-data yang berkaitan dengan akun *user* diantaranya daftar teman dan status. *User* dapat memilih daftar teman yang sedang *online* untuk memulai *chatting*.
3. *User* A sebagai pengirim dan *User* B sebagai penerima. *User* A mengirim pesan teks ke *User* B menggunakan bahasa Indonesia
4. Kemudian *User* B menerima pesan teks dari *user* A masih dalam bahasa Indonesia.
5. Sistem akan mengecek kode bahasa yang digunakan oleh *user* B, kemudian pesan dan kode bahasa Inggris akan dikirimkan *Microsoft Translator* untuk diterjemahkan ke dalam bahasa Inggris.
6. Pesan yang sudah diterjemahkan ke dalam bahasa Inggris akan ditampilkan ke *User* B.

7. *User* B sebagai pengirim dan *User* A sebagai penerima. *User* B mengirim pesan teks ke *User* A menggunakan bahasa Inggris
8. Kemudian *User* A menerima pesan teks dari *user* B masih dalam bahasa Inggris.
9. Sistem akan mengecek kode bahasa yang digunakan oleh *User* A, kemudian pesan dan kode bahasa Indonesia akan dikirimkan *Microsoft Translator* untuk diterjemahkan ke dalam bahasa Indonesia.
10. Pesan yang sudah diterjemahkan ke dalam bahasa Inggris akan ditampilkan ke *User* B.

III.1.2.2 Analisis Otentikasi

Otentikasi / identifikasi adalah pengecekan terhadap identitas suatu entitas, bisa berupa orang, kartu kredit atau mesin. Otentikasi sering melibatkan memverifikasi keabsahan setidaknya satu bentuk identifikasi.

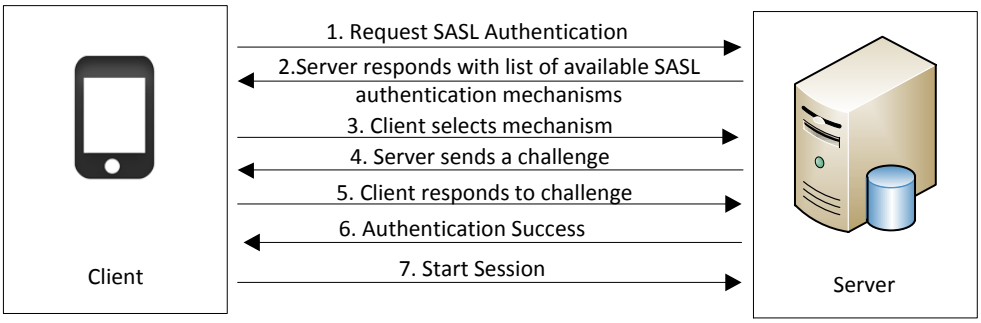
Berikut langkah-langkah proses otentikasi ke *server* yang menggunakan *SASL* :

1. Klien meminta otentikasi *SASL*
2. *Server* merespon dengan daftar mekanisme otentikasi *SASL* tersedia
3. Klien memilih mekanisme
4. *Server* mengirimkan tantangan
5. Klien merespon tantangan
6. *Server* mengirimkan lebih banyak tantangan, klien mengirimkan respon yang lebih

rangkaian pasangan tantangan / tanggapan ini berlanjut sampai salah satu dari tiga hal terjadi:

1. Klien membatalkan koneksi
2. Laporan *server* gagal.
3. Laporan *server* sukses.

Setelah otentikasi selesai, klien mengirimkan sebuah paket untuk memulai sesi.



Gambar III.3 Alur SASL *Authentication*

Contoh dibawah ini menunjukan komunikasi antara *Client* [C] dan *Server* [S].

1. Inisialisasi Aliran

Klien memulai dengan meminta otentikasi *SASL*. *Server* merespon dengan mengirimkan mekanisme otentikasi yang tersedia untuk klien bersama dengan informasi aliran.

Contoh 1 Inisialisasi Aliran

```
C: <stream:stream xmlns='jabber:client'
                  xmlns:stream='http://etherx.jabber.org/streams'
                  xmlns:sasl='http://www.iana.org/assignments/sasl
-mechanisms'
                  to='jabber.org'>
S: <stream:stream xmlns='jabber:client'
                  xmlns:stream='http://etherx.jabber.org/streams'
                  xmlns:sasl='http://www.iana.org/assignments/sasl
-mechanisms'
                  id='12345678'>
   <sasl:mechanisms>
     <sasl:mechanism>PLAIN</sasl:mechanism>
     <sasl:mechanism>DIGEST-MD5</sasl:mechanism>
     <sasl:mechanism>EXTERNAL</sasl:mechanism>
   <sasl:mechanisms>
```

Gambar III.4 Contoh Inisialisasi Aliran

2. Seleksi Mekanisme dan *handshake*

Selanjutnya *client* memilih mekanisme otentikasi :

Contoh 2 memilih mekanisme *plaintext*

```
C: <sasl:auth mechanism='PLAIN'/>
```

Gambar III.5 Contoh Mekanisme Plaintext

Server merespon dengan tantangan mekanisme tertentu, yang klien harus merespon. Lebih dari satu tantangan / tanggapan pasangan dapat berlangsung, ini adalah mekanisme khusus.

Contoh 3 *Plaintext handshake*

```
S: <sasl:challenge/>
C: <sasl:response>cm9iAHNlY3JldA==</sasl:response>
```

Gambar III.6 Contoh *Plaintext Handshake*

Contoh 4 *Digest handshake*

```
S: <sasl:challenge>
     cmVhbG09ImNhdGFjbHlzbS5jeCIsbm9uY2U9Ik9BNk1HOXRFUUdtMmhoIi
     xxb3A9ImF1dGgiLGNoYXJzZXQ9dXRmLTgsYWxnb3JpdGhtPW1kNS1zZXNz
   </sasl:challenge>
C: <sasl:response>
     dXNlcm5hbWU9InJvYiIscmVhbG09ImNhdGFjbHlzbS5jeCIsbm9uY2U9Ik
     9BNk1HOXRFUUdtMmhoIixjbm9uY2U9Ik9BNk1IWGg2VnFUclJrIixuYz0w
     MDAwMDAwMSxxb3A9YXV0aCxkaWdlc3QtdXJpPSJqYWJiZXIvY2F0YWNseX
     NtLmN4IixyZXNwb25zZT1kMzg4ZGFkOTBkNGJiZDc2MGExNTIzMjFmMjE0
     M2FmNyxjaGFyc2V0PXV0Zi04
   </sasl:response>
S: <sasl:challenge>
     cnNwYXV0aD1lYTQwZjYwMzM1YzQyN2I1NTI3Yjg0ZGJhYmNkZmZmZA==
   </sasl:challenge>
C: <sasl:response/>
```

Gambar III.7 Contoh *Digest Handshake*

Untuk mekanisme yang memerlukan klien untuk mengirim data pertama (seperti tantangan pertama dari *server* kosong), secara opsioal klien dapat mengirim *respons* pertama sebagai bagian dari seleksi mekanisme:

Contoh 5 Seleksi mekanisme *plaintext*; *Client* mengirim data perrtama.

```
C: <sasl:auth mechanism='PLAIN'>cm9iAHNlY3JldA==</sasl:auth>
```

Gambar III.8 Contoh Seleksi Mekanisme *Plaintext*

3. Sukses, gagal dan pembatalan dari *client*

Handshake berlanjut sampai otentikasi selesai dengan sukses, otentikasi gagal, atau klien membatalkan *handshake*:

Contoh 6 *Authentication Success*

```
S: <sasl:success/>
```

Gambar III.9 Contoh *Authentication Success*

Contoh 7 *Authentication failure*

```
S: <sasl:failure/>
```

Gambar III.10 Contoh *Authentication Failure*

Contoh 8 *Client Abort*

```
C: <sasl:abort/>
```

Gambar III.11 Contoh *Client Abort*

Opsional, *server* atau klien dapat mengirim pesan informatif bersama dengan perintah keberhasilan, kegagalan atau membatalkan:

Example 9 *Authentication failure*; opsional pesan informasi

```
S: <sasl:failure>Plaintext authentication failed (Incorrect
username or password)</sasl:failure>
```

Gambar III.12 Contoh *Authentication Failure* dan Informasi pesan

Setelah kegagalan atau klien membatalkan, klien dapat memulai *handshake* baru. Setelah otentikasi berhasil, upaya lebih lanjut oleh klien untuk memulai *handshake* otentikasi baru secara otomatis akan mengakibatkan *server* mengirimkan kegagalan.

4. Session Start

Setelah otentikasi berhasil, klien harus mengirim paket set IQ standar dalam `jabber ini: iq: auth namespace` untuk memulai sesi. Klien harus menyediakan nama pengguna dan sumber daya untuk sesi bersama dengan paket ini.

Contoh 10 memulai *session* setelah otentikasi berhasil

```
C: <iq id="a1" type="get">
     <query xmlns="jabber:iq:auth">
       <username>rob</username>
     </query>
   </iq>
S: <iq id="a1" type="result">
     <query xmlns="jabber:iq:auth">
```

```
       <username>rob</username>
       <resource/>
     </query>
   </iq>
C: <iq id="a2" type="set">
     <query xmlns="jabber:iq:auth">
       <username>rob</username>
       <resource>laptop</resource>
     </query>
   </iq>
S: <iq id="a2" type="result">
     <query xmlns="jabber:iq:auth"/>
   </iq>
```

Gambar III.13 Contoh Memulai *Session* Setelah Otentikasi Berhasil

Jika klien mencoba untuk memulai sesi sebelum otentikasi, atau *username* diberikan dalam `jabber ini: iq: auth` paket tidak sesuai dengan nama yang diberikan dalam autentifikasi (ketika mekanisme SASL mendukungnya), *server* akan mengembalikan 401 (*Unauthorized*) *packet error*.

III.1.2.2.1 Analisis Otentikasi GTalk

Ketika klien mengirimkan metode otentikasi ke *server*, ia harus menambah dua atribut baru untuk elemen <auth>. Set atribut `client-uses-full-bind-result` menjadi "*true*."

Contoh : Klien memilih metode otentikasi

```
<auth xmlns='urn:ietf:params:xml:ns:xmpp-sasl'
      mechanism='PLAIN'
      xmlns:ga='http://www.google.com/talk/protocol/auth'
      ga:client-uses-full-bind-result='true'>
  ... name and password information
</auth>
```

Gambar III.14 Klien Memilih Metode Otentikasi

Jika pengguna tidak memasukkan nama *domain*, *service* harus menggunakan gmail.com domain dalam permintaan otentikasi awal.

Nilai `client-uses-full-bind-result` menunjukkan bahwa klien akan menggunakan JID penuh yang dikembalikan oleh *server* dalam *bind response* bukan *domain* yang diajukan dalam permintaan otentikasi.

Contoh : Klien menerima JID penuh yang harus digunakan

```
<iq id="0" type="result">
  <bind xmlns="urn:ietf:params:xml:ns:xmpp-bind">
    <jid>example@googlemail.com/pcp16A90FBD</jid>
  </bind>
</iq>
```

Gambar III.15 Contoh Klien Menerima JID Penuh

JID yang dikembalikan dalam hasil *bind* adalah salah satu yang harus digunakan klien untuk semua komunikasi.

Berikut adalah pertukaran disingkat antara klien dan *server*, di mana kontak klien *server* dengan domain gmail.com, dan diberikan sebuah JID dengan domain googlemail.com. Klien kemudian harus menggunakan googlemail.com *node domain.*

Contoh : Klien dan *server* menegosiasikan sambungan dan domain

```
>>>>>>>>>>>>>>>>>>>>>>> Client sends sign-in information...
<auth xmlns="urn:ietf:params:xml:ns:xmpp-sasl"
      mechanism="PLAIN"
      xmlns:ga='http://www.google.com/talk/protocol/auth'
      ga:client-uses-full-bind-result='true'>
... encoded user name and password ... user=example@gmail.com
password=supersecret
</auth>

<<<<<<<<<<<<<<<<<<<<<<< Server sends success...
<success xmlns="urn:ietf:params:xml:ns:xmpp-sasl"/>

>>>>>>>>>>>>>>>>>>>>>>> Client initiates a new stream...
<stream:stream to="gmail.com" version="1.0"
xmlns:stream="http://etherx.jabber.org/streams"
xmlns="jabber:client">

<<<<<<<<<<<<<<<<<<<<<<< Server sends stream header and binding
feature...
<?xml version="1.0" encoding="UTF-8"?>
<stream:stream from="gmail.com" id="X86C60D2CEA180A3E"
version="1.0" xmlns:stream=" http://etherx.jabber.org/streams"
xmlns="jabber:client">

<stream:features>
  <bind xmlns="urn:ietf:params:xml:ns:xmpp-bind"/>
  <session xmlns="urn:ietf:params:xml:ns:xmpp-session"/>
</stream:features>

>>>>>>>>>>>>>>>>>>>>>>> Client sends binding information...
<iq type="set" id="0">
  <bind xmlns="urn:ietf:params:xml:ns:xmpp-bind">
    <resource>
```

```
      pcp
    </resource>
  </bind>
</iq>

<<<<<<<<<<<<<<<<<<<<<<<< Server sends new JID in bind result...
<iq id="0" type="result">
  <bind xmlns="urn:ietf:params:xml:ns:xmpp-bind">
    <jid>
      example@googlemail.com/pcp16A90FBD
    </jid>
  </bind>
</iq>
```

Gambar III.16 Klien dan *Server* Menegosiasikan Sambungan dan *Domain*

III.1.2.2.2 Analisis Otentikasi *Facebook Chat*

Jabber ID pada pengguna *Facebook* adalah *username Facebook* dengan ditambahkan @chat.facebook.com. Seorang pengguna harus memiliki nama pengguna *Facebook* untuk menggunakan *DIGEST-MD5*. Setelah pengguna mendapatkan nama nya, dia harus *logout* dari dan ke *Facebook* sekali bagi kita untuk menyimpan hash khusus *password*.

Contoh : Otentikasi *Facebook Chat* menggunakan SASL

```
ConnectionConfiguration connConfig = new ConnectionConfiguration(
                  “Chat.facebook.com”, 5222, "chat.facebook.com”);
                  connConfig.setDebuggerEnabled(true);
                  connConfig.setSASLAuthenticationEnabled(true);
                  XMPPConnection      connection      =      new
XMPPConnection(connConfig);
            try {
            connection.connect();
            SASLAuthentication.supportSASLMechanism("PLAIN", 0);
            connection.login("firdaus", "superrahasia");
            Presence           presence           =           new
Presence(Presence.Type.available);
            connection.sendPacket(presence);
             }catch(XMPPException ex)
            {
            ex.printStacKTrace();
            }
```

Gambar III.17 Otentikasi *Facebook Chat* Menggunakan SASL

III.1.2.3 Analisis Koneksi ke *Database Server*

Dalam proses pengambilan data pada *Database* MySQL yang berada di *server*, akan melalui beberapa proses diantaranya sistem melakukan *request* kepada *server* melalui script PHP dimana koneksi terjadi di script PHP tersebut. Kemudian PHP akan membaca *request* dari sistem dan memprosesnya untuk melakukan pengambilan data dari *Database*. Contoh : *Client* melakukan *request* kepada *server*

```
List param = new ArrayList();
            param.add(new BasicNameValuePair("sender", isender));
            param.add(new         BasicNameValuePair("recipient",
irecipient));
            JSONObject     json     =     jParser.makeHttpRequest
("http://atmessenger.p.ht/getdetailhistory.php", "GET", param);
```

Gambar III.18 *Request* dari *Client* ke *Database Server*

Contoh : *Script* PHP untuk melakukan pengambilan data dari *Database*

```
<?php
include 'db_config.php';
mysql_connect($DB_SERVER,$DB_USER, $DB_PASSWORD);
mysql_select_db($DB_NAME);
// menerima request dari Android
$n = $_REQUEST['sender'];
$rc = $_REQUEST['recipient'];
$q   =   mysql_query("SELECT   name,message,tr_message,timea   FROM
history WHERE (sender = '{$n}' and recipient = '{$rc}') or (sender
= '{$rc}' and recipient = '{$n}')");
$r = array();
$c = mysql_num_rows($q);
if($c>0){
    $r["history"] = array();
    while($e = mysql_fetch_array($q)){
        $a = array();
        $a["name"]       = $e["name"];
        $a["message"]      = $e["message"];
```

```
      $a["tr_message"]   = $e["tr_message"];
      $a["timea"]    = $e["timea"];
        array_push($r["history"], $a);
    }
    $r["success"] = 1;
    // mengirimkan data JSON ke Android
    echo json_encode($r);
}else{
    $r["success"] = 0;
    echo json_encode($r);
}
mysql_close();
?>
```

Gambar III.19 *Script* PHP untuk Melakukan Pengambilan Data

setelah itu jika data ditemukan akan dikirimkan kembali ke sistem berupa data JSON. Kemudian sistem akan melakukan parsing data berupa JSON tersebut.

Contoh : *Parsing* object JSON

```
try {
     int success = json.getInt("success");
     if (success == 1) {
     arraychat = json.getJSONArray("history");
     for (int i = 0; i < arraychat.length(); i++) {
         JSONObject c = arraychat.getJSONObject(i);
         if(c.getString("name").equals(MainActivity.username))
            {
              chat  =  new  ListChat("Me",  c.getString("message"),
c.getString("tr_message"), c.getString("timea"));
            }
         else
              chat     =     new     ListChat(c.getString("name"),
c.getString("message"),c.getString("tr_message"),
c.getString("timea"));
              listChat.add(chat);
       }
      }
```

```
      } catch (JSONException e)
        {
         e.printStackTrace();
        }
```

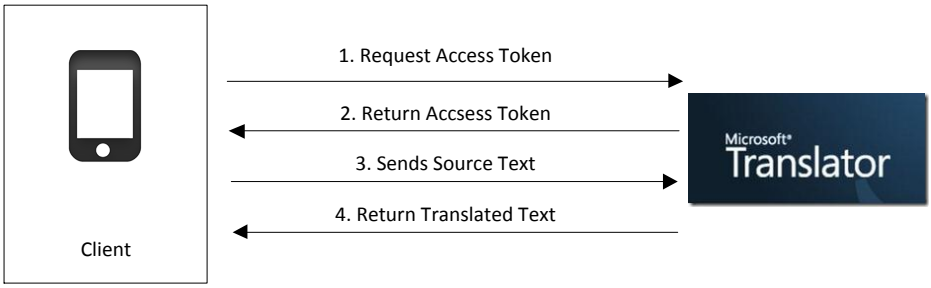
Gambar III.20 *Parsing* Object JSON

III.1.2.4 Analisis *Microsoft Translator*

Micrososft Translator API merupakan *API* yang digunakan fitur penerjemah yang terdapat pada sistem. Secara *default API* ini akan menerjemahkan semua pesan yang diterima ke dalam bahasa yang digunakan oleh *user*.

Berikut Langkah-langkah menerjemahkan pesan *menggunakan Microsoft Translator API* :

1. *Client* Meminta *Access Token* ke *server*
2. *Server* merespon dengan memberikan *Access Token*
3. *Client* mengirim teks sumber untuk diterjemahkan
4. *Server* membalas dengan teks sumber yang telah diterjemahkan



Gambar III.21 Alur Penggunaan *Micrososft Translator API*

1. **Akses Token**

Client memulai dengan meminta *access token* ke *server*, kemudian *server* akan merespon dengan memberikan *access token* ke *Client*. Parameter untuk permintaan token menggunakan *URL-encoded* dan dikirimkan melalui *HTTP request body*. Berikut parameter-parameter yang digunakan dalam meminta *access token*

Tabel III.1 *Token Request Input Parameters*

Parameter	Deskripsi
client_id	*Client id* yang didapatkan ketika mendaftarkan applikasi ke *Azure DataMarket*
client_secret	*Client secret* yang didapatkan ketika mendaftarkan applikasi ke *Azure DataMarket*
scope	Gunakan http://*API*.microsofttranslator.com URL sebagai nilai lingkup untuk Penterjemah *API Microsoft*.
grant_type	Gunakan "*client_credentials*" sebagai nilai *grant_type* untuk Penterjemah *API Microsoft*.

Respon untuk permintaan akses token berisi akses token yang dapat Anda gunakan untuk mengakses *Translator API Microsoft*. Respon berupa *JSON-encoded* dengan beberapa attribut *output* sebagai berikut.

Tabel III.2 *Token Request Output Properties*

Parameter	Deskripsi
access_token	Akses token yang dapat Anda gunakan untuk otentikasi Anda akses ke *Microsoft Translator API.*
token_type	Format akses token.
expires_in	Jumlah detik dimana akses token valid.
scope	*Domain* dimana token ini berlaku. Untuk *Microsoft Translator API*, domain adalah http://*API*.microsofttranslator.com.

Contoh 1 Akses Token

```
static void Main(string[] args)
        {
            AdmAccessToken admToken;
            string headerValue;
            AdmAuthentication admAuth = new
AdmAuthentication("clientID", "client secret");
            try
            {
                admToken = admAuth.GetAccessToken();
                headerValue = "Bearer " + admToken.access_token;
                DetectMethod(headerValue);
            }
            catch (WebException e)
            {
                ProcessWebException(e);
```

```
            Console.WriteLine("Press any key to continue...");
            Console.ReadKey(true);
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            Console.WriteLine("Press any key to continue...");
            Console.ReadKey(true);
        }
```

Gambar III.22 Akses Token

2. Menerjemahkan Pesan

Setelah *client* mendapatkan akses token, *client* dapat memulai menerjemahkan pesan teks dari suatu bahasa ke bahasa lain. Uri yang digunakan untuk mengirimkan pesan memiliki format sebagai berikut :

`http://API.microsofttranslator.com/V2/Ajax.svc/Translate`

Parameter-parameter *query* yang dapat digunakan pada *API Microsoft Translator* di cantumkan pada table berikut

Tabel III.3 Parameter *Query Microsoft Translator API*

Parameter	Deskripsi
appid	Diperlukan. Sebuah *string* yang berisi "*Bearer*" + "" + akses token.
text	Diperlukan. Sebuah *string* yang mewakili teks untuk diterjemahkan. Ukuran teks harus tidak melebihi 10000 karakter.
from	Opsional. Sebuah *string* yang mewakili kode bahasa teks terjemahan. Jika dibiarkan kosong respon akan memberikan hasil dari *language auto-detection*
to	Diperlukan. Sebuah *string* yang mewakili kode bahasa untuk menerjemahkan teks ke bahasa target.
contentType	Diperlukan. Format teks yang diterjemahkan. Format yang didukung adalah "*text* / *plain*" dan "*text* / *html*".
category	Opsional. Sebuah string yang berisi kategori (*domain*) terjemahan. Defaultnya adalah "umum".

Jika *request* sukses, maka *server* akan memberikan sebuah nilai *string* yang mewakili teks yang diterjemahkan.

Contoh 2 menerjemahkan pesan

```
var from = "en", to = "es", text = "hello world";
            var s = document.createElement("script");
            s.src =
"http://API.microsofttranslator.com/V2/Ajax.svc/Translate" +
                "?appId=Bearer " +
encodeURIComponent(window.accessToken) +
                "&from=" + encodeURIComponent(from) +
                "&to=" + encodeURIComponent(to) +
                "&text=" + encodeURIComponent(text) +
                "&oncomplete=mycallback";
```

Gambar III.23 Menerjemahkan Pesan

III.1.3 Analisis Kebutuhan Non Fungsional

Analisis dan kebutuhan non fungsional meliputi analisis dan kebutuhan perangkat keras serta analisis dan kebutuhan perangkat lunak. Adapun kebutuhan non-fungsional untuk menjalankan aplikasi ini meliputi kebutuhan perangkat keras,kebutuhan perangkat lunak dan pengguna sistem yang akan memakai aplikasi. Analisis kebutuhan non-fungsional bertujuan agar aplikasi yang dibangun dapat digunakan sesuai dengan kebutuhan pengguna aplikasi dalam mencari informasi yang dibutuhkan.

III.1.3.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan berdasarkan kebutuhan minimal yang harus terpenuhi untuk menggunakan sistem ini adalah sebagai berikut:

Tabel III.4 Kebutuhan Perangkat keras

Aplikasi Front End	Aplikasi Back End
1. Telepon seluler (OS *Android* Minimal 2.2 *Froyo*)	1. PC/Laptop dengan spesifikasi: a. *Processor Core 2 duo* 2.93GHz b. RAM 2GB *c.* *Harddisk* 250GB
2. Memiliki koneksi internet EDGE,GPRS,2G dan 3G	2. Koneksi Internet

III.1.3.2 Analisis Kebutuhan Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam pembangunan aplikasi ini adalah :

Tabel III.5 Kebutuhan Perangkat Lunak

Aplikasi Front End	Aplikasi Back End
1. *Eclipse Helios*	1. *Dreamweaver 8*
2. *SQLite*	2. *MySQL*
3. *ADT 21.1.0*	3. *Mozilla Firefox 20.0.1*
4. *Android SDK*	

Sedangkan perangkat lunak yang dibutuhkan untuk menjalankan aplikasi ini adalah sebagai berikut :

Tabel III.6 Kebutuhan Perangkat untuk Lunak Menjalankan Aplikasi

Aplikasi Front End	Aplikasi Back End
1. Sistem Operasi *Android* Minimal Versi 2.2 *Froyo*	1. Browser *Mozilla Firefox*, *Internet Explorer, Chrome* atau browser yang lain.

III.1.3.3 Analisis Pengguna Sistem (*User*)

Spesifikasi pengguna sistem dimaksudkan untuk mengetahui siapa saja aktor yang terlibat dalam menjalankan sistem. Berdasarkan hasil pengamatan, pengguna yang terlibat dalam sistem yang sedang berjalan ialah:

1. *End User*, yaitu para pengguna *smartphone* dengan *platform Android* versi 2.2 atau lebih tinggi. Dalam menggunakan aplikasi ini, pengguna terhubung dengan koneksi GPRS/EDGE/2G atau 3G
2. *Admin*, sebagai pengelola aplikasi mempunyai kapabilitas untuk memonitoring sistem, melihat pengguna yang aktif, dan mengelola data pengguna seperti menambah dan menghapus data.

Berikut karakteristik pengguna yang terlibat dalam sistem yang berjalan ditunjukkan oleh tabel berikut.

Tabel III.7 Karakteristik Pengguna

Pengguna	Tanggung Jawab	Hak Akses	Tingkat Keterampilan
Admin	Melakukan pemeliharaan terhadap sistem	Mengakses dan mengelola keseluruhan data	Mengetahui cara melakukan pengolahan data
End user	Menjaga kerahasian *ID* dan *Password*	Mengirim dan menerima pesan, membuat akun baru	Minimal dapat menjalankan aplikasi pada perangkat *Android*

III.1.4 Analisis Kebutuhan Fungsional

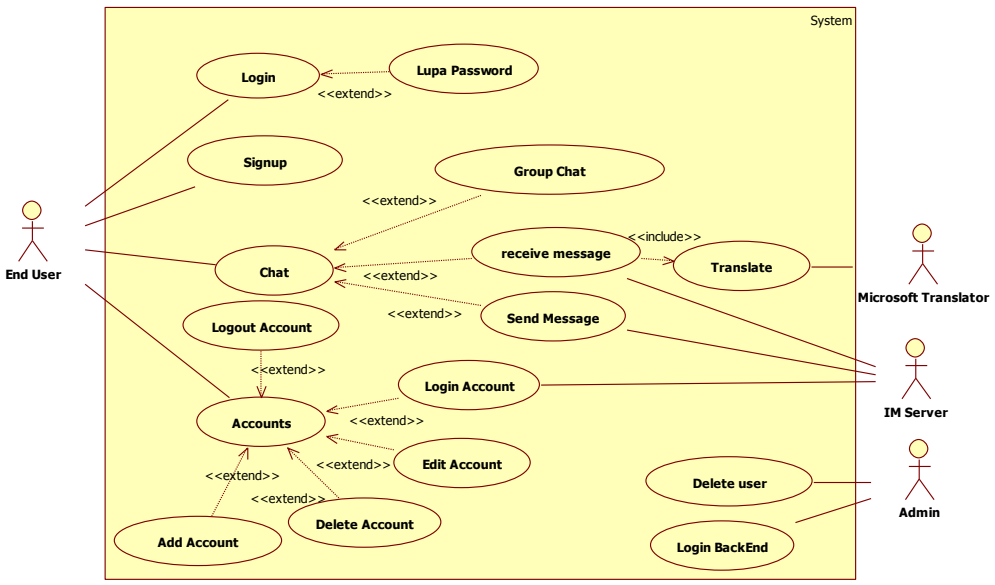
Kebutuhan fungsional berhubungan dengan fitur *software* yang akan dibuat atau dikembangkan. Berikut adalah tahapan analisis kebutuhan fungsional aplikasi *chatting* dengan penerjemah otomatis berbasis *mobile.*

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik.

Analisis yang dilakukan dimodelkan dengan menggunakan UML (*Unified Modeling Language*). Tahap-tahap pemodelan dalam analisis tersebut antara lain identifikasi aktor, *Use Case diagram, skenario, sequence diagram*, *activity diagram*, dan *class diagram*.

III.1.4.1 *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. Dari hasil analisis aplikasi yang ada maka *Use Case diagram* untuk aplikasi ini dapat di lihat pada gambar III.24:



Gambar III.24 *Use Case* Diagram Aplikasi

a) Identifikasi *Actor*

Actor adalah abstraksi dari orang dan sistem yang lain yang mengaktifkan fungsi dari target sistem. Berikut adalah aktor-aktor yang berperan dalam menjalan sistem yang dibangun.

Table III.8 Tabel *Actor Use Case*

No	Aktor	Deskripsi
A-01	*End User*	Merupakan *Actor* dari aplikasi yang dibangun, Aplikasi *frontend*
A-02	*Admin*	Merupakan *Actor* yang berperan mengelola data pengguna
A-03	*IM Server*	Merupakan *Actor* yang berperan mengotentikasi *account*, dan sebagai *server* untuk mengirim dan menerima pesan.
A-04	*Microsoft Translator*	Merupakan *Actor* yang berperan menterjemahkan pesan yang akan dikirim ke bahasa yang di gunakan lawan bicara.

b) Identifikasi *Use Case Diagram*

Berikut tabel identifikasi *Use Case* yang terdapat dalam sistem :

Tabel III.9 Identifikasi *Use Case*

Use Case* Aplikasi *FrontEnd		
No	***Use Case***	**Deskripsi**
UCF-01	*Login*	Fungsionalitas Untuk masuk ke dalam aplikasi *FrontEnd*
UCF-02	*SignUp*	Fungsionalitas untuk membuat sebuah akun baru pada aplikasi *FrontEnd*
UCF-03	Lupa *Password*	Fungsionalitas untuk mereset *password*
UCF-04	*Chat*	Fungsionalitas untuk menampilkan *Chat History*
UCF-05	*Group Chat*	Fungsionalitas Untuk Membuat sebuah *Group Chat*
UCF-06	*Receive Message*	Fungsionalitas untuk menerima pesan
UCF-07	*Send Message*	Fungsionalitas untuk mengirimkan pesan
UCF-08	*Accounts*	Fungsionalitas untuk menampilkan *Accounts IM*
UCF-09	*Login Account*	Fungsionalitas untuk *login Account IM*
UCF-10	*Logout Account*	Fungsionalitas untuk *logout Account IM*
UCF-11	*Add Account*	Fungsionalitas untuk menambahkan *Account IM*
UCF-12	*Edit Account*	Fungsionalitas untuk *mengedit Account IM*

UCF-13	*Delete Account*	Fungsionalitas untuk menghapus *Account IM*
Use Case* Aplikasi *BackEnd		
No	***Use Case***	**Deskripsi**
UCB-01	*Login BackEnd*	Fungsionalitas untuk masuk ke dalam aplikasi *BackEnd*
UCB-02	*Delete User*	Fungsionalitas untuk menghapus data *user*

III.1.4.2 Skenario *Use Case*

Skenario *Use Case* mendeskripsikan urutan langkah-langkah dalam proses bisnis, baik yang dilakukan aktor terhadap sistem maupun yang dilakukan oleh sistem terhadap aktor.

III.1.4.2.1 Sekenario *Use Case FrontEnd*

Table III.10 Skenario *Use Case Login*

Identifikasi	
Nama *Use Case*	*Login*
Aktor	*End User*
Tujuan	Masuk ke dalam sistem
Kondisi awal	Sistem menampilkan halaman *Login*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menginputkan *username* dan *password*	
	2. Sistem melakukan autentikasi *username* dan *password* yang diinputkan
	3. Bila sesuai maka sistem akan menampilkan halaman utama
Skenario Alternatif	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa *username* atau *password* salah
2. Mengisi kembali *username* dan *password*	
	3. Sistem akan melakukan autentikasi *username* dan *password* kembali

	4. Jika sesuai maka sistem akan menampilkan halaman utama
Kondisi Akhir	Sistem menampilkan daftar teman

Tabel III.11 Skenario *Use Case Signup*

Identifikasi	
Nama *Use Case*	*Signup*
Aktor	*End User*
Tujuan	Membuat sebuah akun untuk masuk ke dalam sistem.
Kondisi awal	Menampilkan halaman *Signup*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Mengisi field yang tersedia di halaman *signup*	
	2. Mengecek ketersediaan *username*
	3. Sistem menyimpan data-data *End User* yang di inputkan
	4. Menampilkan pesan *Account* berhasil di buat dan masuk ke halaman *login*
Skenario Alternatif	
Aksi Aktor	**Reaksi Sistem**
	3. Menampilkan Pesan bahwa *username* sudah di gunakan
4. Mengisi kembali *filed username*	
	5. Mengecek ketersediaan *username* kembali
	6. Menyimpan data – data yang di inputkan
	7. Menampilkan pesan *account* berhasi dibuat dan masuk ke halaman *login*
Kondisi Akhir	Menampilkan halaman *login*

Tabel III.12 Skenario *Use Case* Lupa *Password*

Identifikasi	
Nama *Use Case*	*Lupa Password*
Aktor	*End User*
Tujuan	Mereset *password*
Kondisi awal	Menampilkan halaman *Accounts*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**

1. Menekan tombol "*Forgot Password*"	
	2. Menampilkan halaman *Forgot Password*
3. Menginputkan *username, security question* dan *security answer*	
	4. Validasi *username,security question* dan *security answer*
	5. Menampilkan halaman *reset password*
6. Menginputkan *password* baru	
	7. Menampilkan pesan *password* berhasil direset dan menampilkan halaman *login*
Skenario Alternatif	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa data yang dimasukan tidak sesuai
	2. Menampilkan kembali halaman *forgot password*
Kondisi Akhir	Menampilkan halaman *Login*

Tabel III.13 Skenario *Use Case Accounts*

Identifikasi	
Nama *Use Case*	*Accounts*
Aktor	*End User*
Tujuan	Menampilkan *Account* yang dimiliki *User*
Kondisi awal	Menampilkan halaman utama
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol "*Accounts*"	
	2. Menampilkan halaman *Accounts* yang berisi akun-akun yang dimiliki oleh *user*.
Kondisi Akhir	Menampilkan halaman *Accounts*

Tabel III.14 Skenario *Use Case Add Account*

Identifikasi	
Nama *Use Case*	*Add Account*
Aktor	*End User*
Tujuan	Menambahkan Sebuah akun IM
Kondisi awal	Menampilkan halaman *Accounts*
Skenario Utama	

Aksi Aktor	Reaksi Sistem
3. Menekan tombol "*Add Account*"	
	4. Menampilkan halaman *Add Account*
5. Menginputkan alamat *email*, *password* dan jenis *Account*	
	6. Menyimpan data yang di inputkan ke *Database*
	7. Menampilkan *account* IM di halaman *Accounts*
Senario Alternatif	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol "*Add Account*"	2. Menampilkan halaman "*Add Account*"
3. Menginputkan alamat *email*,*password* dan jenis *Account*	4. Menampilkan pesan "*Account already exist*"
	5. Menampilkan Halaman *Add Account*
Kondisi Akhir	Menampilkan halaman *Accounts*

Tabel III.15 Skenario *Use Case Edit Account*

Identifikasi	
Nama *Use Case*	*Edit Account*
Aktor	*End User*
Tujuan	Mengedit *Account* yang sudah ada
Kondisi awal	Menampilkan halaman *Accounts*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tampilan *account* yang akan diedit	
	2. Menampilkan menu *edit*, *delete*, *login*, dan *logout*
3. Menekan Menu *Edit*	
	4. Menampilkan menu *edit account*
5. Mengedit data pada *field* – field yang disediakan	
6. Menekan tombol "*Save*"	
	7. Menyimpan perubahan ke *Database*
	8. Menampilkan Halaman *Accounts*
Kondisi Akhir	Mengedit data *account* sesuai data yang dipilih

Tabel III.16 Skenario *Use Case Delete Account*

Identifikasi	
Nama *Use Case*	*Delete Account*
Aktor	*End User*
Tujuan	Menghapus salah satu akun IM
Kondisi awal	Menampilkan halaman *Accounts*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tampilan *Account* yang akan di hapus	
	2. Menampilkan menu *edit*, *delete*, *login* dan *logout*
3. Menekan Menu "*Delete*"	
	4. Menampilkan pesan "*Are you sure want to delete this account*?"
5. Menekan Tombol "*Yes*"	
	6. Menghapus *Account* dari *Database*
Skenario Alternatif	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan "*Are You sure want to delete this account*?"
2. Menekan Tombol "*No*"	
	3. Menampilkan halaman *Accounts*
Kondisi Akhir	Menghapus data *account* yang dipilih

Tabel III.17 Skenario *Use Case Login Account*

Identifikasi	
Nama *Use Case*	*Login Account*
Aktor	*End user*
Tujuan	*Login* menggunakan akun IM yang di pilih
Kondisi awal	Menampilkan halaman *Accounts*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tampilan *Account*	
	2. Menampikan menu *edit*, *delete*, *login*, dan *logout*
3. Menekan Tombol "*Login*"	
	4. Autentikasi *Account* ke *server* IM
	5. Login Menggunakan *Account* yang dipilih
Skenario Alternatif	
Aksi Aktor	**Reaksi Sistem**

	1. Autentikasi *Account* Ke *server* IM
	2. Menampilkan Pesan "*Login Account failed*"
Kondisi Akhir	Login ke *server* IM menggunakan *account* yang di pilih

Tabel III.18 Skenario *Use Case Logut Account*

Identifikasi	
Nama *Use Case*	*Logout Account*
Aktor	*End user*
Tujuan	*Logout* menggunakan akun IM yang di pilih
Kondisi awal	Menampilkan halaman *Accounts*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tampilan *Account*	
	2. Menampikan menu *edit, delete, login*, dan *logout*
3. Menekan tombol "*Logout*"	
	4. *Logout Account* yang dipilih dari *server* IM
Kondisi Akhir	*Logout acount* yang dipilih dari *server* IM

Tabel III.19 Skenario *Use Case Chat*

Identifikasi	
Nama *Use Case*	*Chat*
Aktor	*End user*
Tujuan	Memulai percakapan dengan *user* lain
Kondisi awal	Menampilkan halaman *Friendlist*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tampilan *friendlist* yang dituju	
	2. Menampikan *chat window* sesuai dengan *user* yang di tuju
Kondisi Akhir	Menampilkan *chat window* sesuai dengan *user* yang di tuju

Tabel III.20 Skenario *Use Case Create Group Chat*

Identifikasi	
Nama *Use Case*	*Create Group Chat*
Aktor	*End user*
Tujuan	Membuat sebuah *group chat*
Kondisi awal	Menampilkan menu *Group chat*

Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol *create group chat*	
	2. Menampikan field "*Group Chat Name*"
3. Menginputkan nama *group chat*	
	4. Membuat sebuah *Group chat* dengan nama yang di tentukan
	5. Menampilkan daftar teman yang tersedia
6. Memilih daftar teman yang tersdia untuk ditambahkan ke dalam *group*	
	7. Menambahkan daftar teman ke dalam *group*
Kondisi Akhir	Membuat Sebuah *Group Chat*

Tabel III.21 Skenario *Use Case Send Message*

Identifikasi	
Nama *Use Case*	*Send Message*
Aktor	*End user*
Tujuan	Mengirim pesan kepada *user* lain yang dituju
Kondisi awal	Menampilkan halaman *chat*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Memilih daftar teman yang tersedia	
	2. Menampilkan *Chat window*
3. Menulis pesan teks pada *filed* yang disediakan	
4. Menekan tombol "*Send*"	
	5. Mengirim Pesan Ke *server* IM untuk disampaikan ke target *user*
Kondisi Akhir	Pesan terkirim ke *Server* IM

Tabel III.22 Skenario *Use Case Receive Message*

Identifikasi	
Nama *Use Case*	*Receive Message*
Aktor	*IM Server*
Tujuan	Menampilkan pesan yang diterima dan sudah diterjemahkan
Kondisi awal	Menampilkan halaman *chat*

Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Mengirim pesan dari *IM server*	
	2. Menerima pesan dari IM *server*. Kemudian sistem mengecek bahasa apa yang digunakan pesan yang diterima. Kemudian sistem akan mengirimkan pesan tersebut ke *Microsoft translator* untuk diterjemahkan ke dalam bahasa yang digunakan oleh *user*. Pesan yang telah diterjemahkan akan ditampilkan dihalaman *chat*.
Skenario Alternatif	
Aksi Aktor	**Reaksi Sistem**
	1. Jika Bahasa dari pesan yang diterima sama dengan bahasa yang digunakan oleh *user* maka pesan akan langsung ditampilkan di halaman *chat*.
Kondisi Akhir	Menampilkan pesan yang sudah diterjemahkan.

Tabel III.23 Skenario *Use Case Translate*

Identifikasi	
Nama *Use Case*	*Translate*
Aktor	*Microsoft Translator API*
Tujuan	Menerjemahkan Pesan
Kondisi awal	Menerima pesan teks
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
	1. Menerima pesan teks dari IM *server*
	2. Sistem mengecek bahasa dari pesan teks yang diterima
	3. Mengirim pesan teks ke *Microsoft Translator API* untuk diterjemahkan
4. Menerima pesan teks, bahasa awal, dan bahasa yang dituju	
5. Menerjemahkan pesan dan	

mengirimkan hasil terjemahan ke sistem	
	6. Menampilkan pesan yang telah diterjemahkan dihalaman *chat*
Skenario Alternatif	
Aksi Aktor	**Reaksi Sistem**
	1. Mengecek bahasa dari pesan yang diterima, apabila bahasa dari pesan teks yang diterima sama dengan bahasa yang digunakan *user* maka pesan teks akan langsung ditampilkan di halaman *chat*
Kondisi Akhir	Pesan teks yang diterima telah diterjemahkan.

III.1.4.2.2 Sekenario *Use Case BackEnd*

Tabel III.24 Skenario *Use Case Login BackEnd*

Identifikasi	
Nama *Use Case*	*Login BackEnd*
Aktor	*Admin*
Tujuan	Masuk ke dalam aplikasi *BackEnd*
Kondisi awal	Sistem menampilkan halaman *Login*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menginputkan *username* dan *password*	
	2. Sistem melakukan autentikasi *username* dan *password* yang diinputkan
	3. Bila sesuai maka sistem akan menampilkan halaman utama
Skenario Alternatif	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan pesan bahwa *username* atau *password* salah
2. Mengisi kembali *username* dan *password*	
	3. Sistem akan melakukan autentikasi *username* dan *password* kembali
	4. Jika sesuai maka sistem akan menampilkan halaman utama
Kondisi Akhir	Sistem menampilkan Halaman Utama

Tabel III.25 Skenario *Use Case Delete User*

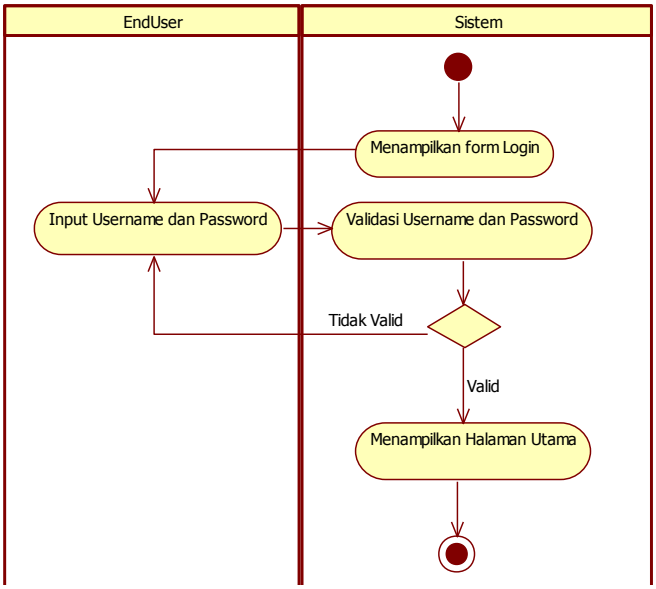
Identifikasi	
Nama *Use Case*	*Delete User*
Aktor	*Admin*
Tujuan	Menghapus *user* dari *Database*
Kondisi awal	Menampilkan *Database user*
Skenario Utama	
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol delete pada *user* yang dipilih	
	2. Menampilkan pesan "*Are you sure want to delete this user*?"
3. Menekan Tombol "*yes*"	
	4. Menghapus *user* dari *Database*
Skenario Alternatif	
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan "*Are you sure want to delete this user*?"
2. Menekan Tombol "*no*"	
	3. Mambatalkan penghapusan *user*
Kondisi Akhir	Menghapus data *user* yang dipilih

III.1.4.3 *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi dan bagaimana mereka berakhir. Penggambaran *activity diagram* memiliki kemiripan dengan *flowchart diagram*. *Activity diagram* memodelkan event-event yang terjadi pada *Use Case* dan digunakan untuk pemodelan aspek dinamis dari sistem.

III.1.4.3.1 Activity Diagram Aplikasi *FrontEnd*

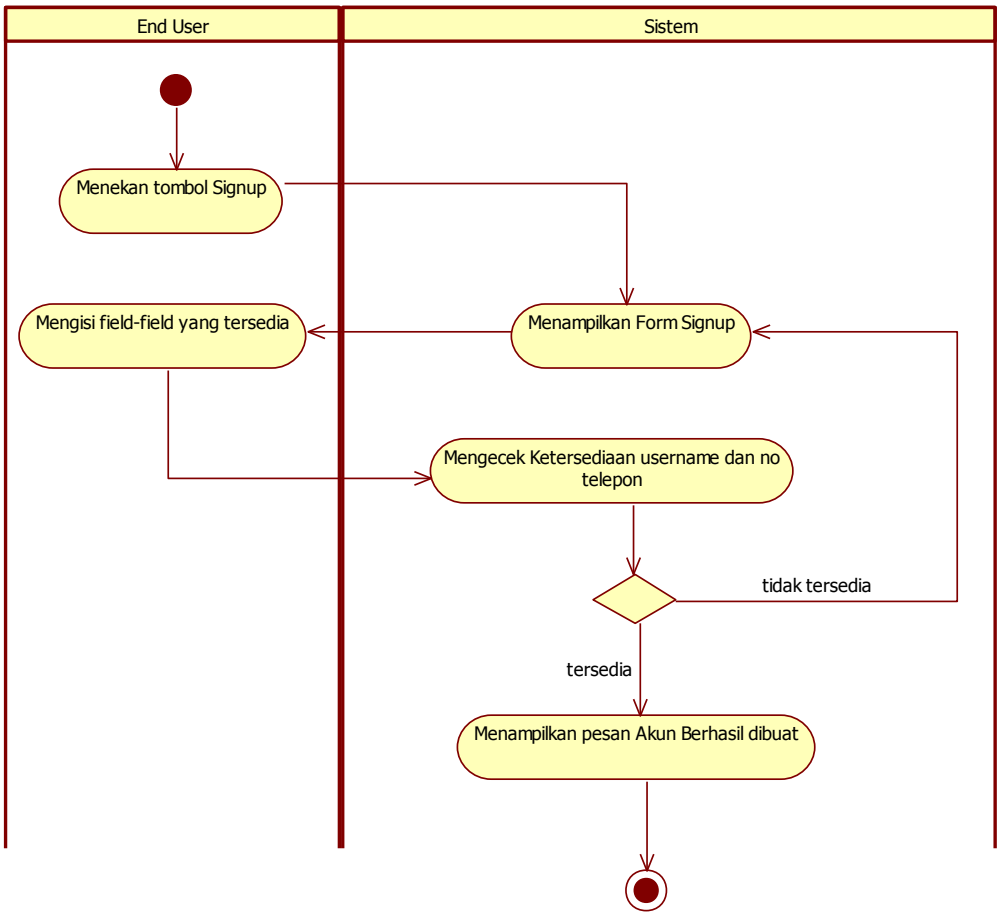
a. *Activity Diagram* Login



Gambar III.25 *Activity Diagram Login*

Gambar III.25 menggambarkan aktifitas yang terjadi pada sistem ketika *user* menekan tombol *login*. Sistem akan menampilkan form *login* yang terdiri dari *username* dan *password* yang harus di inputkan. Jika *username* dan *password* tidak sesuai maka sistem akan menampilkan *form login* untuk diisi kembali. Jika *username* dan *password* sudah sesuai maka sistem akan *login* ke IM *server* menggunakan *account* IM yang tersedia, jika *valid* sistem akan menampilkan tampilan *user* yang *online*. Jika tidak valid sistem akan menampilkan pesan "*Unable to login this* "*account name", please check email and password*." Dan menampilkan tampilan *user* yang sedang *online*.

b. Activity Diagram Signup

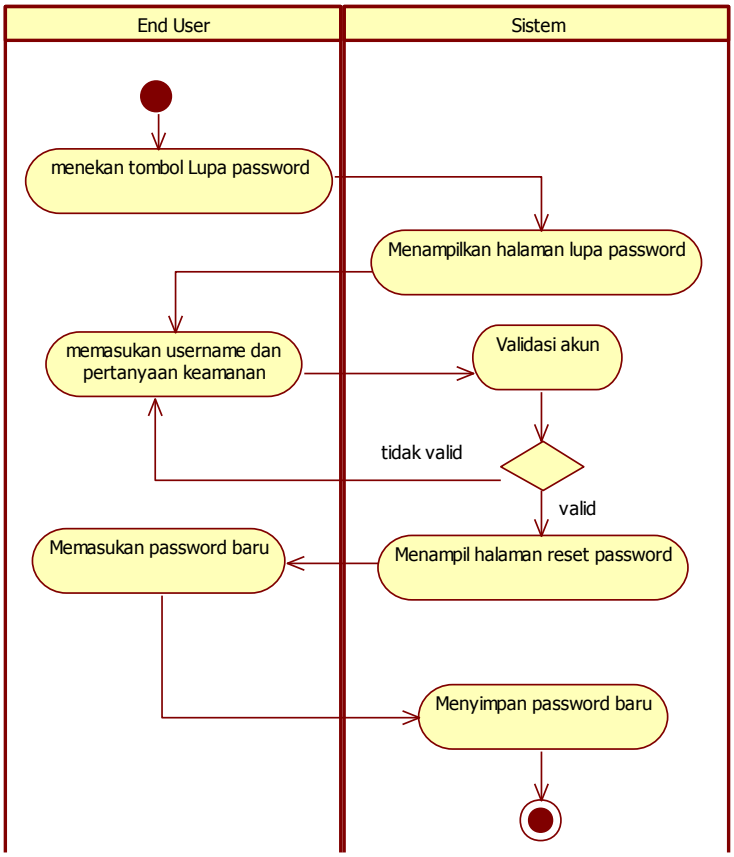


Gambar III.26 *Activity Diagram Signup*

Gambar III.26 Menggambarkan aktifitas yang terjadi pada sistem ketika *user* menekan tombol *signup*. Sistem akan menampilkan *form signup* yang terdiri dari *filed-field username, password, re-password, email, phone number* yang harus diinputkan. Sistem akan mengecek ketersediaan *username* dan *email*, jika tersedia, sistem akan menampilkan pesan "*account has been successfully created*" jika tidak tersedia maka sistem akan menampilkan *from signup* untuk diisi kembali.

c. Activity diagram lupa *password*

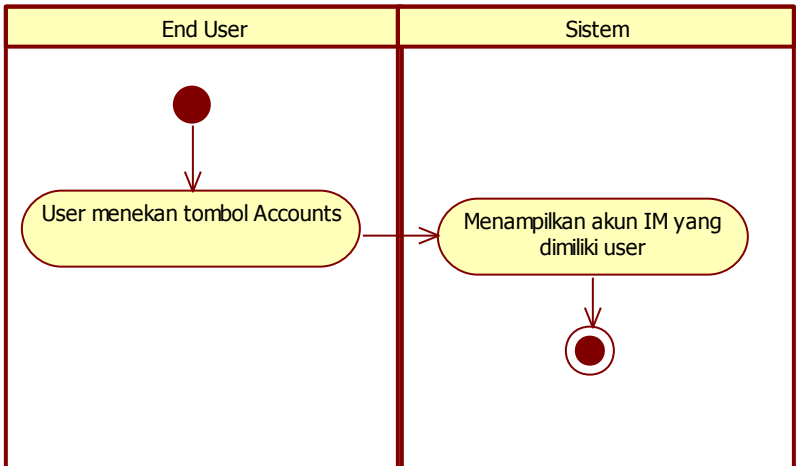
Activity Diagram lupa *password* menggambarkan aktifitas yang terjadi pada sistem ketika *user* menekan tombol lupa *password*.



Gambar III.27 *Activity Diagram* Lupa *Password*

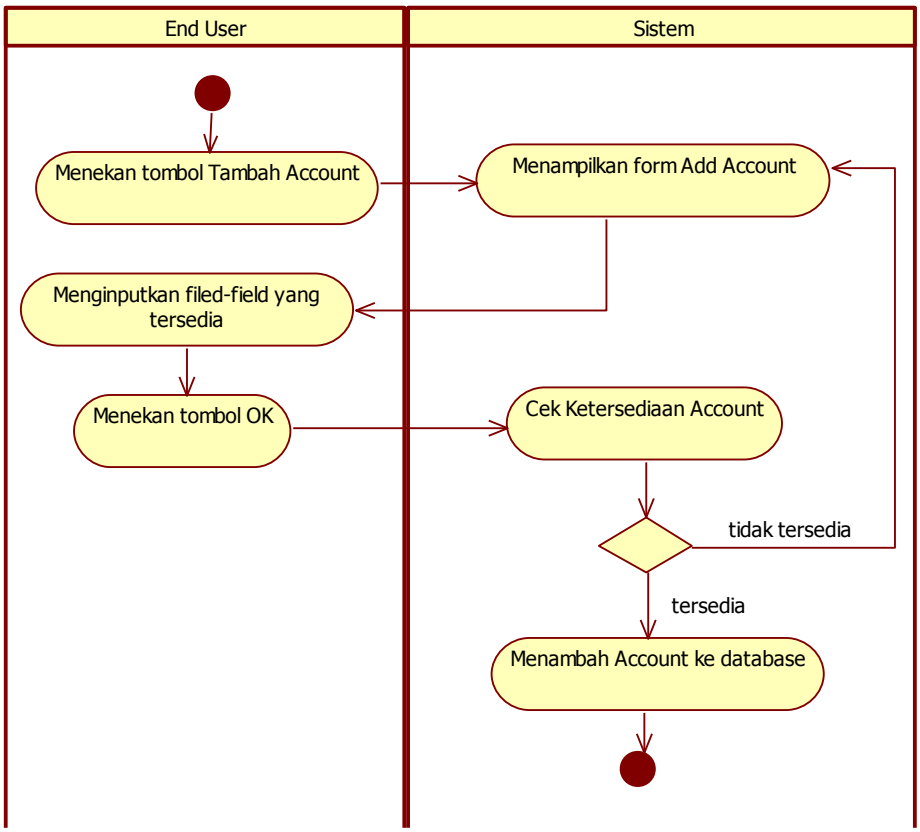
d. *Activity Diagram Accounts*

Activity diagram Accounts menggambarkan aktifitas yang terjadi ketika *user* menekan tombol *accounts*



Gambar III.28 Activity Diagram Accounts

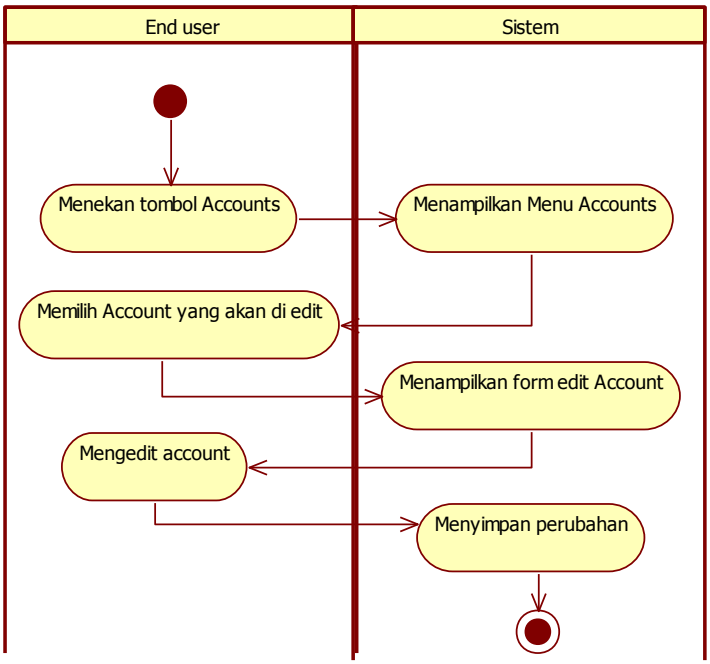
e. Activity Diagram Add Account



Gambar III.29 *Activity Diagram Add Account*

Gambar III.29 Menggambarkan aktifitas yang terjadi pada sistem ketika *user* menekan tombol *Add Account.* Sistem akan menampilkan *form Add Account* yang terdiri dari *field email* dan *password* yang harus diinputkan oleh *user*. Kemudian sistem akan menyimpannya ke dalam *Database*.

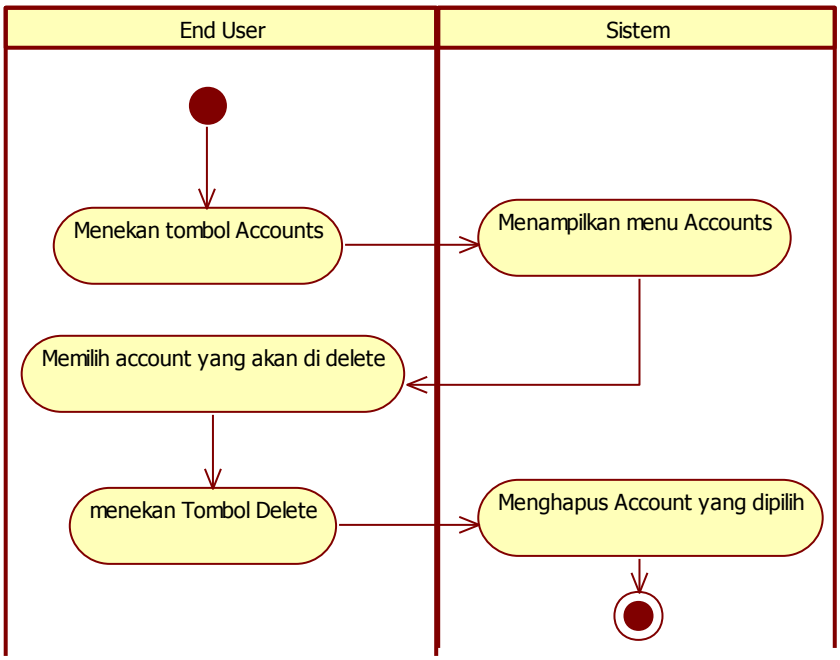
f. Activity Diagram Edit Account



Gambar III.30 *Activity Diagram Edit Account*

Gambar III.30 menggambarkan aktivitas yang terjadi pada sitem ketika *user* menekan tombol *Edit Account. User* menekan tombol Accounts, sistem akan menampilkan *Account* yang dimiliki oleh *user*. Kemudian *user* memilih *Account* yang akan diedit, sistem akan menampilkan form *Edit Account* dengan *field-field* yang sudah terisi data *user* untuk diedit. Setelah selesai diedit sistem akan menyimpan perubahan ke dalam *Database*.

g. *Activity Diagram Delete Account*

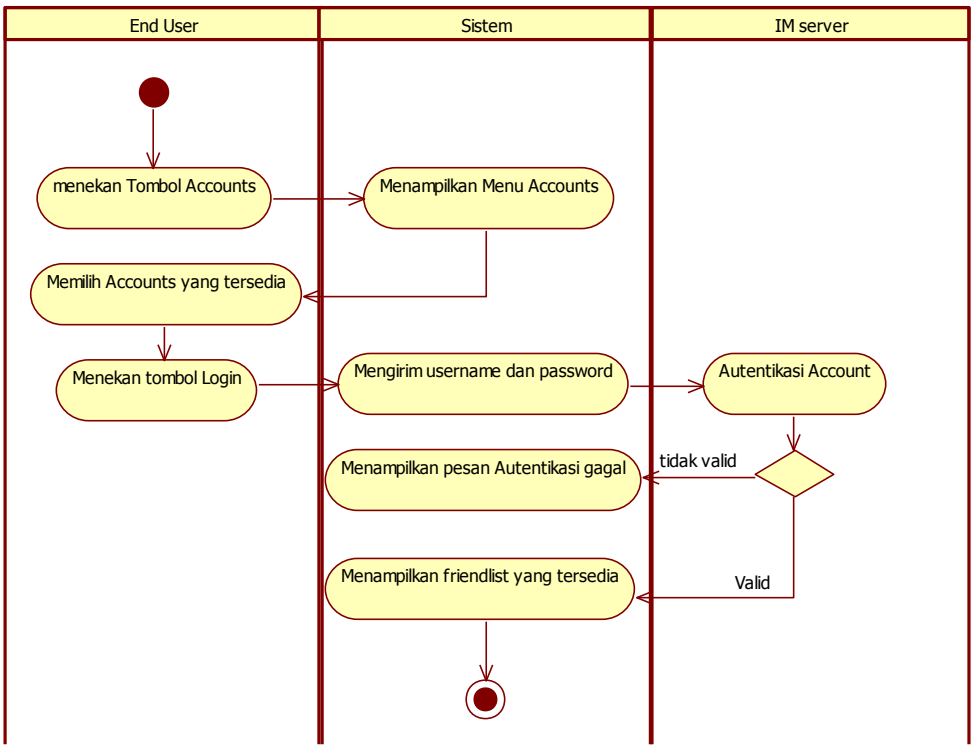


Gambar III.31 *Activity Diagram Delete Account*

Gambar III.31 Menggambarkan aktivitas yang terjadi pada sistem ketika *user* menghapus *Account*. Sistem akan menampilkan *Account-account* yang dimiliki oleh *user*, kemudian *user* akan memilih *account* yang akan dihapus, ketika *user* menekan tombol *Delete*, maka *Account* yang dipilih akan dihapus dari *Database*.

h. Activity Diagram Login Account

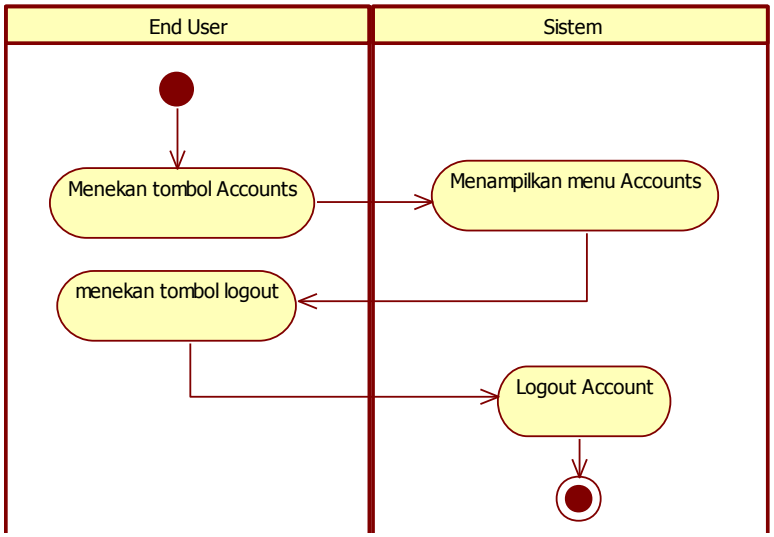
Activity diagram login Account menjelaskan aktifitas yang terjadi pada sistem ketika *user* menekan tombol *login* pada *menu account. Activity diagram Login Account* dapat dilihat pada gambar III.32



Gambar III.32 *Activity Diagram Login Account*

i. Activity Diagram Logout Account

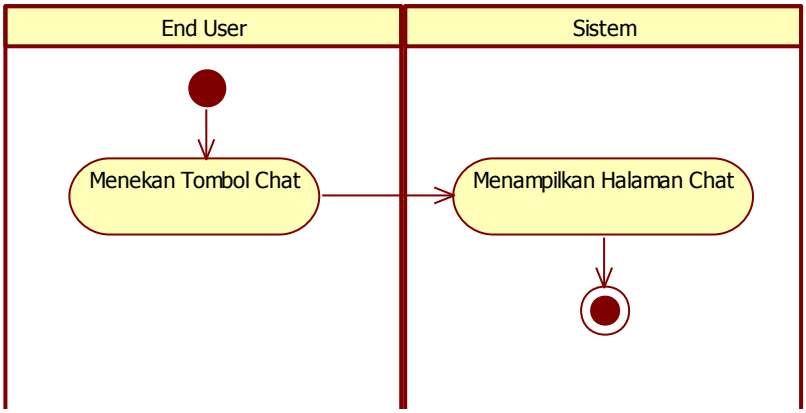
Activity diagram Logout Account menjelaskan aktifitas yang terjadi pada sistem ketika *user* menekan tombol *logout* pada menu *accounts*. *Activity Diagram Logout Account* dapat dilihat apda gambar III.33



Gambar III.33 *Activity Diagram Logout Account*

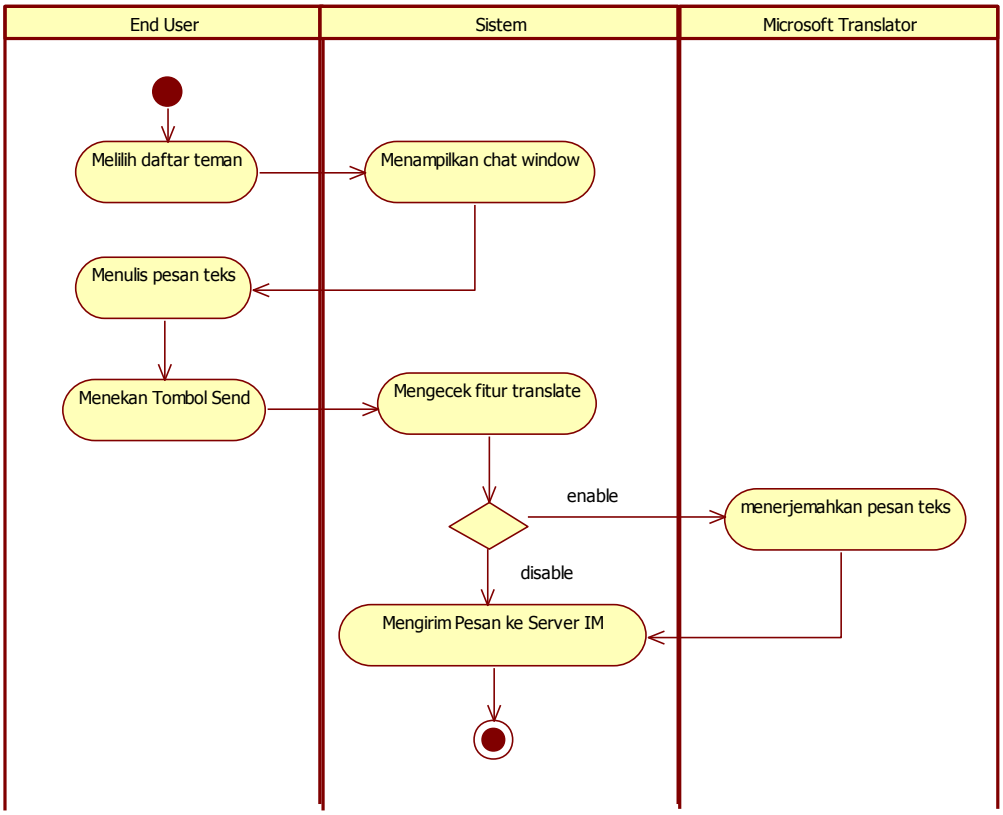
j. Activity Diagram Chat

Activity diagram Chat menggambarkan aktifitas yang terjadi ketika *user* menekan tommbol *chat*



Gambar III.34 *Activity Diagram Chat*

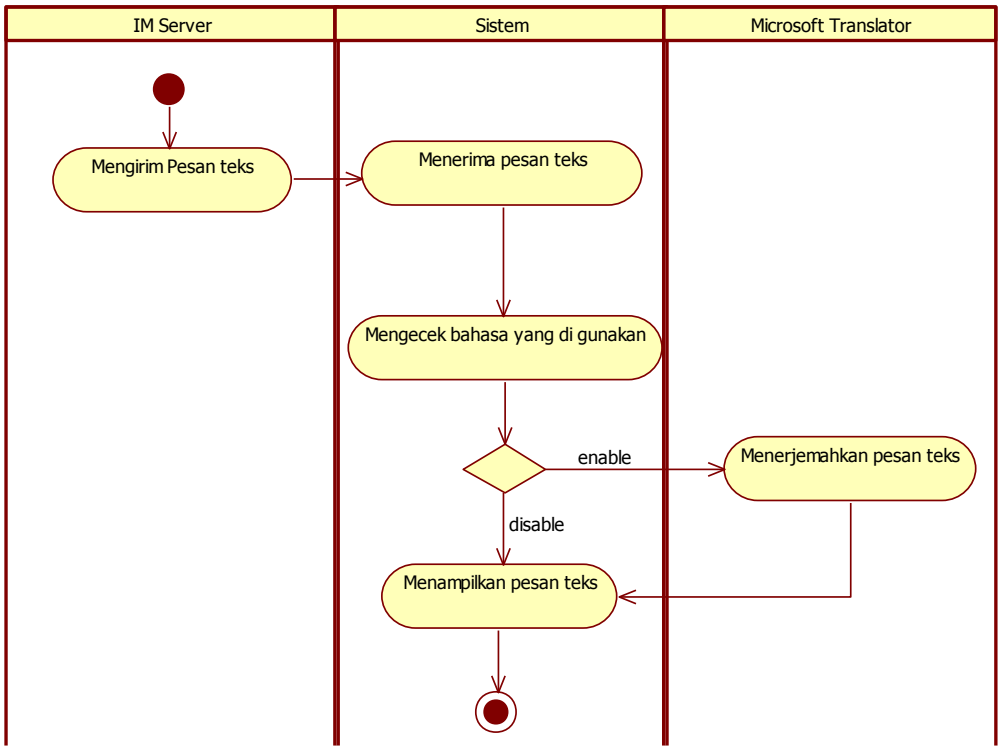
k. Activity Diagram Send Message



Gambar III.35 *Activity Diagram Send Message*

Gambar III.35 menggambarkan aktifitas yang terjadi pada sistem ketika *user* mengirimkan pesan pada *user* lain. Pesan akan dikirimkan oleh sistem ke *server* IM untuk di sampaikan ke target *user*.

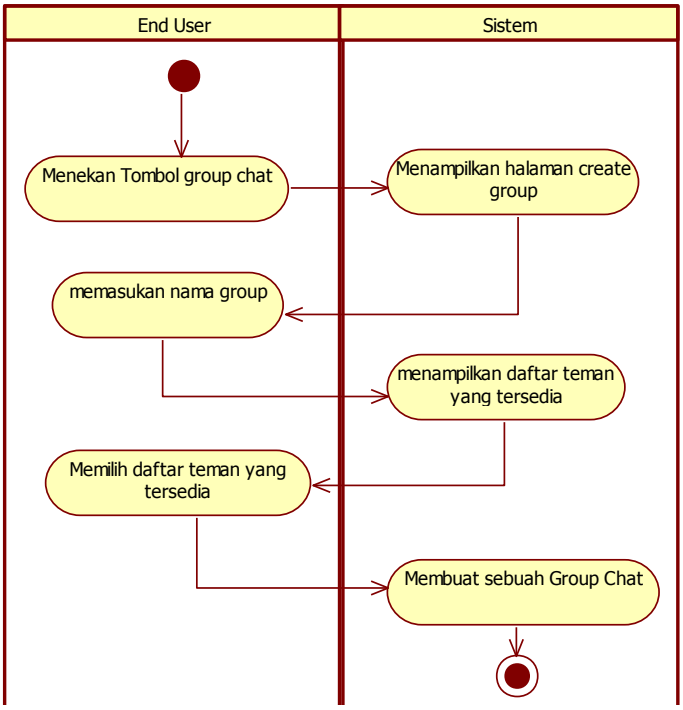
l. Activity Diagram Receive Message



Gambar III.36 *Activity Diagram Receive Message*

Gambar III.36 menggambarkan aktifitas yang terjadi pada sistem ketika sistem menerima pesan dari *user* lain. Pesan akan dikirimkan oleh sistem ke *Microsoft Translator* untuk di terjemahkan terlebih dahulu, kemudian akan di tampilkan ketika pesan sudah diterjemahkan .

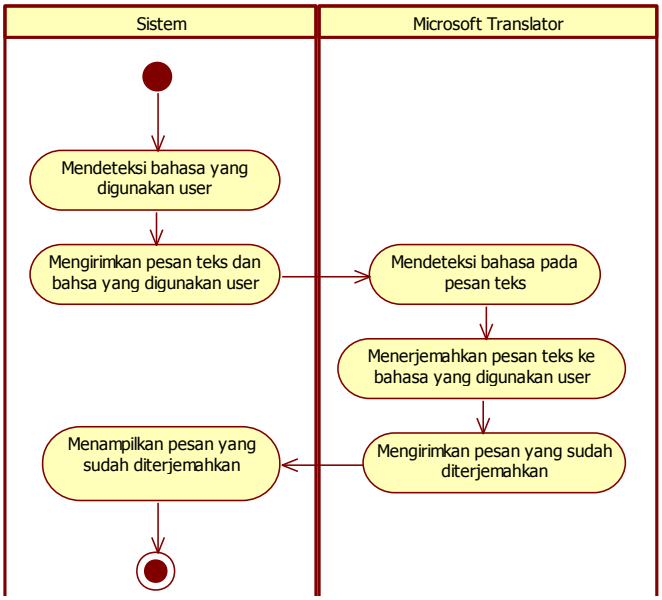
m. Activity Diagram Create Group



Gambar III.37 *Activity Diagram Crate Group*

Gambar III.37 menggambarkan aktifitas yang terjadi pada sistem ketika *user* membuat sebuah *group chat.*

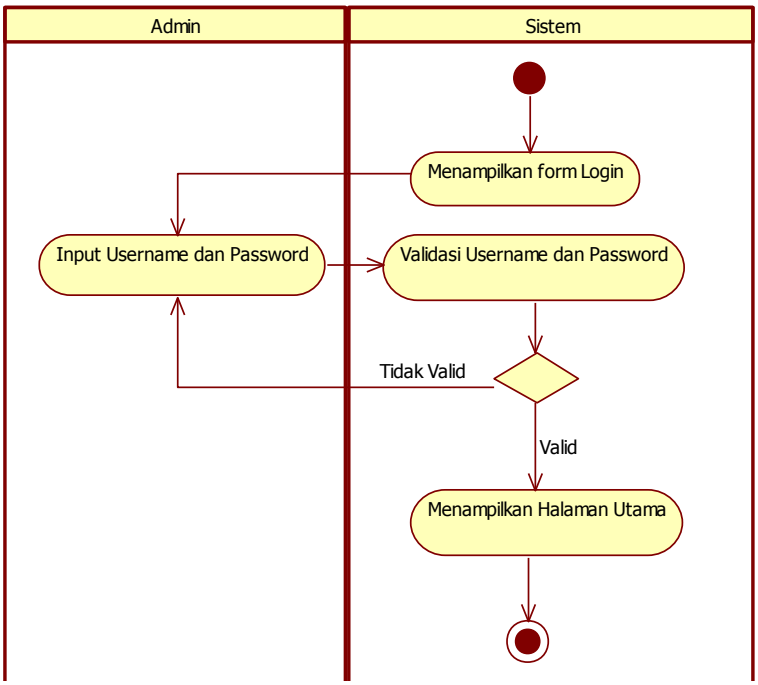
n. Activity Diagram Translate



Gambar III.38 *Activity Diagram Translate*

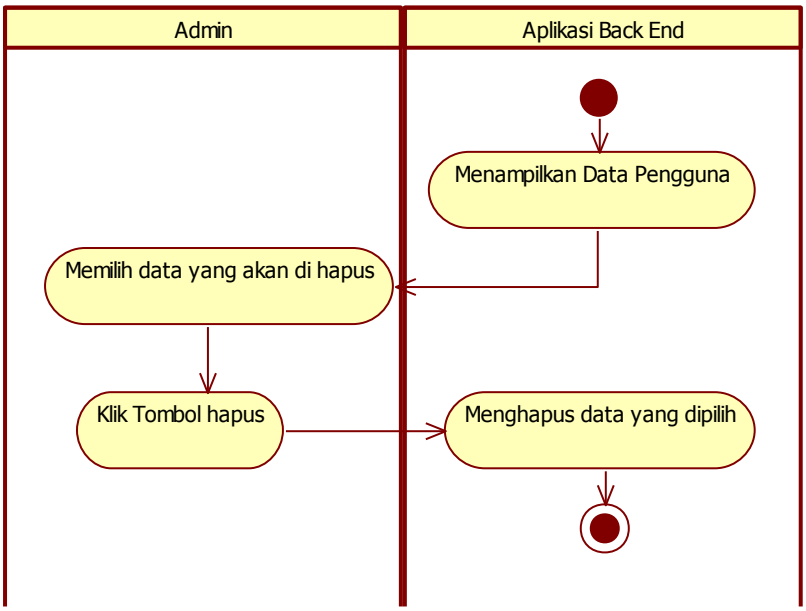
III.1.4.3.1 Activity Diagram Aplikasi *BackEnd*

a. Activity Diagram Login BackEnd



Gambar III.39 *Activity Diagram Login BackEnd*

b. Activity Diagram Delete User



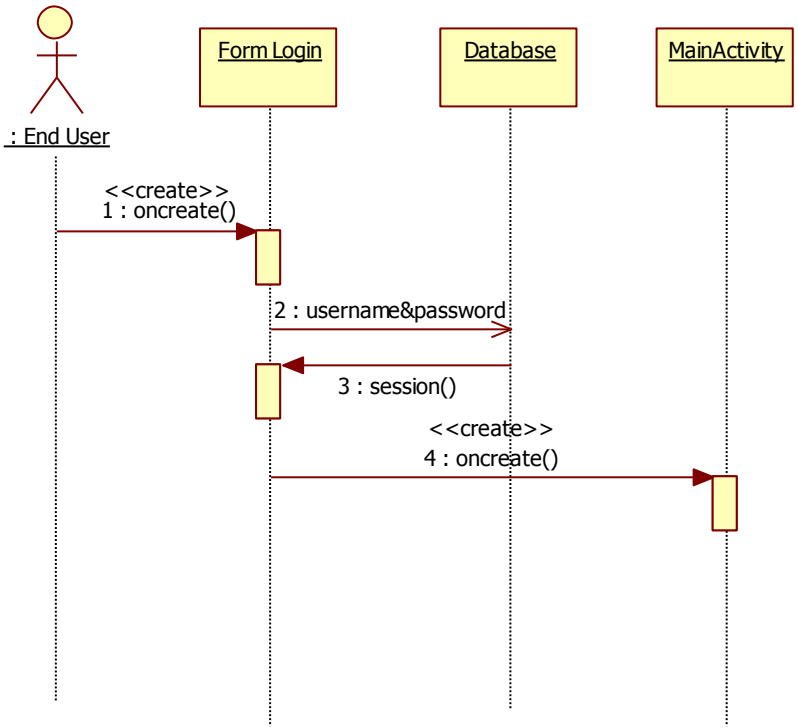
Gambar III.40 *Activity Diagram Delete User*

III.1.4.4 *Sequence Diagram*

Sequence diagram menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *Use Case*. Komponen utama *sequence diagram* terdiri atas obyek yang dituliskan dengan kotak segiempat. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progres vertikal.

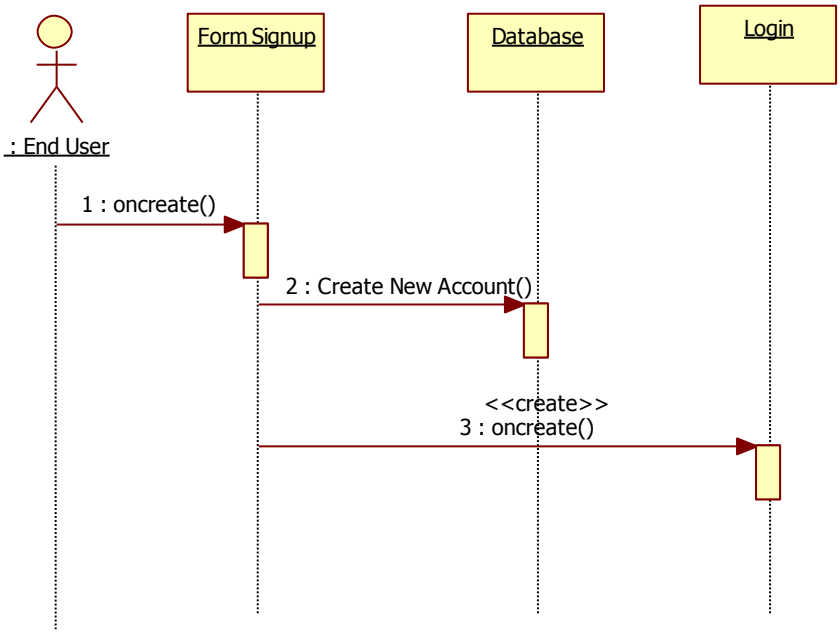
III.1.4.4.1 *Sequence Diagram* Aplikasi *FrontEnd*

a. *Sequence Diagram Login*



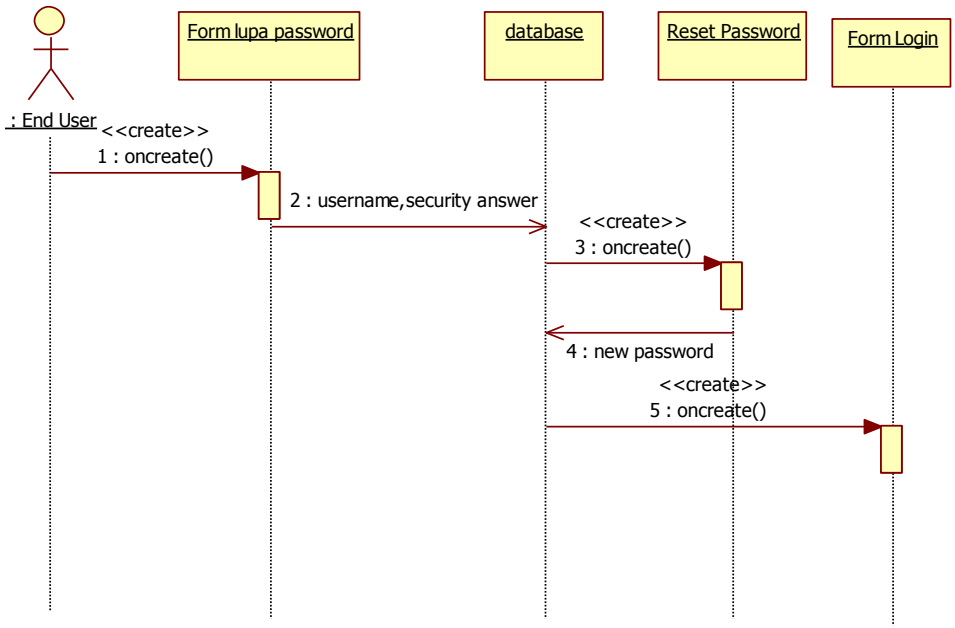
Gambar III.41 *Sequence Diagram Login*

b. Sequence Diagram Signup



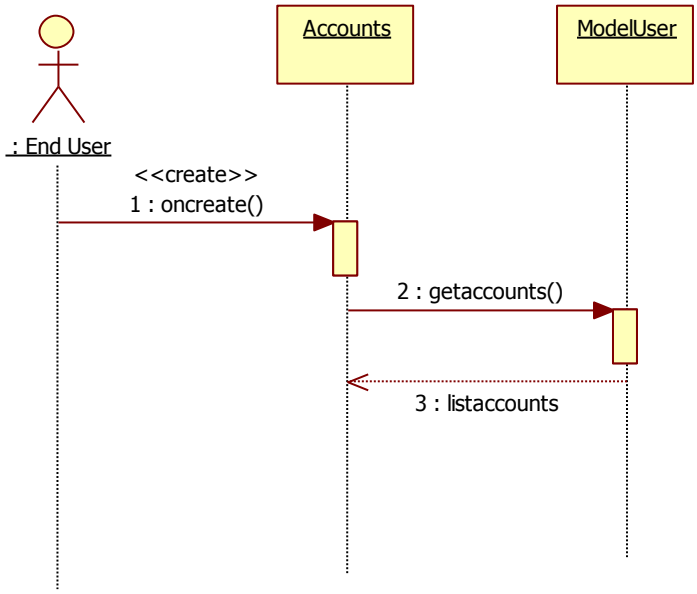
Gambar III.42 *Sequence Diagram Signup*

c. Sequence Diagram Lupa *Password*



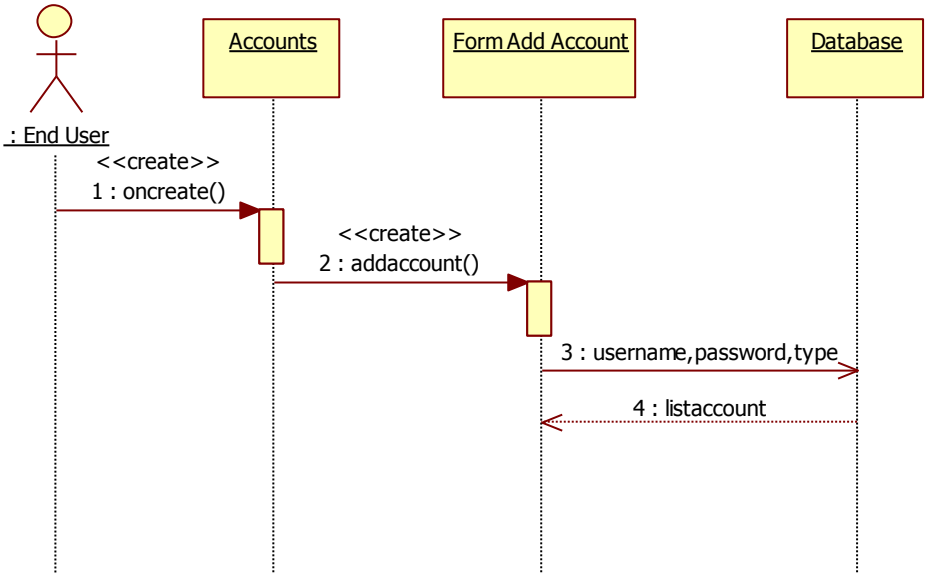
Gambar III.43 Sequence Diagram Lupa *Password*

d. Sequence Diagram Accounts



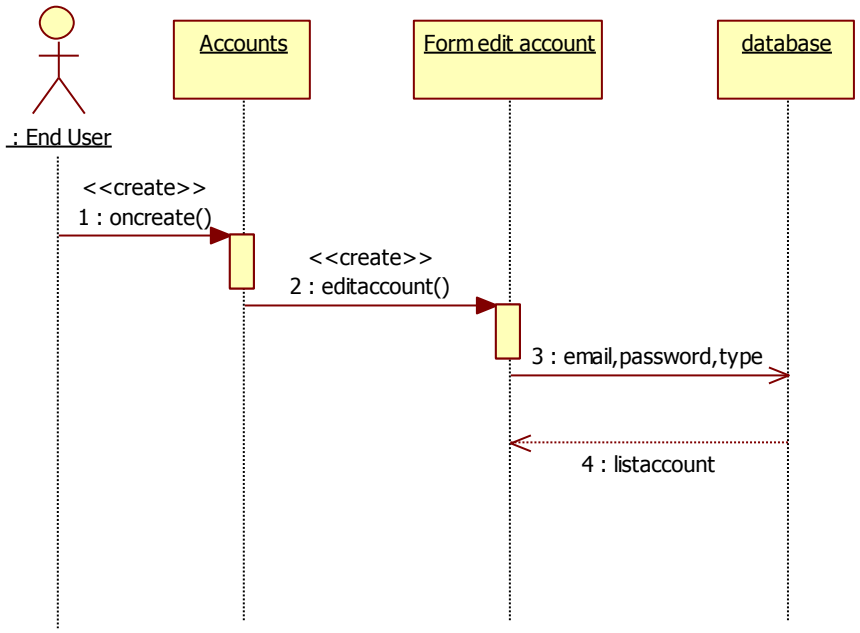
Gambar III.44 *Sequence Diagram Accounts*

e. Sequence Diagram Add Account



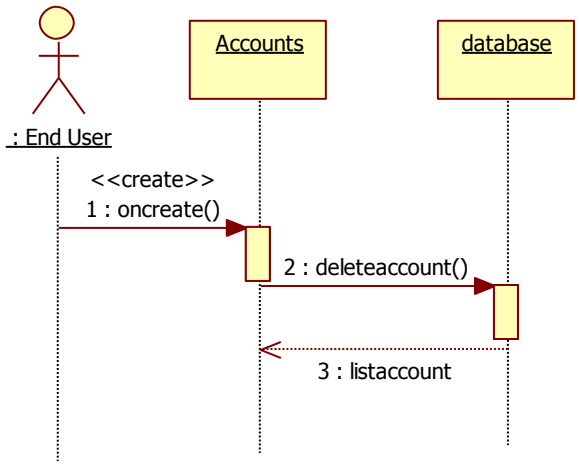
Gambar III.45 *Sequence Diagram Add Account*

f. Sequence Diagram Edit Account



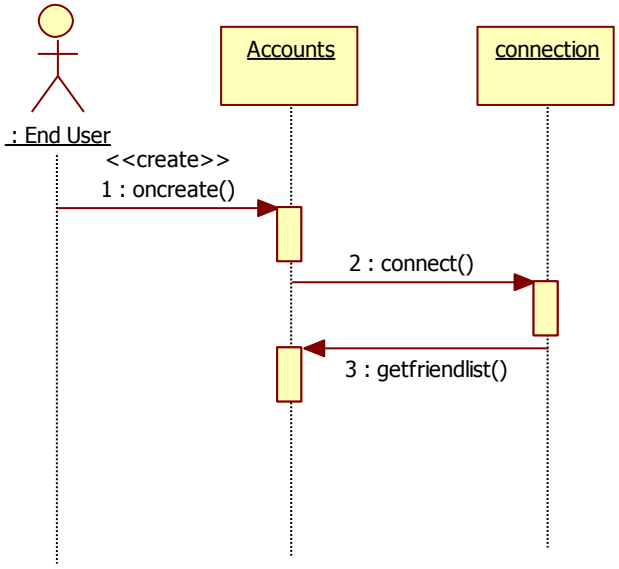
Gambar III.46 *Sequemce Diagram Edit Account*

g. Sequence Diagram Delete Account



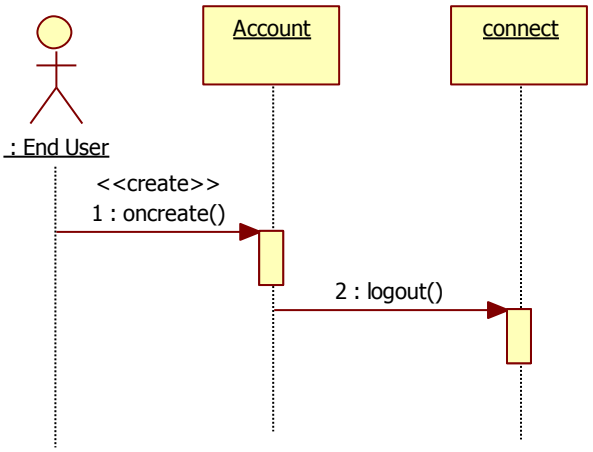
Gambar III.47 *Sequence Diagram Delete Account*

h. Sequence Diagram Login Account



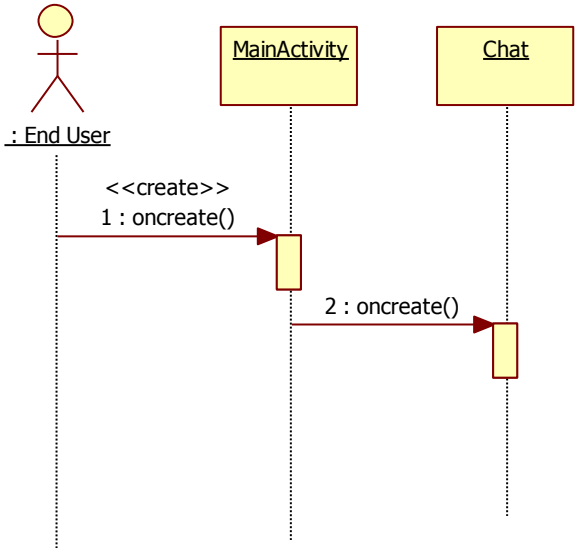
Gambar III.48 *Sequence Diagram Login Account*

i. Sequence Diagram Logout Account



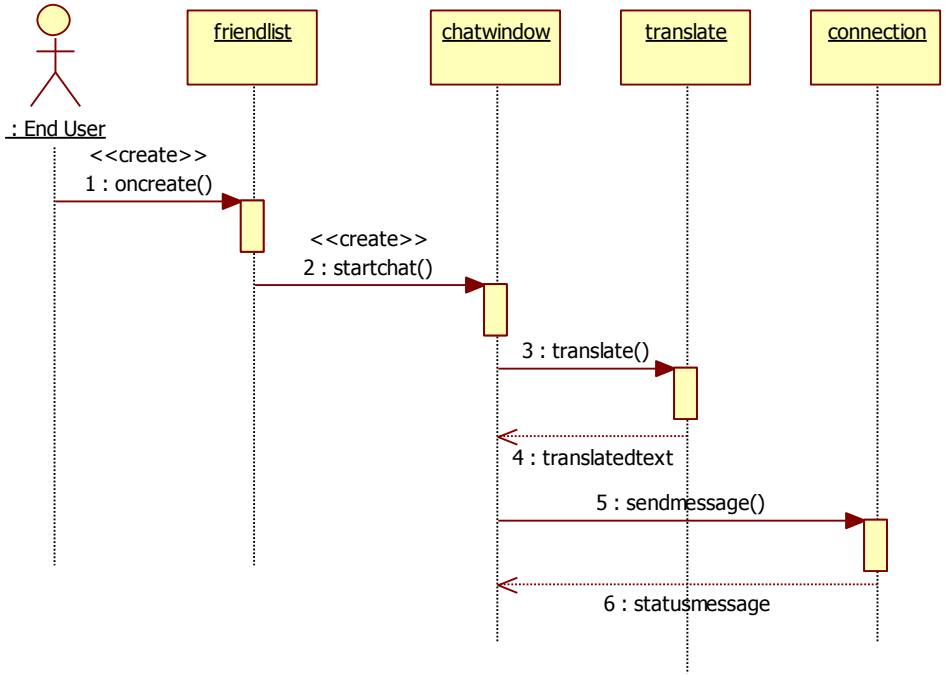
Gambar III.49 *Sequence Diagram Logout Account*

j. Sequence diagram chat



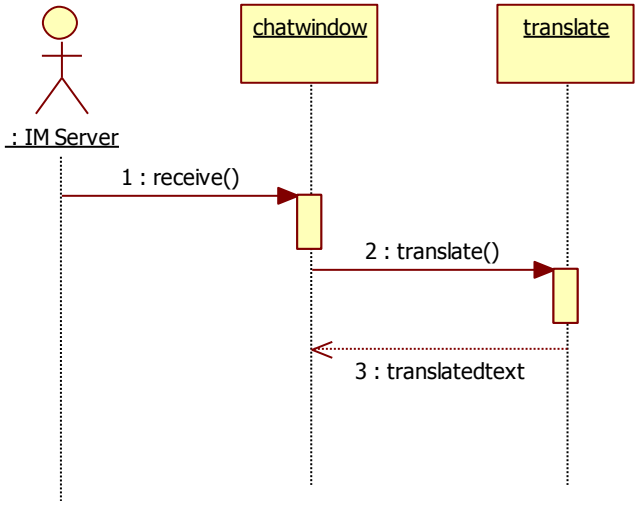
Gambar III.50 *Sequence Diagram Chat*

k. Sequence Diagram Send Message



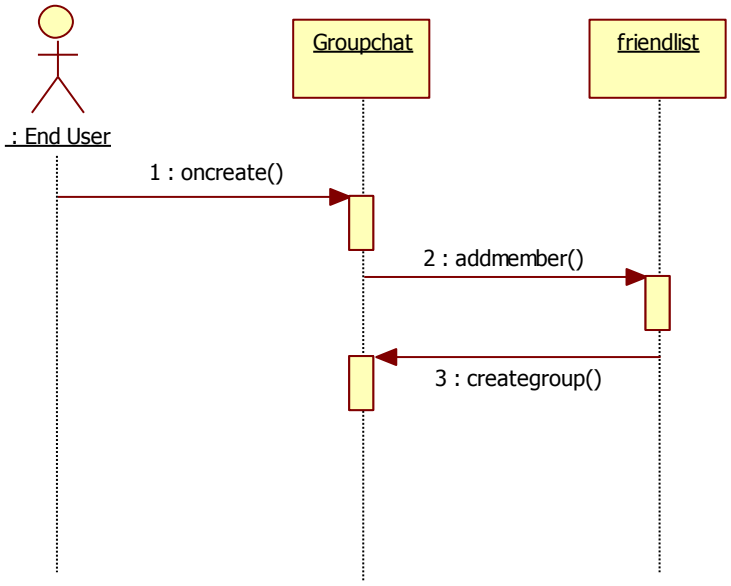
Gambar III.51 *Sequence Diagram Send Message*

l. Sequence Diagram Receive Message



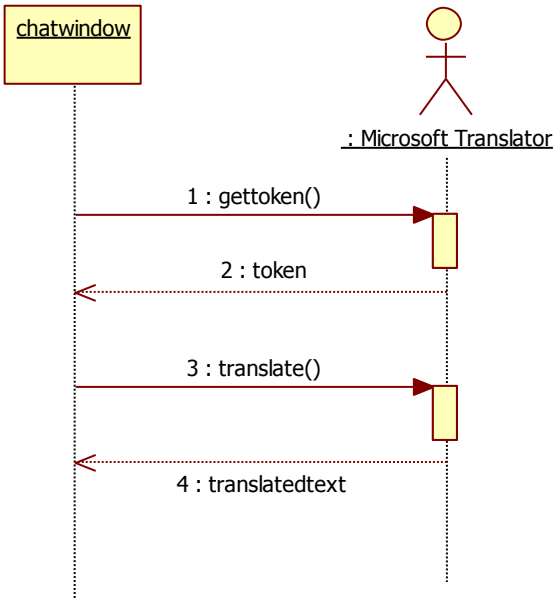
Gambar III.52 *Sequence Diagram Receive Message*

m. Sequence Diagram Groupchat



Gambar III.53 *Sequence Diagram Groupchat*

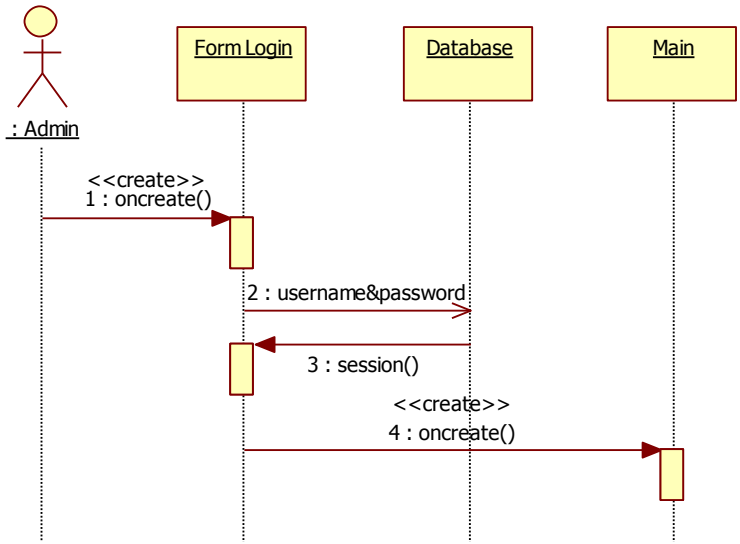
n. Sequence Diagram Translate



Gambar III.54 *Sequence Diagram Translate*

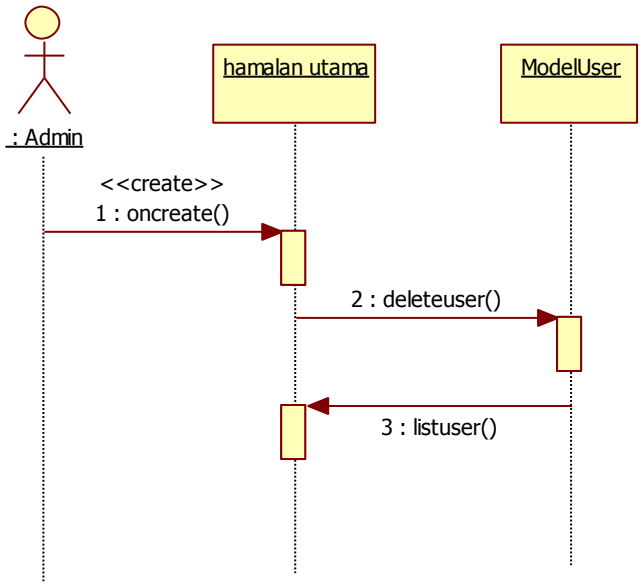
III.1.4.4.2 *Sequence Diagram* Aplikasi *BackEnd*

a. Sequence Diagram Login BackEnd



Gambar III.55 *Sequence Diagram Login BackEnd*

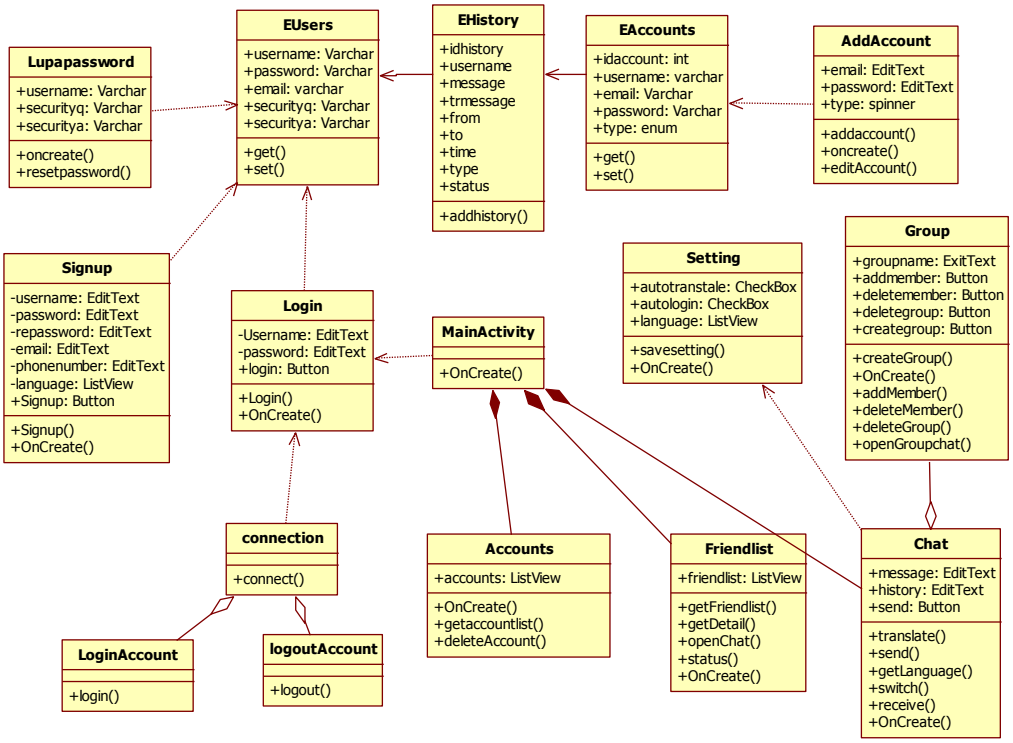
b. Sequence Diagram Delete User



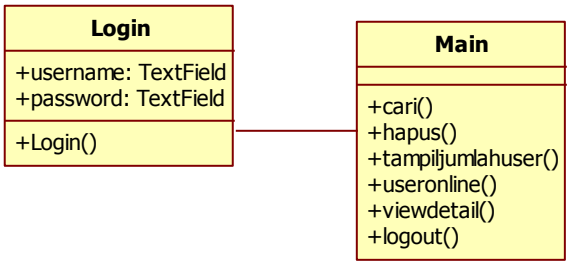
Gambar III.56 *Sequence Diagram Delete User*

1.1.4.4 *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 *class*. Hubungannya dari masing- masing *class* yang ada tersebut digambarkan seperti pada gambar III.57



Gambar III.57 *Class Diagram Frontend*



Gambar III.58 *Class Diagram BackEnd*

Tabel III.26 Deskripsi *Class Diagram*

Aplikasi *Frontend*		
Class	**Jenis Class**	**Deskripsi**
MainActivity	*Boundary*	Merupakan *class* yang digunakan untuk menampilkan Halaman utama
Signup	*Control*	*Class* yang berisi *method-method* untuk membuat suatu akun dalam sistem.

Login	*Control*	*Class* yang berisi *method-method* untuk masuk ke dalam sistem
Lupa*password*	*Control*	*Class* yang digunakan untuk mereset *password*
Setting	*Control*	*Class* yang berisi *method-method* untuk mengontrol pengaturan sistem
Accounts	*Control*	*Class* yang berisi *method-method* untuk mengatur account IM yang digunakan dalam sistem
AddAccount	*Control*	*Class* yang digunakan untuk menambahkan akun IM baru
Connection	*Control*	*Class* yang digunakan untuk membuat sebuah koneksi ke IM sever
Loginaccount	*Control*	*Class* yang digunakan untuk autentikasi ke IM *server*
LogoutAccount	*Control*	*Class* yang digunakan untuk *logout* dari IM *server*
Friendlist	*Control*	*Class* untuk menampilkan daftar teman yang tersedia
Chat	*Control*	*Class* yang berisi *method-method* untuk *chatting* dengan *user* lain
Group	*Control*	*Class* yang digunakan untuk membuat sebuah *group chat*
EUser	*Entity*	*Class* yang digunakan untuk menyimpan *Database user* yang telah terdaftar dalam sistem
EHistory	*Entity*	*Class* yang digunakan untuk menyimpan riwayat percakapan
EAccount	*Entity*	*Class* yang digunakan untuk menyimpan akun IM pengguna

Aplikasi *Backend*		
Login	*Control*	*Class* yang berisi *method-method* untuk masuk ke dalam sistem *Backend*
Main	*View*	*Class* yang menampilkan *Database user*

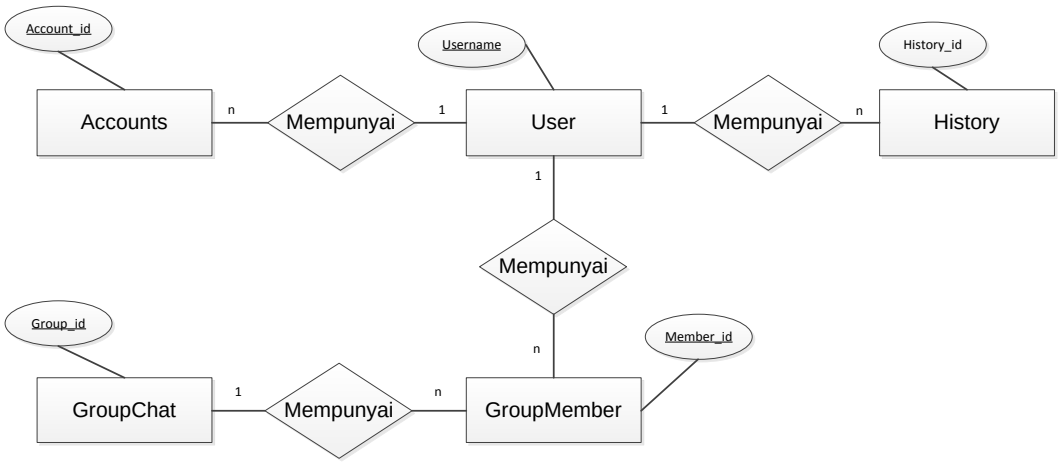
III.2 Perancangan Sistem

Perancangan sistem bertujuan untuk menspesifikasikan aspek-aspek teknik yang menjadi solusi dalam perencanaan. Pada tahap ini perancangan akan didefinisikan secara detail untuk mengatasi masalah-masalah yang lebih teknis, berkaitan dengan kegiatan implementasi seperti perancangan arsitektur sistem, perancangan *Database*, perancangan antarmuka, dan perancangan pesan.

III.2.1 Perancangan Data

Perancangan data merupakan tahapan untuk memetakan model konseptual ke model basis data yang akan digunakan. Perancangan data dari aplikasi *mobile* merupakan perancangan struktur tabel dari tabel kategori. Struktur tabel kategori menggambarkan detail tabel yang berisi *field*, tipe data, panjang data, dan keterangan lainnya. Berikut ini gambaran struktur tabel pengaturan *Database sistem (SQLite)* dan *Database web Service*

III.2.1.1 ERD



Gambar III.59 Diagram ERD

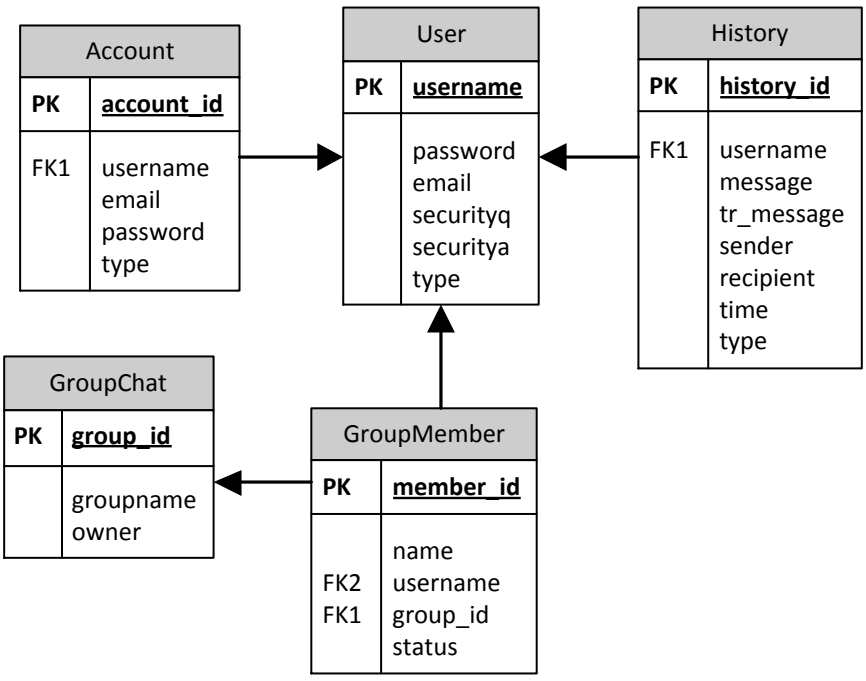
III.2.1.2 Skema Relasi

Tabel III.27 Skema Relasi

Nama Tabel	Attribut
User	*Username, password, email, securitya, securityq, type*
Account	*Account_id, username, email, password, type*
History	*History_id, username, message, tr_message, sender, recipient, time, type*
GroupChat	*Group_id, groupname, owner*
GroupMember	*Member_id, name, username, group_id, status*

III.2.1.3 Diagram Relasi

Relasi antar tabel merupakan gabungan antar *file* yang mempunyai kunci utama yang sama, sehingga *file-file* tersebut menjadi satu kesatuan yang dihubungkan oleh *field* kunci (*Primary Key*). Pada proses ini elemen-elemen data dikelompokkan menjadi satu *file Database* beserta entitas dan hubungannya. Skema relasi ini dapat dilihat pada Gambar III.60.



Gambar III.60 Diagram Relasi

III.2.1.3 Struktur Tabel

Perancangan struktur tabel adalah perancangan tabel-tabel yang akan digunakan pada *Database*. Tabel-tabel yang terdapat dalam *Database* yang digunakan dalam sistem ini adalah :

Tabel III.28 Struktur Tabel *User*

Nama Field	Tipe	Panjang	Keterangan
*User*name	*Varchar*	30	*Primary key*
Password	*Varchar*	30	
Securityq	*Varchar*	100	
Securitya	*Varchar*	100	
Type	*Enum*	1	

Tabel III.29 Struktur Tabel *Account*

Nama Field	Tipe	Panjang	Keterangan
Account_id	*Int*		*Primary key*
*User*name	*Varchar*	30	*Foreign key reference user*(*username*)
Email	*Varchar*	50	
password	*Varchar*	30	
Type	*Enum*		

Tabel III.30 Struktur Tabel *History*

Nama Field	Tipe	Panjang	Keterangan
History_id	*Int*		*Primary key*
Username	*Varchar*	30	*Foreign key reference user*(*username*)
message	*Varchar*	1000	
tr_message	*Varchar*	100	
sender	*Varchar*	50	
recipient	*Varchar*	50	

time	Date		
type	Enum		

Tabel III.31 Struktur Tabel *GroupChat*

Nama Field	Tipe	Panjang	Keterangan
group_id	*Int*		*Primary key*
groupname	*Varchar*	20	
owner	*Varchar*	50	

Tabel III.32 Struktur Tabel *GroupMember*

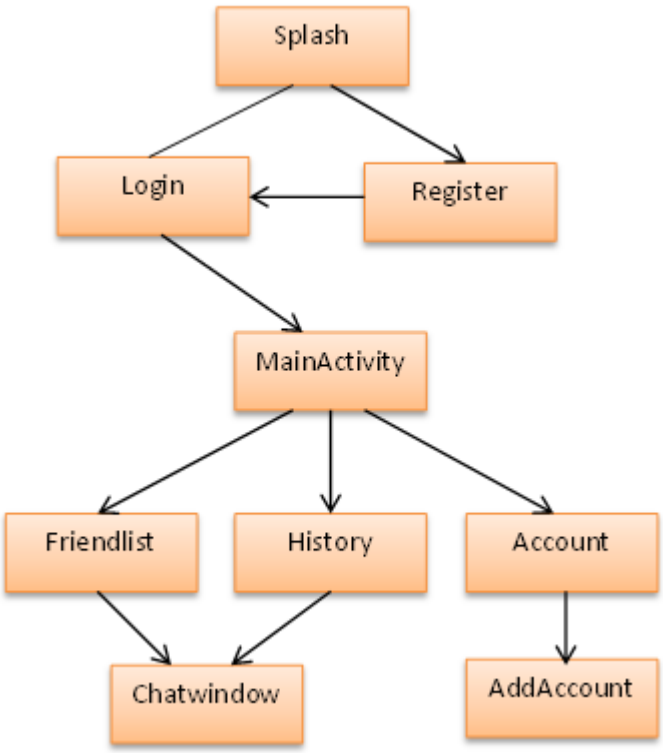
Nama Field	Tipe	Panjang	Keterangan
Member_id	*Int*		*Primary key*
name	*Varchar*	30	
*user*_id	*Varchar*	50	
group_id	*Int*		
status	*Enum*		

III.2.2 Perancangan Menu

Perancangan struktur menu merupakan gambaran jalur pemakaian aplikasi. Perancangan struktur menu dari aplikasi ini dapat dilihat pada gambar berikut ini.

III.2.2.1 Struktur Menu *FrontEnd*

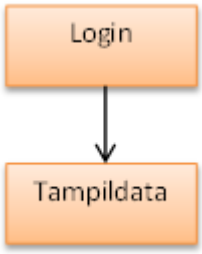
Perancangan struktur menu merupakan gambaran jalur pemakaian aplikasi *user* pengguna aplikasi *mobile* ini. Perancangan struktur menu dari aplikasi ini dapat dilihat pada gambar III.61



Gambar III.61 Struktur Menu Aplikasi *FrontEnd*

III.2.2.2 Struktur Menu Back End

Sedangkan struktur menu pada aplikasi Back End dapat dilihat pada gambar III.62



Gambar III.62 Struktur Menu Aplikasi BackEnd

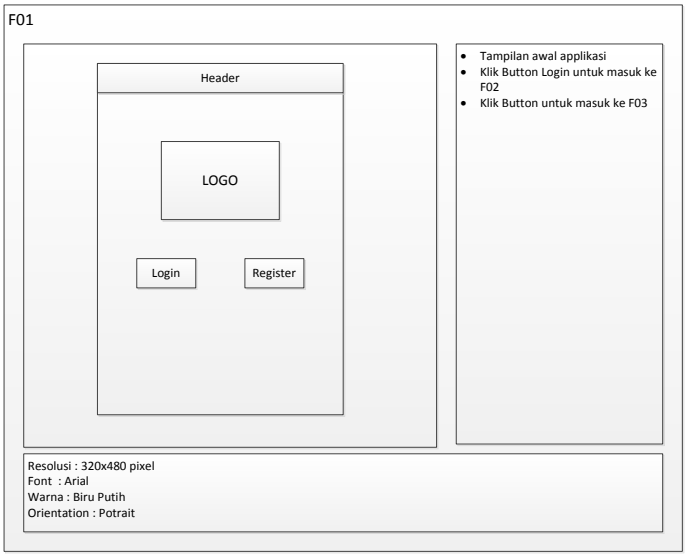
III.2.3 Perancangan Antarmuka Sistem

Tahap perancangan antarmuka sistem bertujuan untuk mencari bentuk yang optimal dari aplikasi yang akan dibangun dengan pertimbangan faktor-faktor permasalahan dan kebutuhan yang ada pada sistem seperti yang telah ditetapkan pada tahap analisis. Dalam tahap ini upaya yang dilakukan yaitu dengan cara mengkombinasikan penggunaan teknologi perangkat keras dan perangkat lunak yang tepat sehingga diperoleh yang optimal dan mudah diimplementasikan.

Perancangan antarmuka sistem terbagi menjadi dua bagian yaitu antarmuka aplikasi *frontend* dan antarmuka aplikasi *backend*.

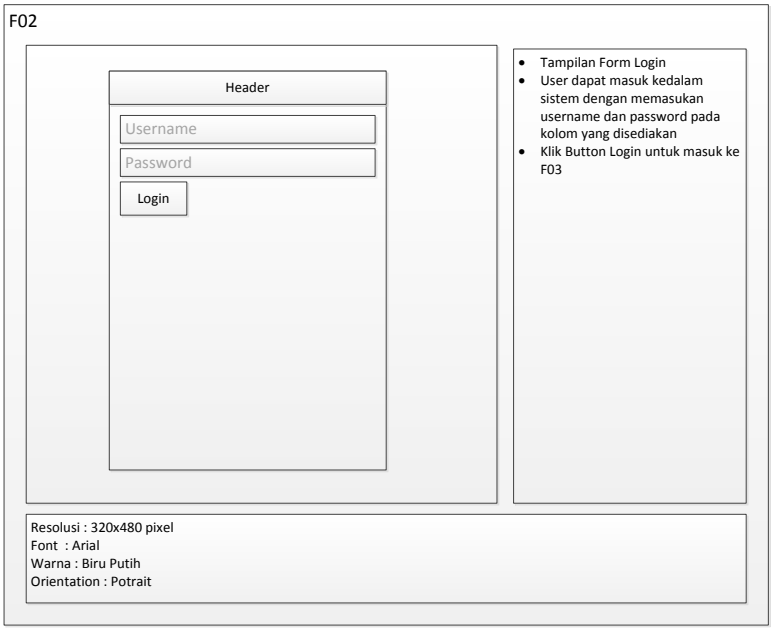
III.2.3.1 Perancangan Antarmuka Aplikasi FrontEnd

a. Perancangan Antarmuka *Splash*



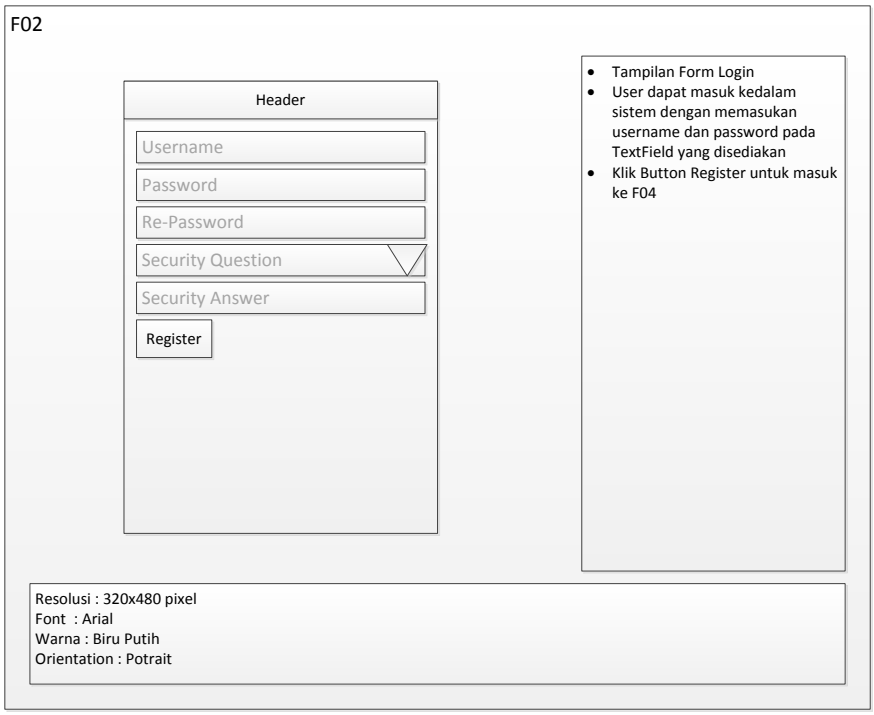
Gambar III.63 Antarmuka *Splash*

b. Perancangan Antarmuka L*ogin*



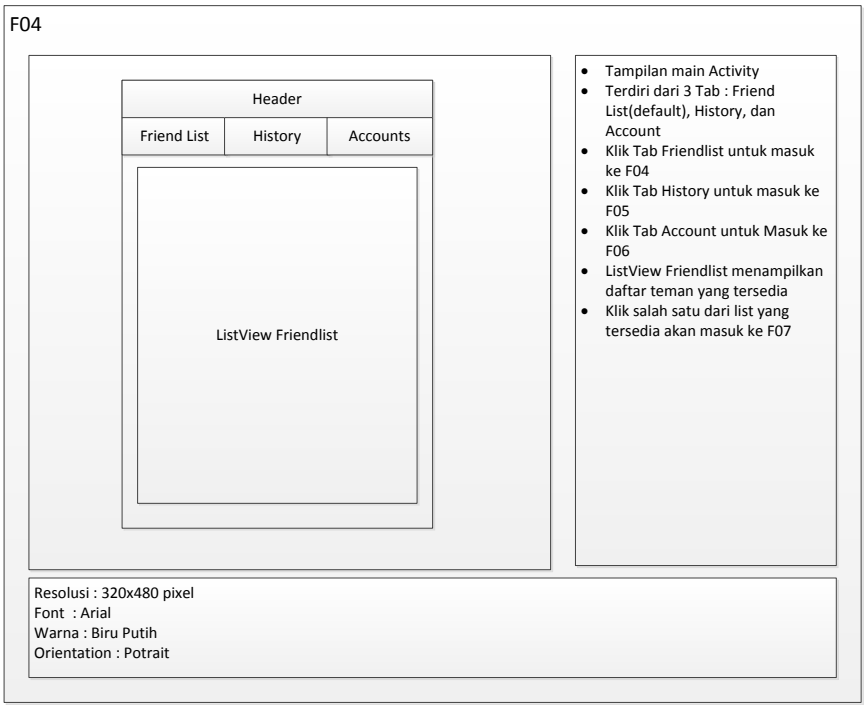
Gambar III.64 Antarmuka *Login*

c. Perancangan Antarmuka *Register*



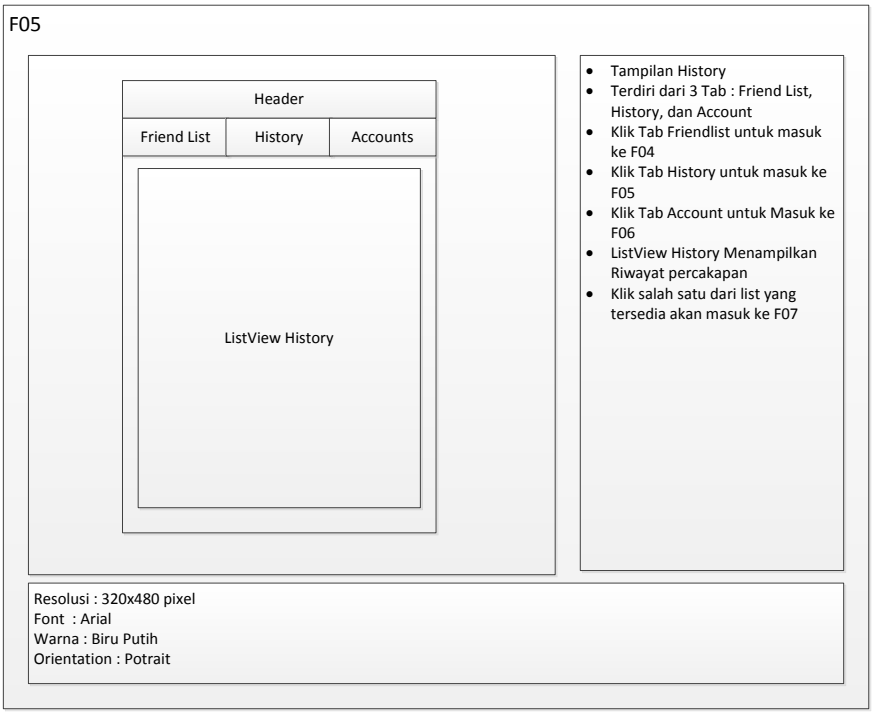
Gambar III.65 Antarmuka *Register*

d. Perancangan Antarmuka *Main Activity*



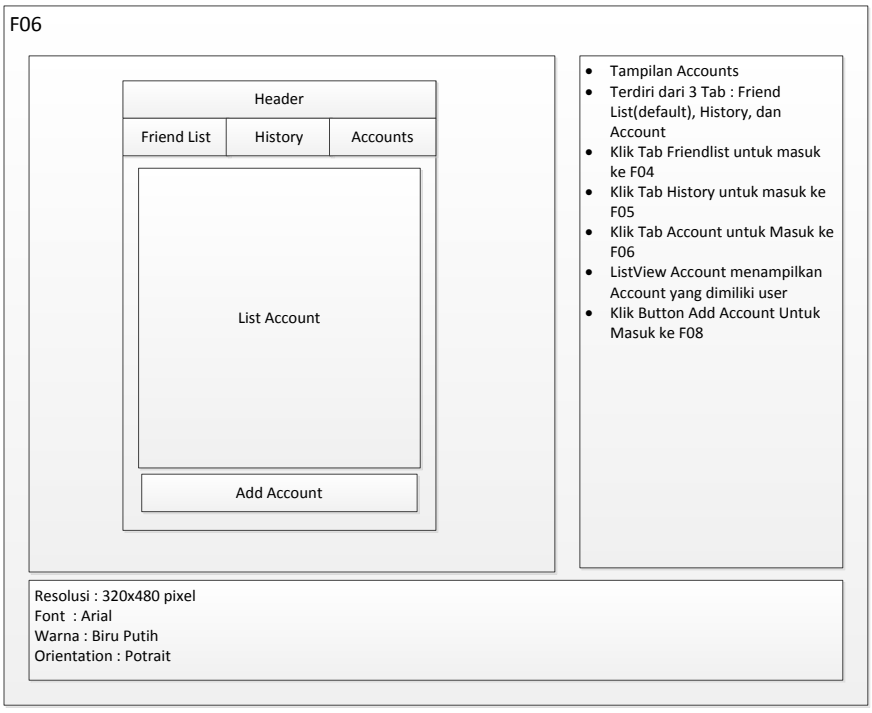
Gambar III.66 Antarmuka *Main Activity*

e. Perancangan Antarmuka *History*



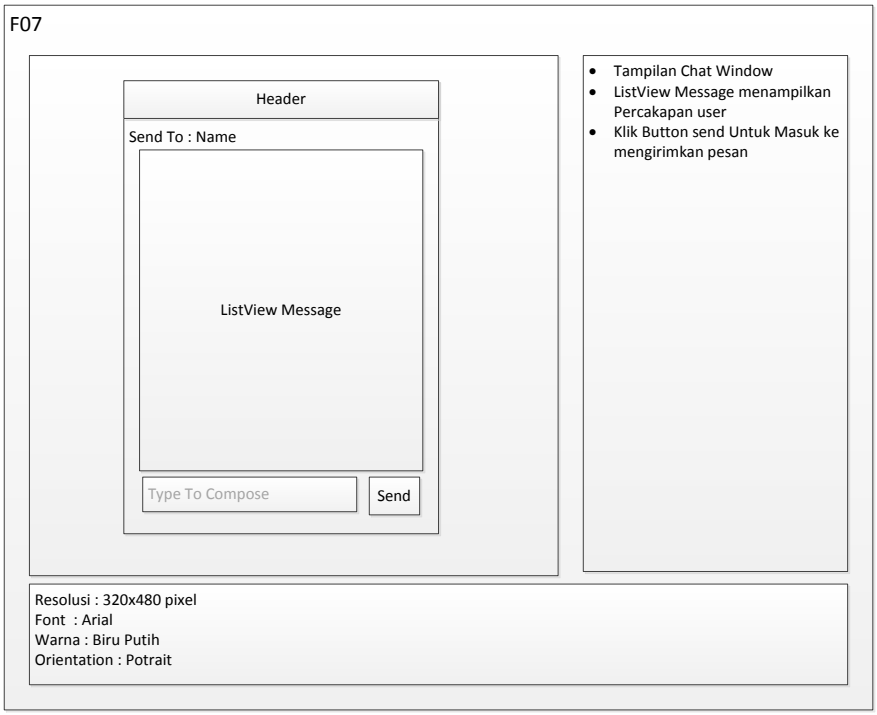
Gambar III.67 Antarmuka *History*

f. Perancangan Antamuka *Accounts*



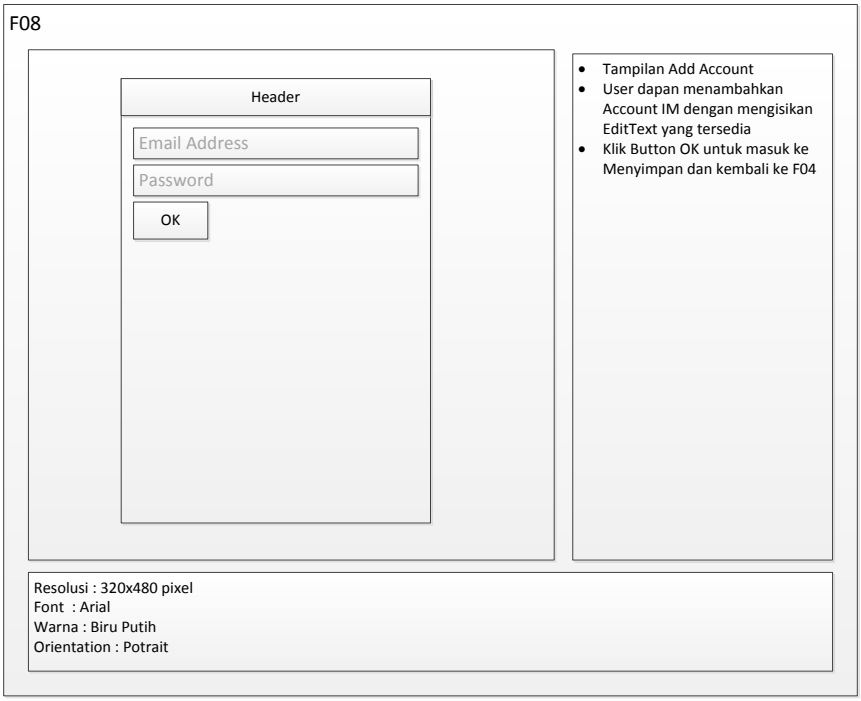
Gambar III.68 Antarmuka *Accounts*

g. Perancangan Antarmuka *Chat Window*



Gambar III.69 Antarmuka *Chat Window*

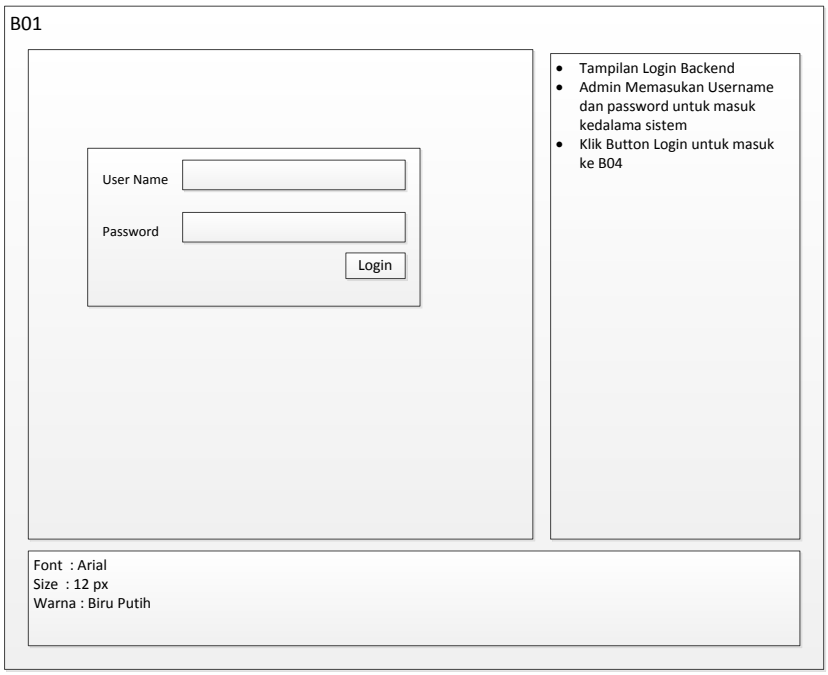
h.Perancangan Antarmuka *Add Account*



Gambar III.70 Antarmuka *Add Account*

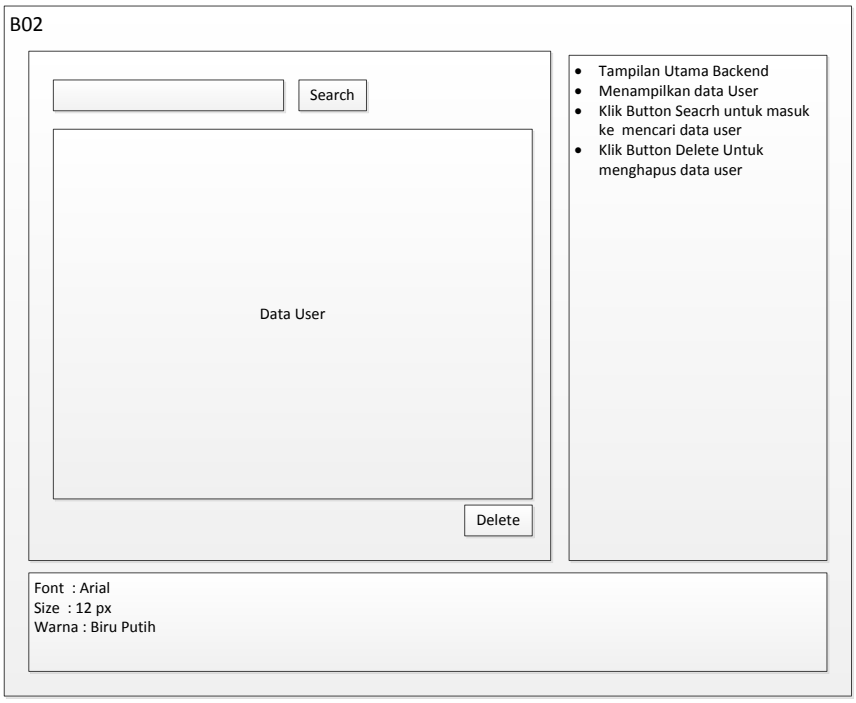
III.2.3.2 Perancangan Antarmuka Aplikasi *Backend*

a. Perancangan Antarmuka *Login*



Gambar III.71 Antarmuka *Backend Login*

b. Perancangan antarmuka *Backend* Menu Utama



Gambar III.72 Antarmuka *Backend* Menu Utama