BAB 2
LANDASAN TEORI

2.1 Lembar Jawaban Komputer

Lembar Jawaban Komputer (LJK) merupakan formulir khusus yang telah diformat dan umumnya memiliki *barcode* agar dapat dipindai menggunakan alat pemindai khusus untuk membaca informasi pada LJK tersebut. LJK umumnya digunakan untuk ujian pilihan ganda, kuesioner, dan kegiatan pendataan dengan tujuan untuk mempercepat pengolahan data [9]. Meskipun begitu, kecepatan tersebut tergantung dari spesifikasi alat pemindai yang digunakan. Pada awal prakteknya, LJK digunakan untuk kepentingan Ujian Nasional (UN) dengan daya tampung 50 jawaban Pilihan Ganda (PG) serta cara pengisian dengan menghitamkan bulatan pada LJK.

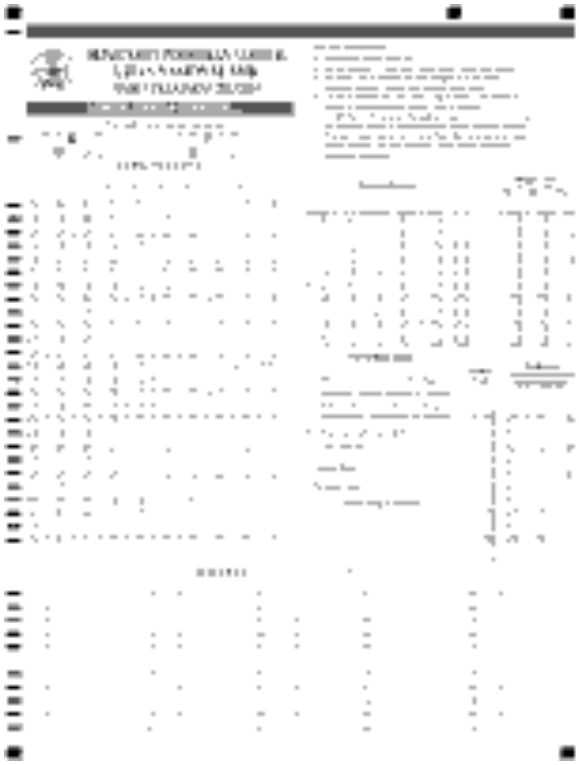
LJK yang digunakan di sekolah di Indonesia saat ini memiliki berbagai macam variasi. Variasi tersebut muncul disebabkan oleh LJK yang sekarang ini digunakan oleh masing – masing sekolah secara pribadi. Dari cara pengisian, LJK saat ini dapat diisi dengan cara menghitamkan bulatan atau menyilang kotak. Sedangkan pada bagian jawaban, daya tampung LJK bervariasi dari 40 hingga 100 jawaban PG dan memiliki bagian jawaban *essay*. Gambar 2.1 adalah contoh LJK yang digunakan oleh SMAN M***** Kab. B*****.

Gambar 2.1 Contoh LJK SMAN M***** Kab. B*****

2.2 Optical Mark Recognition

Optical Mark Recognition (OMR) merupakan metode yang digunakan untuk pengumpulan data dengan cara mengidentifikasi tanda (silang, bulatan hitam, atau ceklis) pada lembar formulir hasil cetakan [21]. Istilah OMR mulai digunakan saat teknologi optik untuk membaca tanda pertama kali dikenalkan pada tahun 1962 [16]. Perangkat optik tersebut mendeteksi tanda dengan cara menangkap refleksi cahaya dari lembar formulir. Konsep inilah yang kemudian melahirkan metode OMR.

OMR pada LJK bekerja mengidentifikasi tanda dengan cara mendeteksi intensitas cahaya yang dipantulkan oleh lembar formulir [22]. Sehingga berdasarkan Hukum Pemantulan Cahaya dapat disimpulkan bahwa bagian yang memiliki tanda akan merefleksikan lebih sedikit cahaya. Teknologi OMR umumya diimplementasikan pada *scanner* khusus yang bertugas untuk memindai lembar formulir [23]. Selain itu, lembar formulir yang digunakan untuk *scanner* OMR umumnya merupakan lembar formulir khusus yang memiliki penanda pada tepian atau biasa disebut *landmark*. *Landmark* ini membantu OMR pada *scanner* mengenali area yang ada pada lembar formulir sehingga pemrosesan menjadi lebih cepat [24]. Gambar 2.2 adalah contoh LJK yang memiliki *landmark* di tepinya.



Gambar 2.2 Contoh LJK dengan landmark

Sumber:
https://www.kompasiana.com/digitalsense/54f9776ea333111a648b46a2/apa-sih-ljk-itu

Setelah proses pengenalan objek pada lembar formulir, tahap selanjutnya proses OMR pada kasus LJK adalah melakukan penilaian. Proses penilaian terhadap LJK dilakukan dengan cara membandingkan lembar jawaban dengan kunci jawaban. Maka dari itu sebelum proses pembacaan lembar jawaban, informasi kunci jawaban sudah terlebih dahulu disimpan. Perbandingan yang dilakukan adalah pola PG pada lembar jawaban dengan kunci jawaban. Setiap titik yang sama merupakan kategori jawaban benar, sedangkan titik yang berbeda merupakan kategori jawaban salah.

2.3 Citra

Citra atau lebih tepatnya citra digital merupakan sebuah *array* piksel yang tersusun dalam bidang dua dimensi dan memiliki baris serta kolom [25]. Citra dapat terbagi menjadi dua jenis, yaitu raster dan vektor.

Sedangkan berdasarkan jenis warnanya citra dapat terbagi diantaranya sebagai berikut:

1. Citra biner

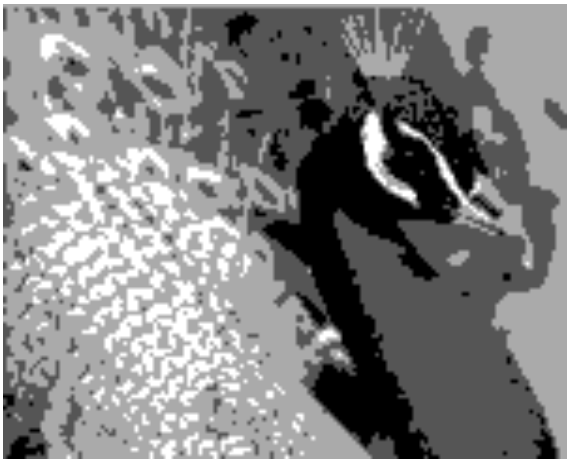
   Citra biner merupakan citra yang pikselnya terdiri dari angka 0 dan 1. Angka 0 merepresentasikan warna hitam sedangkan angka 1 merepresentasikan warna putih, karena itulah citra biner akan berwarna hitam putih. Gambar 2.3 merupakan salah satu contoh citra biner.

Gambar 2.3 Contoh Citra Biner

Sumber: https://www.trivusi.web.id/2022/09/image-processing.html

2. Citra *grayscale*

   Citra *grayscale* merupakan citra yang pikselnya terdiri dari bilangan bulat antara 0 hingga 255. Angka 0 merepresentasikan warna hitam dan angka 255 merepresentasikan warna putih, sedangkan angka antara 0 hingga 255 merepresentasikan tingkat keabuan [26]. Sehingga dapat disimpulkan angka yang mendekati 0 akan menghasilkan warna yang lebih gelap, sedangkan angka yang mendekati 255 akan menghasilkan warna yang lebih terang. Gambar 2.4 merupakan salah satu contoh citra *grayscale*.

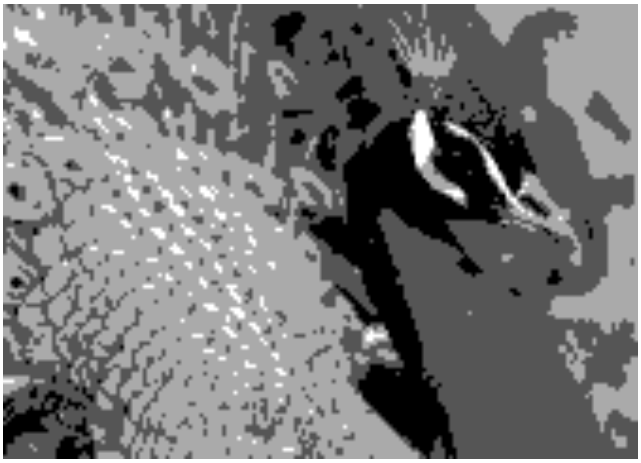


Gambar 2.4 Contoh Citra *Grayscale*

Sumber: https://medium.com/@rndayala/gray-scale-images-8d6aacb3b761

3. Citra RGB

Citra RGB merupakan citra yang setiap pikselnya memiliki tiga kanal warna, yaitu *Red* (Merah), *Green* (Hijau), dan *Blue* (Biru) [27]. Warna yang dihasilkan pada setiap pikselnya ditentukan oleh gabungan dari intensitas ketiga warna tersebut. Ketiga warna tersebut pun masing – masing memiliki rentang warna dari 0 hingga 255. Citra RGB umumnya disimpan dalam format 24 bit dimana warna merah, hijau, dan biru masing – masing memiliki 8 bit. Format ini memungkinkan citra RGB untuk menghasilkan sekitar 16 juta warna, karena itulah citra RGB sering juga disebut citra *truecolor* atau citra yang memiliki tingkat ketepatan warna tinggi dengan dunia nyata. Gambar 2.5 merupakan salah satu contoh citra RGB.



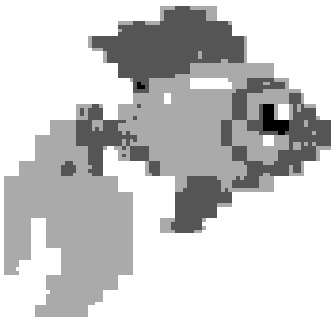
Gambar 2.5 Contoh Citra RGB

Sumber: https://medium.com/@rndayala/gray-scale-images-8d6aacb3b761

2.3.1 Piksel

Piksel merupakan unit logika terkecil yang ada pada sebuah citra [28]. Dimensi pada citra dapat memberikan informasi seberapa banyak piksel yang ada pada citra tersebut. Contohnya, citra dengan dimensi 500x500 memiliki 500 baris piksel dan 500 kolom piksel, sehingga citra tersebut memiliki total 250000 piksel. Piksel sendiri merupakan unit yang umumnya memiliki tiga kanal warna, yaitu *Red* (Merah), *Green* (Hijau), dan *Blue* (Biru). Masing – masing kanalnya memberikan tingkat intensitas yang berbeda – beda dan tingkat intensitas tersebut dideklarasikan oleh nilai dari 0 hingga 255. Semakin nilainya mendekati 0 maka warna yang dihasilkan akan semakin gelap dan semakin nilainya mendekati 255 maka warna yang dihasilkan akan semakin terang. Gabungan ketiga warna dengan tingkat intensitas yang berbeda tersebut dapat menghasilkan hingga 16 juta warna. Untuk dapat melihat piksel sendiri dapat dilakukan dengan memperbesar citra hingga terlihat bentuk persegi pada citra tersebut. Bentuk persegi ini merupakan piksel yang membentuk citra tersebut, maka dari itulah semakin banyak piksel

yang dimiliki oleh sebuah citra semakin jelas dan tajam juga citra tersebut. Contoh bentuk piksel dapat dilihat pada Gambar 2.6.



Gambar 2.6 Contoh Bentuk Piksel

Sumber: https://play-lh.googleusercontent.com/RpDJ23us5d4GoUMI1cgv_I0VSK-t_88ZNx1vbBjte0wnBJ3pfWzNwpfmzM7swOiovQ

2.3.2 Citra Raster

Citra raster merupakan citra yang tersusun oleh piksel. Citra jenis ini adalah citra yang paling umum beredar di internet. Citra raster sering disebut juga citra bitmap dan memiliki ciri, yaitu tersusun atas piksel dan mudah diberi efek khusus, seperti *distortion*, *blur*, dan *grain*.

Format *Joint Photographic Expert Group* (JPEG/JPG) merupakan format raster yang paling banyak digunakan saat ini. Format ini merupakan format dengan kedalaman 24 bit dan umumnya digunakan untuk keperluan *web* serta fotografi. Ekstensi yang terdapat pada citra JPEG/JPG umumnya adalah .jpeg/.jpg.

2.4 Image Processing

Image processing atau pemrosesan citra merupakan proses dengan serangkaian metode yang dilakukan terhadap citra atau lebih tepatnya citra digital dengan tujuan untuk memanipulasi citra atau mengekstraksi informasi yang ada pada citra tersebut [29]. Manipulasi citra dapat meliputi mengurangi *noise*, meningkatkan ketajaman, atau menghilangkan objek yang mengganggu. Sedangkan salah satu contoh

ekstraksi informasi pada citra adalah rekognisi atau proses mengenali objek yang ada pada citra. Pemrosesan citra secara garis besar memiliki tiga tahapan, yaitu tahapan *input* citra, tahapan analisis serta manipulasi citra, dan hasil akhir yang dapat berupa citra setelah manipulasi atau laporan hasil analisis citra.

2.4.1 Preprocessing

Preprocessing merupakan tahap awal dalam pemrosesan citra yang bertujuan menghasilkan citra yang baik untuk dilakukan pemrosesan lebih lanjut [30]. Beberapa metode yang sering digunakan dalam melakukan *preprocessing* pada pemrosesan citra diantaranya adalah *resizing*, *grayscaling*, gaussian blur, *thresholding*, dan *edge detection*.

Meskipun terdapat beberapa metode, tidak semua metode tersebut wajib digunakan. Penggunaan metode tersebut bergantung pada kebutuhan dalam pemrosesan citra. Metode *preprocessing* yang digunakan dalam penelitian ini diantaranya adalah *grayscaling*, gaussian blur, *thresholding*, dan *edge detection*.

2.4.1.1 Grayscaling

Grayscaling merupakan metode untuk mengubah citra RGB menjadi citra *grayscale* dengan tujuan untuk mereduksi waktu komputasi [31]. Citra RGB dikonversi menjadi citra *grayscale* dengan cara mengeliminasi warna pada setiap piksel sehingga meninggalkan hanya tingkat keabuan. Metode ini membantu meminimasi waktu komputasi karena tidak banyak informasi yang harus diproses di setiap piksel yang ada pada citra.

2.4.1.2 Gaussian Blur

Gaussian Blur merupakan metode yang umum digunakan untuk mereduksi *noise* atau derau pada citra [32]. Derau merupakan piksel atau objek yang mengganggu kualitas citra [33]. Penyebab derau diantaranya adalah kualitas

perangkat akuisisi yang kurang baik atau proses akuisisi yang kurang baik, seperti tidak mengatur fokus atau ketidakstabilan saat mengakuisisi citra. Metode Gaussian Blur dapat digunakan untuk menghilangkan derau agar tidak mengganggu pemrosesan citra.

Selain untuk menghilangkan derau, metode Gaussian Blur juga memiliki manfaat lain, yaitu untuk mengurangi tepian pada citra. Pengurangan tepian pada citra dapat membantu dalam proses deteksi objek dengan memperjelas tepian objek pada citra.

2.4.2 Segmentation

Segmentation atau segmentasi merupakan proses membagi citra menjadi bagian kecil [34]. Segmentasi bekerja dengan cara membangun batas atau area pada citra dimana terdapat informasi penting. Area yang terbentuk ini biasa disebut juga *Region of Interest* (RoI). Terbentuknya RoI membantu komputer untuk melakukan analisis hanya terhadap area tertentu pada citra, alhasil waktu kompilasi pun dapat menjadi lebih cepat.

2.4.2.1 Edge Detection

Edge detection atau deteksi tepi merupakan metode segmentasi yang dilakukan dengan cara mencari tepi objek pada citra. Sebuah tepi dapat terbentuk dari hasil perbedaan intensitas cahaya, warna, *shade*, maupun tekstur. Perbedaan intensitas yang jelas antara titik pada citra digunakan pada metode deteksi tepi sebagai acuan untuk mendeteksi tepi [35]. Metode deteksi tepi membantu untuk mendapatkan batasan yang dimiliki setiap objek pada citra. Terbentuknya batasan ini memungkinkan untuk melakukan penyaringan terhadap objek pada citra, sehingga objek yang tidak relevan untuk pemrosesan dapat disisihkan. Penyisihan objek ini juga

membantu untuk meningkatkan kecepatan kompilasi komputer.

Teknik Canny merupakan teknik deteksi tepi yang banyak digunakan saat ini untuk melakukan segmentasi pada citra. Teknik ini dapat menghasilkan citra yang lebih halus, rapat, dan jelas dibandingkan dengan teknik Sobel atau Prewitt. Teknik Canny terdiri dari 5 tahapan dalam melakukan deteksi tepi, yaitu:

1. Pengurangan *noise* menggunakan fungsi Gaussian.
2. Kalkulasi *gradien* untuk mendeteksi intensitas dan arah tepi.
3. *Non-maximum suppression* untuk merapatkan tepi.
4. *Double threshold* untuk menyaring piksel dengan intensitas kuat, lemah, dan piksel yang tidak relevan.
5. *Edge tracking by hysteresis* untuk mentransformasi piksel dengan intensitas lemah menjadi piksel dengan intensitas kuat.

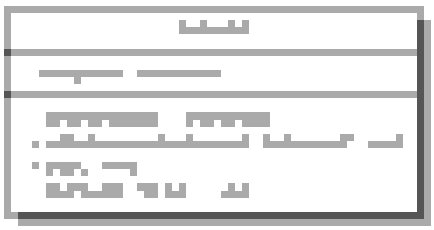
2.4.2.2 Thresholding

Thresholding merupakan metode yang tergolong mudah dalam melakukan segmentasi terhadap citra yang telah melewati tahap *grayscaling*. Metode ini bekerja dengan cara membagi piksel menjadi dua kelas berdasarkan nilai *threshold* yang telah ditentukan [36]. Piksel yang memiliki nilai diatas *threshold* akan diubah nilainya menjadi 1 sedangkan piksel yang memiliki nilai dibawah *threshold* akan diubah menjadi 0. Selain untuk melakukan segmentasi pada citra, *thresholding* juga membantu untuk menghilangkan obstruksi yang dapat mengganggu pemrosesan citra. Terdapat beberapa teknik untuk melakukan *thresholding*, seperti *global thresholding*, *binary thresholding*, dan *threshold to zero* [37]. Teknik *thresholding*

yang digunakan pada penelitian ini adalah *adaptive thresholding*.

2.5 Unified Modeling Language

Unified Modeling Language (UML) merupakan bahasa *modeling* standar untuk pengembangan sistem maupun perangkat lunak [38]. *Modeling* yang dimaksud adalah abstraksi dari sistem yang akan dibangun. Karena itulah UML dapat melibatkan *pseudo-code*, baris kode sebenarnya, citra, diagram, maupun paragraf yang dapat membantu mendeskripsikan sistem yang akan dibangun. Elemen penyusun bahasa *modeling* disebut juga *notation* atau notasi. Contoh notasi pada UML dapat dilihat pada Gambar 2.7.



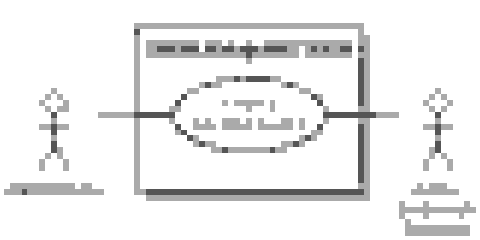
Gambar 2.7 Contoh notasi UML

Sumber: Learning UML 2.0 [38]

Terdapat empat jenis diagram yang umum digunakan dalam UML, yaitu *use case, activity, class,* dan *sequence*. Masing – masing diagram ini memiliki notasi serta kegunaan dalam memodelkan sistem yang akan dibangun.

2.5.1 Use Case Diagram

Use case diagram digunakan untuk mendeskripsikan kebutuhan fungsional sistem. Diagram ini memodelkan fungsionalitas sistem serta lingkungan yang mempengaruhi sistem. *Use case diagram* umumnya disertakan dengan *use case description*, yaitu tabel yang berisi rincian dari sebuah *use case*. Gambar 2.8 merupakan contoh *use case diagram* beserta elemen yang digunakan dalam *use case diagram*.



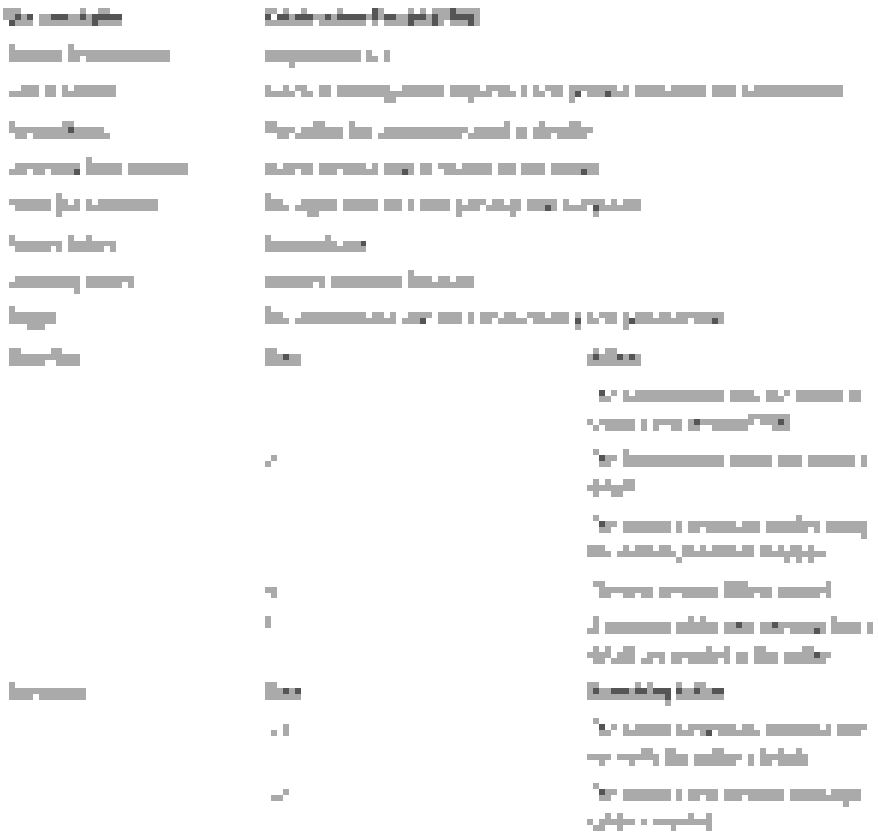
Gambar 2.8 Contoh *Use Case Diagram*

Sumber: Learning UML 2.0 [38]

Terlihat pada Gambar 2.8 *use case diagram* dapat memiliki beberapa elemen sebagai berikut:

1. *Actor*, merupakan lingkungan di luar sistem yang berinteraksi dengan sistem.
2. *Use case*, merupakan fungsionalitas yang ada pada sistem.
3. *Communication lines*, merupakan garis yang menggambarkan hubungan antara *actor* dengan *use case* serta *use case* dengan *use case*. Tipe hubungan antara *use case* dengan *use case* dapat meliputi *include* dan *extend*.

Gambar 2.9 merupakan contoh *use case description* untuk satu *use case* dengan nama "*Create a new Personal Wiki*".



Gambar 2.9 Contoh *Use Case Description*

Sumber: Learning UML 2.0 [38]

Keterangan setiap *description* yang digunakan dalam *use case description* dapat dilihat pada Tabel 2.1.

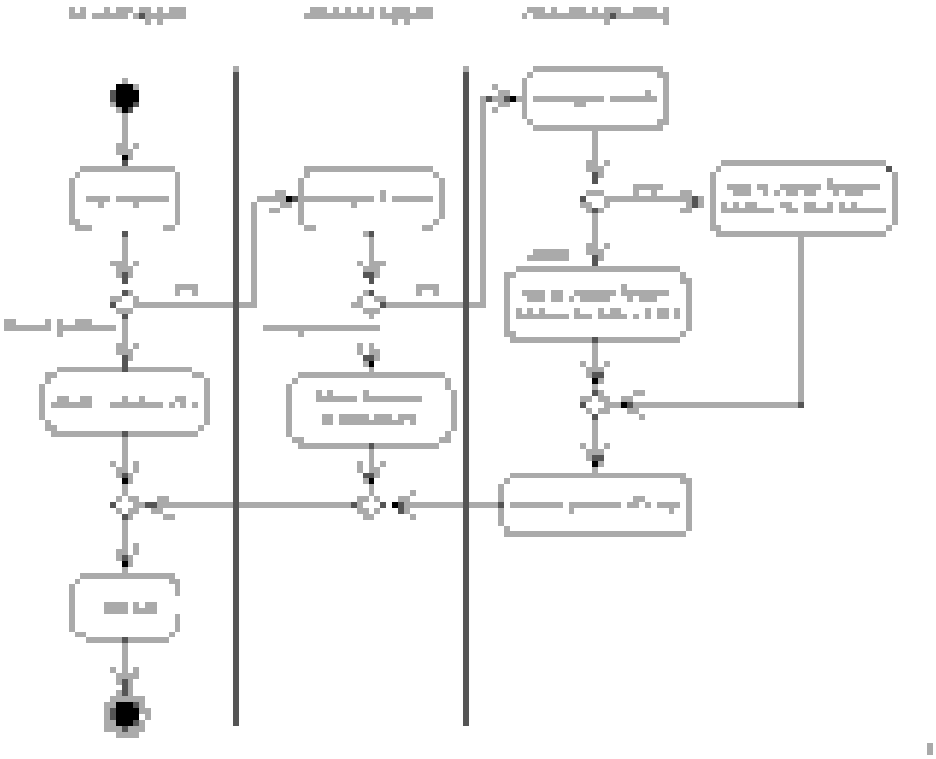
Tabel 2.1 Keterangan Setiap *Description*

No	Nama *description*	Keterangan
1	*Use case name*	Nama dari *use case*.
2	*Related requirements*	Spesifikasi kebutuhan yang dipenuhi oleh *use case*.
3	*Goal in context*	Tujuan *use case*.
4	*Preconditions*	Kondisi yang harus terpenuhi untuk mengakses *use case*.
5	*Successful end condition*	Kondisi apabila *use case* berhasil dieksekusi.

6	*Failed end condition*	Kondisi apabila *use case* gagal dieksekusi.
7	*Primary actors*	Aktor utama yang terlibat dalam *use case*. Umumnya aktor yang memicu serta menerima informasi dari *use case*.
8	*Secondary actors*	Aktor yang terlibat dalam *use case*, namun bukan pemeran utama.
9	*Trigger*	Pemicu yang dilakukan oleh aktor untuk mengeksekusi *use case*.
10	*Main flow*	Deskripsi tahapan dalam eksekusi *use case*.
11	*Extensions*	Deskripsi tahapan alternatif dari tahapan utama pada *main flow.*

2.5.2 Activity Diagram

Activity diagram merupakan diagram yang efektif digunakan untuk mendeskripsikan proses bisnis atau alur aktivitas dalam sistem. Alur pada *activity diagram* merujuk ke tahapan dari satu *use case* pada *use case description*. Diagram ini membantu dalam memvisualisasikan tahapan dari satu *use case* sehingga dapat lebih mudah dimengerti. Gambar 2.10 merupakan contoh *activity diagram* serta elemen yang digunakan dalam *activity diagram*.



Gambar 2.10 Contoh *Activity Diagram*

Sumber: Learning UML 2.0 [38]

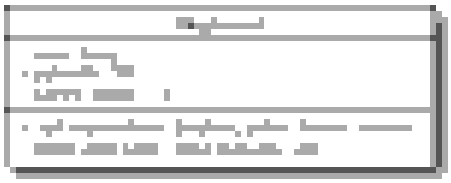
Terlihat pada Gambar 2.10 *activity diagram* dapat memiliki beberapa elemen sebagai berikut:

1. *Initial node*, merupakan titik awal dimulainya tahapan dari satu *use case*.
2. *Action* atau aksi, merupakan elemen yang berisi deskripsi singkat tahapan dari satu *use case*.
3. *Control flow* atau *decision*, merupakan titik percabangan dimana tahapan berikutnya bergantung pada tahapan sebelum percabangan.
4. *Final node*, merupakan titik akhir tahapan dari satu *use case*.

2.5.3 Class Diagram

Class diagram adalah dasar dari sistem dengan desain *object-oriented*. Diagram ini membantu memodelkan struktur dari sistem

dengan menggambarkan kelas beserta metode dan atribut yang dimilikinya. *Class diagram* juga membantu dalam menggambarkan hubungan antar kelas yang ada pada sistem. Gambar 2.11 merupakan contoh *class diagram* serta elemen yang digunakan dalam *class diagram*.



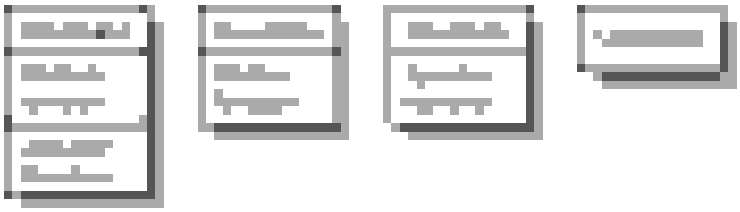
Gambar 2.11 Contoh *Class Diagram*

Sumber: Learning UML 2.0 [38]

Terlihat pada Gambar 2.11 *class diagram* dapat memiliki beberapa elemen sebagai berikut:

1. *Class*, merupakan elemen utama pada *class diagram* yang membungkus atribut beserta metode yang dimilikinya.
2. *Association*, merupakan hubungan yang dimiliki antar kelas pada sistem. Tiga hubungan utama yang umumnya digunakan adalah *one-to-one*, *one-to-many*, dan *many-to-many*.
3. *Attribute* atau atribut, merupakan fitur atau ciri yang dimiliki satu kelas. Elemen ini umumnya dideklarasikan beserta dengan tipe data yang dimilikinya.
4. *Operation* atau *method*, merupakan metode atau fungsi yang dimiliki oleh satu kelas.

Satu *class diagram* tidak harus memperlihatkan atribut atau metode yang dimilikinya. Contoh deklarasi *class diagram* dapat dilihat pada Gambar 2.12.

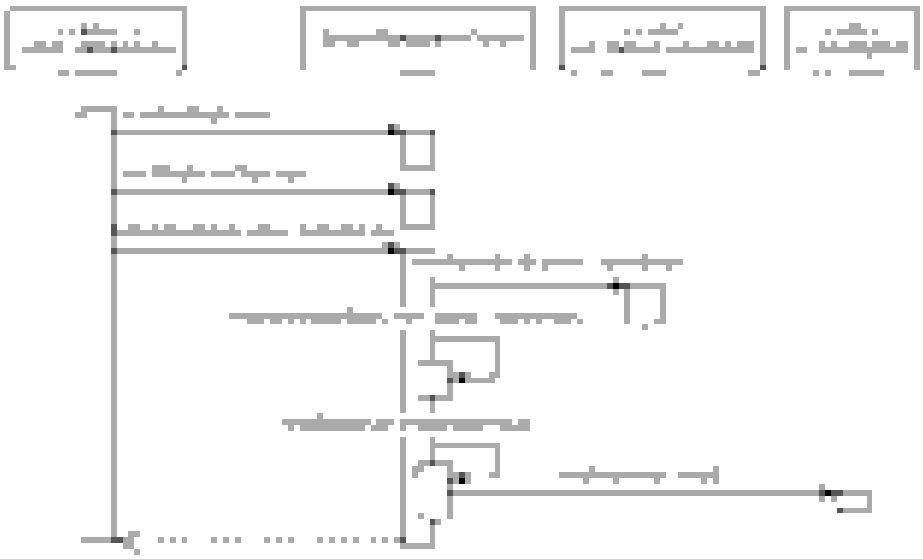


Gambar 2.12 Contoh Deklarasi *Class Diagram*

Sumber: Learning UML 2.0 [38]

2.5.4 Sequence Diagram

Sequence diagram adalah diagram yang menggambarkan interaksi antar objek dalam sistem berdasarkan tahapan *use case* pada *use case description*. Interaksi tersebut tersusun secara terurut sehingga menjadi sebuah skenario yang utuh. Gambar 2.13 merupakan contoh *sequence diagram* serta elemen yang digunakan dalam *sequence diagram*.



Gambar 2.13 Contoh *Sequence Diagram*

Sumber: Learning UML 2.0 [38]

Terlihat pada Gambar 2.13 *sequence diagram* dapat memiliki beberapa elemen sebagai berikut:

1. *Participant*, merupakan objek yang terlibat dalam skenario.

2. *Lifeline,* merupakan indikator apabila satu objek masih terlibat atau tidak dalam skenario.
3. *Activation*, merupakan indikator apabila satu objek melakukan sesuatu disebabkan oleh pesan yang objek tersebut terima.
4. *Message*, merupakan bentuk interaksi antar objek yang terlibat dalam skenario.

2.6 Python

Python merupakan bahasa pemrograman level tinggi dengan semantik yang dinamis, berorientasi objek, serta memiliki interpreter [39]. Python tergolong sebagai bahasa pemrograman *backend* dan ramah untuk pemula karena *syntax* yang mudah dipahami. Python juga memiliki komunitas yang besar serta dokumentasi lengkap yang merupakan faktor pendukung bagi pemula. Bahasa pemrograman ini dibuat oleh Guido van Rossum pada tahun 1990 dan pertama kali dirilis pada tahun 1991 [40]. Penamaan Python sendiri terinspirasi dari acara komedi favorit pembuatnya yang berjudul *Monty Python's Flying Circus*.

Berdasarkan [40] Python mengedepankan konsep *quality*, *productivity*, *portability*, dan *integration* dalam pengembangannya. Saat ini Python dipelihara oleh Python Software Foundation yang merupakan organisasi *non-profit* yang fokus dalam mengembangkan, memperbaiki, serta mempopulerkan bahasa pemrograman Python [41]. Versi terbaru Python saat ini adalah Python 3.11.3 yang dirilis pada 5 April 2023. Beberapa *framework* Python yang populer digunakan adalah Django, Flask, dan Jupyter Notebooks.

Gambar 2.14 Logo Global Python

Sumber: https://cdnlogo.com/logo/python_38146.html

Gambar 2.14 merupakan logo global yang digunakan bahasa pemrograman Python sedangkan Gambar 2.15 merupakan logo khusus yang dirilis untuk Python 3.11.

Gambar 2.15 Logo Khusus Python 3.11

Sumber: https://www.python.org/downloads/release/python-3113/

2.7 Django

Django merupakan *web framework* level tinggi untuk Python yang mendukung pengembangan perangkat lunak cepat dan bersih serta desain pragmatik [42]. Django dapat membantu mempercepat pengembangan perangkat lunak dengan beragam fitur yang disediakan sehingga pengembang tidak perlu membangun hal – hal dasar terlebih dahulu. *Framework* ini dibangun oleh Adrian Holovaty dan Simon Willison pada tahun 2003 dan pertama kali dirilis pada Juli 2005 [43]. Penamaan Django terinspirasi dari gitaris dengan nama Django Reinhardt. Berdasarkan [42] kelebihan utama Django adalah *fast*, *secure*, dan *scalable*.

Saat ini Django dipelihara oleh Django Software Foundation yang didirikan pada tahun 2008. Versi terbaru Django saat ini adalah Django 4.2.1 yang dirilis pada 3 Mei 2023 untuk memperbaiki masalah keamanan dengan tingkat bahaya rendah serta beberapa *bug* lainnya. Gambar 2.16 merupakan logo yang digunakan Django saat ini.

Gambar 2.16 Logo Django

Sumber: https://icons8.com/icons/set/django

2.8 TinyJPG

TinyJPG merupakan *web* yang menyediakan jasa kompresi citra dengan format WebP, PNG, dan JPEG. *Web* ini dapat membantu optimasi citra dengan mengurangi ukuran *file* citra, namun tetap mempertahankan kualitas citra tersebut. TinyJPG melakukan proses *encoding* citra berdasarkan konten yang ada pada citra, karena itulah citra yang berbeda dapat memiliki persentase reduksi ukuran *file* yang berbeda juga. Meskipun begitu, TinyJPG dapat membantu mereduksi ukuran *file* hingga 70% dari ukuran aslinya.

TinyJPG menyediakan API untuk dapat digunakan oleh pengembang dengan beragam fitur. Fitur yang tersedia di API TinyJPG saat ini diantaranya adalah kompresi citra, konversi format citra, *resizing*, dan pendeteksi *area of interest*. TinyJPG saat ini sudah dipercaya oleh berbagai perusahaan besar, seperti Samsung, Amazon, dan Sony. Logo yang digunakan TinyJPG saat ini dapat dilihat pada Gambar 2.17.

Gambar 2.17 Logo TinyJPG

Sumber: https://twitter.com/tinyjpg/photo

Tahapan dalam melakukan konfigurasi TinyJPG menggunakan bahasa pemrograman Python adalah sebagai berikut:

1. Konfigurasi API key. Konfigurasi API key dapat dilakukan di alamat berikut:

   ```
   https://tinyjpg.com/developers
   ```

   Setelah memasukkan nama lengkap serta alamat *email*. TinyJPG akan mengirimkan alamat dashboard kepada *email* tersebut. API key untuk melakukan *request* sudah tersedia di dashboard tersebut.
2. Instalasi tinify. Tinify merupakan *package* yang dapat membantu proses komunikasi dengan API TinyJPG. Sebelum melakukan *request* kepada API, perlu dilakukan konfigurasi API key pada baris kode menggunakan sintaks berikut:

   ```
   tinify.key = “API_KEY”
   ```

2.9 OpenCV

Open Source Computer Vision Library (OpenCV) merupakan *open-source library* dengan fungsi utama untuk membantu analisis citra dan video [44]. OpenCV pertama kali dirilis pada tahun 1999 oleh Intel dengan tujuan untuk mendorong riset di bidang CV serta menyebarluaskan ilmu CV. Maka dari itu Intel membuat *library* ini dapat digunakan oleh siapapun tanpa dipungut biaya serta siapapun dapat berkontribusi dalam pengembangan OpenCV.

OpenCV saat ini memiliki lebih dari 2500 algoritma yang meliputi algoritma CV serta *machine learning* dan dapat digunakan untuk bahasa pemrograman C++, Python, Java, serta perangkat lunak MATLAB. *Library* ini mendukung sistem operasi Windows, Linux, Android, dan macOS serta telah dipercaya oleh perusahaan besar, seperti Google, Microsoft, Intel, Sony, dan Honda. Versi terbaru OpenCV saat ini adalah OpenCV 4.7.0 yang dirilis pada 29 Desember 2022. Logo yang digunakan OpenCV saat ini dapat dilihat pada Gambar 2.18.

Gambar 2.18 Logo OpenCV

Sumber: https://icons8.com/icons/set/opencv

2.10 Google OAuth

Google *Open Authorization* (OAuth) merupakan protokol otorisasi milik Google yang memungkinkan sistem untuk mendapatkan akses informasi akun pengguna. Google OAuth dapat membantu sistem untuk mengidentifikasi pengguna yang melakukan otentikasi dalam sistem. Metode ini terbilang lebih aman dibandingkan dengan membuat sistem otentikasi karena pengguna tidak perlu memberikan informasi sensitif, seperti *password* kepada sistem.

Berikut adalah rincian alur implementasi OAuth menggunakan *framework* Django:

1. Instalasi allauth. Allauth merupakan merupakan *package* yang dapat membantu proses autentikasi menggunakan pihak ketiga seperti OAuth.
2. Konfigurasi Google API. Konfigurasi ini dapat dilakukan di alamat berikut:

https://console.cloud.google.com

   Tahapan konfigurasi yang dilakukan diantaranya adalah membuat *project* baru, membuat kredensial dengan tipe OAuth Client ID, dan mendapatkan Client Secret serta Client ID.
3. Membuat social app. Pembuatan social app dapat dilakukan di alamat berikut:

http://127.0.0.1:8000/admin

Pada tahapan ini dilakukan registrasi autentikasi pihak ketiga dengan mendaftarkan Client Secret dan Client ID yang sudah didapatkan di tahap kedua.