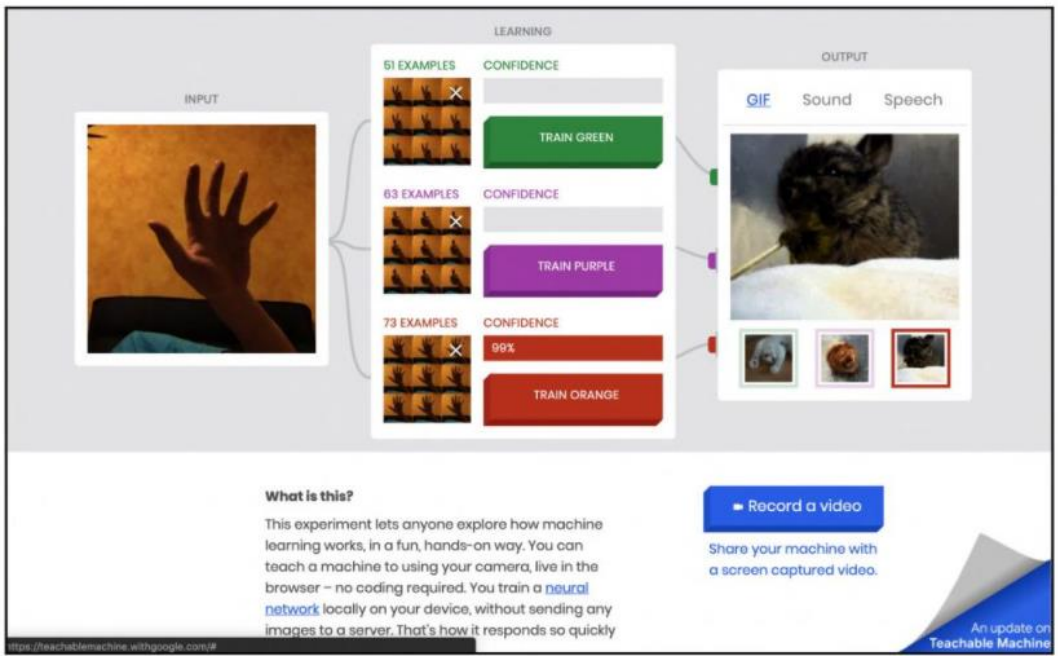
BAB II
LANDASAN TEORI

2.1 *Teachable Machine*

Sasaki (2019) menguraikan bahwa mesin yang dapat diajarkan adalah contoh yang baik dari kekuatan mengintegrasikan dengan *web*. menggunakan input dari kamera *web* dengan menggunakan *api browser*. Gambar 2.1 berikut ini adalah screenshot dari mesin yang dapat diajar.(http://teachablemachine.withgoogle.com):



Gambar 2.1 *Teachable Machine.*

Teachable machine akan membuat sebuah model cerdas berdasarkan kebutuhan. Pada penelitian ini, peneliti menggunakan *teachable machine* untuk membuat model berdasarkan input gambar wajah mahasiswa. *Output teachable machine* menghasilkan model yang telah dilatih untuk mengenali gambar wajah mahasiswa.

Tahapan penggunaan *Teachable Machine*:

a. *Gather*

Kumpulkan dan kelompokkan contoh Anda ke dalam kelas, atau kategori, yang Anda ingin komputer pelajari.

b. *Train*

Latih model Anda, lalu langsung uji untuk melihat apakah itu dapat mengklasifikasikan contoh baru dengan benar.

c. *Export*

Ekspor model Anda untuk proyek Anda: situs, aplikasi, dan banyak lagi. Anda dapat mengunduh model Anda atau meng-host secara online secara gratis.

Gambar 2.2 Proses Deteksi Wajah Oleh Linda Rahmayanti, Yesi Diah Rosita, Dinarta Hanum Yesy et al. (2019).

2.2 *Tensorflow*

Laborde (2021) menguraikan bahwa *Tensorflow* adalah salah satu kerangka pembelajaran mesin paling populer di pasar. ini didukung oleh pikiran top *Google* dan bertanggung jawab untuk memberi daya pada banyak perusahaan paling berpengaruh di dunia.

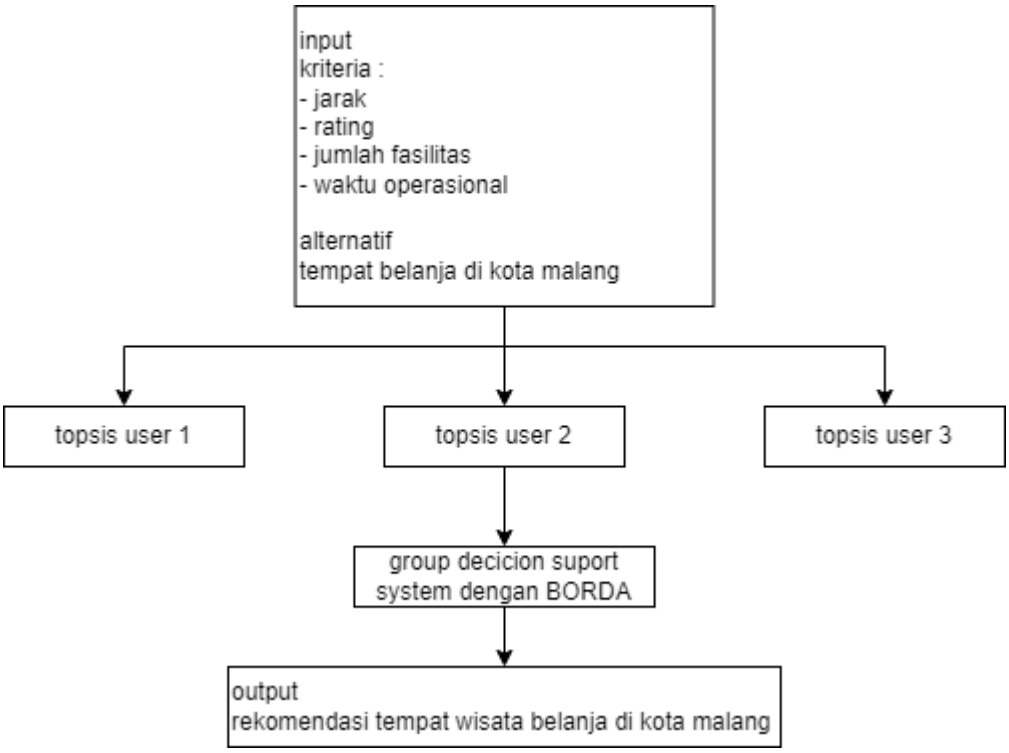
Pada penelitian ini sebuah model tensorflow akan menjadi model cerdas yang dapat mengenali wajah. Sebuah model cerdas akan digunakan untuk mengenali wajah seorang mahasiswa.

2.3 *Base Location*

Forrester (2021) menguraikan bahwa tetelah berhasil mendapatkan izin dari pengguna mengakses lokasi mereka, kami sekarang dapat meminta perangkat pengguna untuk memberi kami lokasi terakhir yang diketahui, yang juga biasanya akan menjadi lokasi pengguna saat ini. Lokasi tersebut akan menjadi parameter dalam melakukan absensi.

Pada penelitian *ini Base location* berperan sebagai parameter ketika dilakukan absensi oleh mahasiswa. *Base location* akan mendeteksi posisi atau lokasi perangkat android yang terhubung dengan jaringan internet. Penerapan *base location* pada penelitian ini memiliki beberapa aturan yaitu sebagai berikut :

a. Jika posisi latitude dan longtitude perangkat ketika absen sama dengan posisi latitude dan longtitude tyang didefinisikan oleh seorang dosen maka absensi dapat dilakukan.
b. Jika posisi latitude dan longtitude sebuah kelas belum didefinisikan maka absensi tidak dapat dilakukan
c. Jika posisi latitude dan longtitude tidak sesuai dengan posisi latitude dan longtitude yang didefinisikan dosen maka status lokasi akan bernilai false dan tidak dapat melakukan absensi
d. Sorang dosen wajib mendefinisikan posisi latitude dan longtitude sebuah kelas.



Gambar 2.3 Perancangan Algoritama *Location Base Service* Oleh Felinda Gracia Lubis, Ratih Kartika Dewi, Komang Candra Brata.Ramadhan et al. (2020)

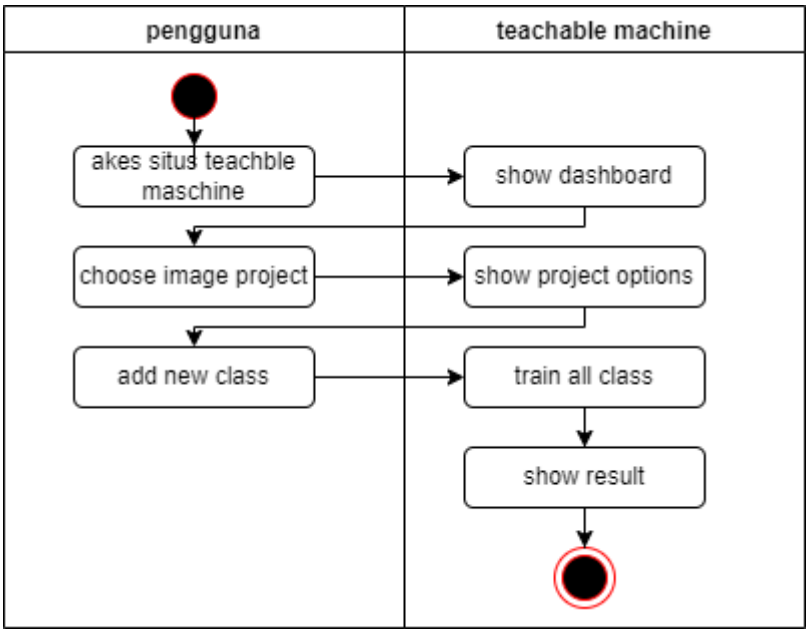
2.4 Validasi Wajah

Riyanto (2019) menguraikan bahwa Validasi adalah konfirmasi melalui pemeriksaan dan penyediaan bukti objektif bahwa persyaratan tertentu yang dimaksudkan tertentu terpenuhi.

Muttaqin (2020) menguraikan bahwa Wajah merupakan suatu hal alami yang didapatkan oleh manusia. Dalam kehidupan sehari hari manusia mampu mendeteksi setiap orang dengan validasi wajah.

Pada penelitian ini validasi wajah mahasiswa dilakukan oleh aplikasi ketika seorang mahasiswa selesai melakukan input gambar wajahnya. Wajah akan divalidasi oleh aplikasi dan ditampilkan prediksi nama berdasarkan input gambar wajah dan ditampilkan presentase kecocokan input wajah dengan model yang terdapat dalam aplikasi. Model wajah mahasiswa dibuat dengan bantuan *teachable machine* dan model tersebut akan di load oleh aplikasi ketika input

gambar wajah mahasiswa dilakukan. Dalam pembuatan model peneliti melakukan tahapan demi tahapan sebagai berikut.



Gambar 2.4 Gambar *Activity Diagram* Validasi Wajah

a. Mengakses teachable machine melalui wesite resmi pada techablemachine https://teachablemachine.withgoogle.com
b. Create image project
c. Chose standart image model
d. Add new class
e. Train model
f. Download model
g. Import model to android project

Dengan melalui tahapan demi tahapan peneliti memperoleh model yang telah dilatih dengan bantuan teachable machine dan siap digunakan pada penelitian ini.

2.5 *Android Studio*

Alexander, Onki. Suprihadi (2021) menguraikan bahwa Android Studio adalah *Integrated Development Environment* (IDE) untuk sistem operasi *Android*, yang dibangun diatas perangkat lunak *Jetbrains Intellij IDEA* dan di desain khusus

untuk pengembangan android. *Tools* ini merupakan pengganti dari Eclipse *Android Development Tools*(ADT) yang sebelumnya merupakan *Tools* utama dalam pembuatan aplikasi android.

Anadroid studio pada versinya memiliki fitur-fitur yang berguna yaitu :

a. Dukungan gradle-base built
b. *Android-sppesific* refactoring dan perbaikan cepat
c. Lint tools untuk menangkap kinerja keggunaan, kompabilitas versi, dan masalah lainnya.
d. Integrasi proguard dan kemampuan penenda tanganan aplikasi
e. Template base *wizards* untuk membuat *template* desain umum seperti drawer atau *empty activity*
f. Mendukung pengembangan aplikasi *android wear*
g. Editor tata letak yang memungkinkan pengguna untuk menyeret dan menjatuhkan (*drag-and-drop*) komponen *UI*, opsi untuk melihat tata letak beberapa konfigurasi layar
h. Dukungan bawaan untuk *google cloud platform*, memungkinkan intergrasi dengan *firbase cloud messanging*(perpesanan *google cloud'* sebelumnya) dan *google app engine*
i. *Android virtual device*(emulator) untuk menjalankan dan men-debug aplikasi di android studio.

Android Studio menawarkan banyak fitur yang meningkatkan produktifitas anda dalam membuat apliaksi *android*, seperti:

a. Sistem build berbasis *gradle* yang fleksibel
b. Emulator yang cepat dan kaya fitur
c. Lingkungan terpadu tempat anda bias mengembangkan aplikasi untuk semua perangkat android
d. Terapkan perubahan untuk melakukan push pada perubahan kode dan *resource* ke aplikasi yang sedang berjalan tanpa memulai ulang aplikasi

e. Template kode dan integrasi github untuk membantu anda membuat fitur aplikasi umum dan mengimpor kode sampel.
f. *Framework* dan pengujian yang lengkap
g. Alat lint untuk merekam peforma, kegunaan, kompabilitas versi, dan masalah lainya
h. Dukungan C++ dan NDK
i. Dukungan bawaan untuk google cloud platform, yang memudahkan integrasi *google messanging* dan *app engine.*

2.6 Java

Kadir (2020) menguraikan bahwa Java merupakan bahasa pemrograman berorientasi objek yang diciptakan oleh James Gosling dan kawan-kawanya di Sun Microsystem. Bahasa ini mengadopsi perintah perintah pada C++.

Saat ini, java dimiliki oleh perusahaan Oracle. Perusahaan ini menyediakan dua jenis piranti pengembangan java(JDK – java Development Kit), yaitu Oracle JDK dan OpenJDK. JDK pertama bersifat komersial yang kedua bersifat "*open source*" yang disediakan oleh perusahaan lain.

Berbeda denngan C++, java menerapkan konsep "*Write Once, Run Anyware*" . kode java yang teelah dikompilasi dalam bentuk *bytecode* dapat dijalankan pada berbagai platform. Kelebihan memungkinkan program tidak perlu menyediakan perintah yang secara eksplisit digunakan untuk membebaskan memori suatu object yang sudah tidak digunakan lagi. Semua objek yang tidak digunakan dengan sendirinya akan dibebaskan.

2.7 Mysql

Fitri (2020) menguraikan bahwa MySql adalah DBMS yang *open source* dengan dua bentuk lisensi, yaitu *free software* (perangkat lunak bebas) dan *shareware* (perangkat lunak berpemilik yang penggunanya terbatas). Jadi MySql adalah *database server* yang *open source* dengan lisensi GNU *General Public License*

(GPL) sehingga dapat dipakai untuk keperluan pribadi atau komersial tanpa harus membayar lisensi yang ada.

Seperti yang sudah disebutkan sebelumnya, MySql masuk kedalam jenis RDBMS(*Relational Database Manajement Sistem*). Maka dari itu, istilah semacam baris, kolom, table, dipakai pada MySql. MySql merupakan *database* yang mendukung dalam mengelola data pada aplikasi absensi mahasiswa dengan validasi wajah dan base location.

2.8 Adobe Photoshop

Jubilee (2019) menguraikan bahwa *Adobe Photoshop* merupakan *software* "sejuta umat". *Software* ini unik sebab digunakan oleh banyak jenis profesi. Beberapa profesi tersebut yaitu fotografer, editor foto, desainer grafis, desainer website, layouter majalah, penulis buku, dan pekerja kreatif lainya.

Oleh karena keunikan tersebut, penulis memanfaatkan adobe photshop untuk merancang berbagai desain. Pada penelitian ini aplikasi *Adobe Photoshop* digunakan untuk membuat rancangan *User Interface* aplikasi absensi mahasiswa menggunakan validasi wajah dab *base location*.

2.9 Draw Io

Jgraph (2022) menguraikan bahwa draw.io pro adalah aplikasi diagram aktif *Google Drive* (TM) gratis yang memungkinkan anda untuk menggambar. *Tools* ini sangat diakses dan dikelola dengan fitur yang ditawarkan tools ini yaitu :

a. HTML 5 *Native* dengan dukungan penuh untuk IE 6-8
b. Kaya pustaka sensil bawaan
c. Antarmuka seret dan lepas yang intuisif
d. Fungsi tambah dan cari gambar
e. Ekspor ke PNG/JPG/XML/SVG
f. Dukungan perangkat sentuh
g. Kolaborasi *realtime*

Dengan fitur-fitur yang ditawarkan tools ini, peneliti menggunakan Draw Io untuk membuat rancangan alur kerja (*workflow*) aplikasi absensi mahasiswa menggunakan validasi wajah dan base location.

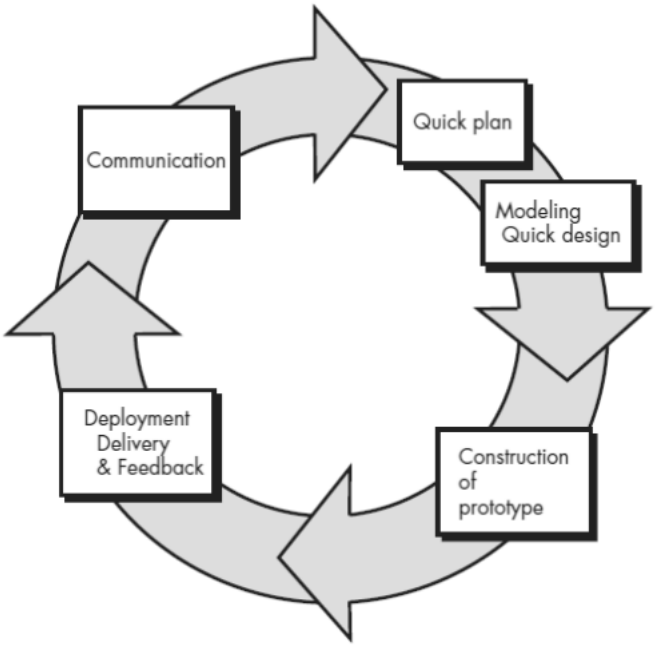
2.10 Metode *Prototype*

Pressman (2017) menguraikan bahwa dalam melakukan perancangan sstem yang akan dikembangkan dapat menggunakan metde *prototype*. Metode ini cocok digunakan untuk mengembangkan sebuah perangkat lunak yang dikembangkan kembali. Metode ini dimulai dengan pengumpulan kebutuhan pengguna. Kemudian membuat sebuah rancangan kilat yang selanjutnya akan dievaluasi kembali sebelum di produksi secara benar.

Prototype bukanlah merupakan sesuatu yang lengkap, tetapi sesuatu yang harus dievaluasi dan dimodifikasi kembali. Segala perubahan dapat terjadi pada saat *prototype* dibuat untuk memenuhi kebutuhan pengguna dan saat yang sama memungkinkan pengembangan untuk lebih memahami kebutuhan pengguna secara baik.

Berikut adalah tahapan dalam metode *protoype*:

a. *Comunication*, yaitu analisis terhadap kebutuhan pengguna.
b. *Quick plan,* yaitu melakukan analisa dan perancanaan setelah mendapatkan data-data dari tahapan komunikasi.
c. *Quick design* (desain cepat), yaitu pembuatan desain secara umum untuk selanjutnya dikembangkan kembali.
d. *Construction of prototype*, yaitu memproduksi perangkat *prototype* termasuk pengujian dan penyempurnaan.
e. *Deployment delivery & feedback*, yaitu merupakan tahapan ketika aplikasi telah selesai dibuat.



Gambar 2.5 Metode *Prototype* R. Pressman (2017)

2.11 *Unified Modeling Language*(UML)

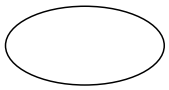
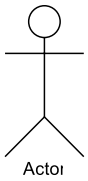
Salahudin (2019) menguraikan bahwa UML (*Unified Modeling Language*) adalah standard bahasa yang digunakan di dunia industry untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambar arsitektur dalam program berorientasi objek. Karena UML hanya berfungsi untuk melakukan pemodelan, penggunaan UML tidak terbatas pada metodologi tertentu, tetapi dalam praktiknya UML terutama digunakan untuk metodologi berorientasi objek. Pengembangan dengan UML tergantung pada tingkat abstraksi yang Anda gunakan, dan dengan UML Anda bisa salah, tetapi Anda perlu mempelajari di mana harus menggunakan UML dan apa yang ingin Anda visualisasikan. UML terdiri dari 13 jenis diagram, dibagi menjadi 3 kategori sebagai berikut.

2.9.1 *Use Case Diagram*

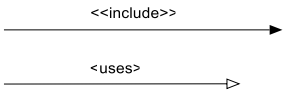
Salahudin (2019) menguraikan bahwa *Use Case Diagram* atau Case Diagram merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. Pada dasarnya, use case memberi tahu Anda apa fungsi dalam sistem informasi dan siapa yang memiliki izin untuk menggunakan fungsi tersebut.

Symbol dan deskripsi *use case diagram* adalah seperti yang ditunjukan pada tabel 2.1.

Tabel 2.1 Simbol *Use Case Diagram*

No	Symbol	Deskripsi
1	*Use case* 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau *actor*; biasanya dinyatakan dengan menggunakan kata kerja diawal di frase nama *use case*
2	Aktor/*actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari actor adalah gambar orang, tapi *actor* belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
3	Asosiasi/*association*	Komunikasi antara *actor* dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan *actor*.
4	Ekstensi / *extended*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi *object*.

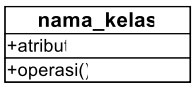
Tabel 2.2 Simbol *Use Case Diagram*

5	Geberalisasi / *generalization*	Hubungan generalisasi dan spesifikasi (umum - khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang kebuh umum dari lainya.
6	Menggunakan / *include / uses* 	Realisasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini.

2.9.2 Class Diagram

Salahudin (2019) menguraikan bahwa Diagram Kelas atau *Class Diagram* mengambarkan struktur sistem dari segi definisi kelas-kelas yang akan dibuat untuk membangun sistem. Simbol–simbol yang ada pada diagram kelas ditunjuakan pada tabel 2.2 berikut.

Tabel 2.3 Simbol *Class Diagram*

No	Symbol	Keterangan
1	kelas 	Kelas pada stuktur sistem
2	Antarmuka / *interface* 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
3	Asosiasi / *association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga *multiplicity*
4	Asosiasi berarah / *directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai *multiplicity*

Tabel 2.4 Simbol *Class Diagram*

5	Generalisasi	Relasi antar kelas dengan makna generalisasi-spesialisasi(umum-khusus)
6	Kebergantungan / *dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas
7	Agregasi / *aggregation*	Relasi antar kelas dengan makna semua-bagian(*whole-part*)

2.9.3 Activity Diagram

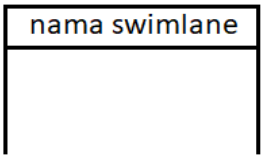
Salahudin (2019) menguraikan bahwa Diagram Aktifitas atau *Activity Diagram* menggambarkan *workflow*(aliran kerja) atau aktifitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak.

Diagram Aktivitas membantu memvisualisasikan alur kerja dari satu aktivitas ke aktivitas lainnya. Penekanannya adalah pada kondisi aliran dan urutan terjadinya. Aliran dapat berurutan, bercabang, atau simultan dan menangani jenis aliran ini. Berikut adalah simbo;-simbol yang terdapat pada diagram aktifitas:

Tabel 2.5 Simbol *Activity Diagram*

No	Symbol	Deskripsi
1	Satus awal 	Status awal aktivitas sistem, sebuah diagram aktivitas, memiliki sebuah status awal.
2	Aktivitas 	Akktivitas yang dilakukan sitem, aktivitas biasanya diawali dengan kata kerja
3	Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.

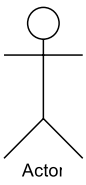
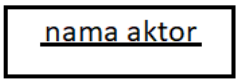
Tabel 2.6 Simbol *Activity Diagram*

4	Penggabungan / *join*	Asosiasi penggabungan dimana libih dari satu aktivitas digabungkan menjadi satu.
5	Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram memiliki status akhir.
6	Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

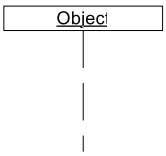
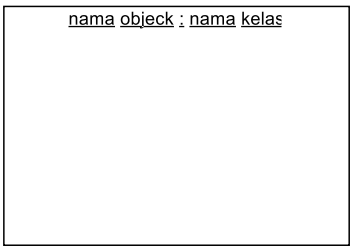
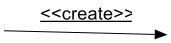
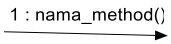
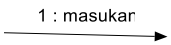
2.9.4 *Sequence Diagram*

Salahudin (2019) menguraikan bahwa Diagram Sikuen menggambarkan kelakuan *object* pada *Use Case* dengan mendeskripsikan waktu hidup objek dari *message* yang dikirimkan dan diterima antar objek. Berikut adalah symbol symbol yang ada pada diagram sekuen:

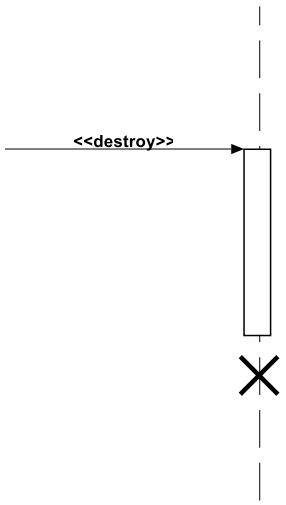
Tabel 2.7 Simbol *Sequence Diagram*

No	Symbol	deskripsi
1	*Actor*  Atau  Tanpa waktu aktif	Orang, proses, atau sistem lain yang berorientasi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari *actor* adalah gambar orang, orang; biasanya dinyatakan kata benda di awal frase nama *actor*.

Tabel 2.8 Simbol *Sequence Diagram*

2	Garis hidup / *lifeline* 	Menyatakan kehidupan suatu objek
3	Objek 	Menyatakan objek yang berinteraksi pesan
4	Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya
5	Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
6	Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi / metode yang ada padaa objek lain atau dirinya sendiri.
7	Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainya, arah panah mengarah pada objek yang dikirim
8	Pesan tipe *return* 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.

Tabel 2.9 Simbol *Sequence Diagram*

9	Pesan tipe destroy 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada obejk yang diahkiri, sebaliknya jika ada create maka ada destroy

2.12 Pengujian Kotak Hitam (*Black-Box Testing*)

Pressman (2012) menguraikan bahwa *Black Box Testing* atau Pengujian Kotak Hitam atau juga disebut *Behavioral Testing*, berfokus pada persyaratan fungsional dari perangkat lunak. Artinya, teknik *Black-Box Testing* memungkinkan untuk mendapatkan set kondisi masukan yang sepenuhnya akan melaksanakan semua persyaratan fungsional untuk suatu program.

Black-Box Testing bukan merupakan alternatif dari pengujian *White Box Testing*. Sebaliknya, *Black-Box Testing* adalah pendekatan komplementer yang mungkin untuk mengungkap kelas yang berbeda dari kesalahan daripada metode *White Box Testing. Black Box Testing* untuk menemukan kesalahan dalam kategori berikut:

a. Fungsi tidak benar atau hilang.
b. Kesalahan interface atau antarmuka.
c. Kesalahan dalam struktur data atau akses database eksternal.
d. Kesalahan kinerja atau perilaku.
e. Kesalahan inisialisasi dan terminasi.

2.13 Penelitian Terkait

Pada beberapa penelitian sebelumnya sudah dilakukan pengujian dengan metode dan hasil yang didapat. Guna mencapai hasil yang maksimal dalam penelitian yang dilakukan maka perlu dilakukan analisa terkait dengan penelitian sebelumnya dari berbagai sumber. Berikut merupakan penelitian terkait menegenai absensi dan *base location*. Dari penelitian terkait diperoleh pengetahuan untuk membuat keterbaruan penelitian saat ini.

Keterbaharuan pada penelitian yang peneliti teliti adalah penggunaan model *tensorflow* sebagai *image* model untuk digunakan sebagai model cerdas yang dapat mengenali objek gambar, sehingga pada kasus validasi dasi wajah mahasiswa dapat dikenali. Pada penelitian ini disertai *base location* untuk mengetahui posisi perangkat yang sedang digunakan.

Tabel 2.10 Penelitian Terkait

No	Nama	Judul	Terbit/ Tahun	Keterangan
1	Evta Indra, M Diarmansyah Batubara, Muhammad Yasir dan Sugandi Chau	Desain dan Implementasi Sistem Absensi Mahasiswa Berdasarkan Fitur Pengenalan Wajah dengan Menggunakan Metode Haar-Like FeatureIndra et al. (2019)	Evta Indra, M Diarmansyah Batubara, Muhammad Yasir dan Sugandi Chau/2019	Dalam penelitian tersebut diterapkan suatu stem pengenalan wajah yang dirancang menggunakan metode Haar-Like Feature yang memberikan indikasi secara spesifik pada sebuah gambar atau image yang digunakan untuk mengenali objek berdasarkan nilai sederhana dari fitur.

Tabel 2.5 Penelitian Terkait

2	Rendy Rian Chrisna Putra dan Fransiskus Panca Juniawan	Penerapan Algoritma Fisherfaces Untuk Pengenalan Wajah Pada Sistem Kehadiran Mahasiswa Berbasis Android Schlimm (2018)	Jurnal Telematika Vol. 10 No. 1 Februari 2017	Pada penelitian ini algoritma yang digunakan yaitu local binary pattern (LBP) cascade untuk mendeteksi wajah dan algoritma fisherfaces untuk pengenalan wajah
3	Naviza Qois, Yuwan Jumaryadi	Implementasi Location Based Service pada Sistem Informasi Kehadiran Pegawai Berbasis Android Qois & Jumaryadi (2021)	Jurnal Sistem Informasi/2021	Pada penelitian dibuat untuk mendapatkan laporan absensi secara realtime dan disertai dengan koordinaat pegawai saat absensi, validasi yang digunakan dengan melakukan foto selfie wajah pegawai dan validasi atasan terhadap daily timesheet yang diberikan melalui aplikasi. Pengembangan aplikasi ini menggunakan metodologi waterfall

Tabel 2.5 Penelitian Terkait

4	Angelia, Della	Rancang Bangun Perangkat Lunak Presensi Mahasiswa Dan Dosen Secara Realtime Berbasis Web MobileAngelia (2020)	Repository Institute Informatika Dan Bisnis Darmajaya/2020	Penelitian ini bertujuan untuk membangun perancangan perangkat lunak mahasiswa dan dosen secara real-time berbasis web mobile. Dalam kegiatan presensi menggunakan QR Code sebagai teknologi yang dikembangkan.
5	Linda Rahmayanti, Yesy Diah Rosita, Dinarta Hanum	Sistem Absensi Mahasiswa Berdasarkan Citra Wajah Menggunakan Metode Principal Component Analysis (Pca) Yesy et al. (2019)	Universitas Islam Majapahit	Proses pengenalan wajah menggunakan pengenalan wajah berdasarkan ekstrasi ciri citra dengan metode Principal Component Analysis (PCA). Dimana pada pada penelitian ini hasil nilai tingkat akurasi pengenalan wajah dengan nilai tingkat akurasi sebesar 85%.