BAB II
LANDASAN TEORI

2.1 Tinjauan Literatur

Tinjauan studi digunakan sebagai bahan pertimbangan yang memiliki kaitan dengan penelitian yang dilakukan saat ini. Terdapat beberapa penelitian terdahulu yang menjadi refrensi untuk penelitian ini, antara lain :

Tabel 2.1 Penelitian Terdahulu

No	Judul	Tahun	Deskripsi
1	Rancang Bangun Aplikasi Pencarian Lokasi dan Informasi ATM dengan Menggunakan *Location Based Service* Berbasis *Android* di Provinsi Sulawesi Selatan	2018	Pada penelitian ini Asmaul Husna melakukan penelitian mengenai sistem informasi ATM *(Automated Teller Machine),* menggunakan *Location Based Service* Berbasis *Android* pengembangan sistem perangkat lunak yang digunakan adalah metode *Waterfall* model. Dan teknik pengujian sistem menggunakan *Black box Testing dan White box Testing* hasil dari penelitian ini adalah Aplikasi dapat menampilkan peta ATM, jenis ATM, nominal pecahan uang di setiap ATM dan jumlah mesin ATM.
2	Implementasi *Location Based Service* Berbasis *Android* Untuk Pencarian Akomodasi Pariwisata di Pesisir Barat	2018	Pada penelitian yang dilakukan oleh Bayu Pandu Putra Pratama adalah mencari akomodasi pariwisata di pesisir barat menggunakan *Location Based Service,* dan menggunakan *Prototype* model sebagai pengembangan sistemnya. Tujuan dibuatnya aplikasi ini adalah untuk memudahkan wisatawan dari luar kota maupun di wilayah pesisir barat dalam mencari

No	Judul	Tahun	Deskripsi
			pusat oleh-oleh, tempat penginapan, puskesmas dan kantor polisi terdekat.
3	Implementasi *Location Based Service* Untuk Pencarian Sanggar Bunga di Bandar Lampung Berbasis *Android*	2019	Pada penelitian yang dilakukan M.Imroni Muslikhin melakaukan penelitian tentang mencari sanggar bunga di bandar lampung, dengan menggunakan *Location Based Service* berbasis *Android* dengan *Java* sebagai Bahasa pemrogramannya. Tujuan dilakukannya penelitian ini adalah untuk mengtahui jarak latak sanggar bunga di Bandar lampung dan rute nya
4	*Location Based Service* Untuk Pencarian Lokasi Usaha Lokal Menggunakan Ionic Framework	2019	Berdasarkan penelitian yang dilakukan oleh Kasmawi, yamg berjudul *Location based service* untuk pencarian lokasi usaha lokal dikota bengkalis menggunakan *Ionic Framwork*. Penelitian ini bertujuan untuk memberikan solusi bagi pelaku usaha untuk memperkenalkan usaha dan produk kepada pengguna, aplikasi ini dirancang menggunakan pendekatan metode *Prototype* dengan Bahasa pemrograman *Java*, PHP dan MySQL sebagai database. Aplikasi ini dapat membantu pengguna dalam proses pencarian lokasi usaha dalam bentuk informasi peta, mengetahu jarak tempuh dan dapat melakukan transaksi jual beli.

No	Judul	Tahun	Deskripsi
5	Sistem Informasi Location Based Service Sentra Keripik Kota Bandar Lampung Berbasis *Android*	2020	Pada penelitian yang dilakukan oleh Bayu Pratama dan Adhie Thyo Priandika yang berjudul sistem informasi *Location Based Service* sentra kripik berbasis *Android*, dengan adanya penelitian ini diharapkan dapat memudahkan wisatawan dari dalam maupun luar bandar lampung untuk mencari sentra kripik dengan jarak terdekat. Pada penelitian ini menggunakan Algoritma A* dan pengujian menggunakan *Black box* testing. Dalam pengujian menggunakan *Black box* telah diketahui bahwa sistem dapat melakukan fungsinya dengan baik.

2.2 Tinjauan Pustaka

2.2.1 Pengertian LBS (*Location Based Service*)

Location Based Service (LBS) atau layanan berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan *mobile network*, yang dilengkapi kemampuan untuk memanfaatkan lokasi dan *mobile device* tersebut. LBS memberikan kemungkinan komunikasi dan interaksi dua arah. Selain itu LBS digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat atau suatu objek tertentu. (Nazruddin Safaat,2015) dalam (Ratnasari et al., 2018). Untuk menampilkan peta suatu lokasi diperlukan dua unsur utama dari *Location Based Service* yaitu :

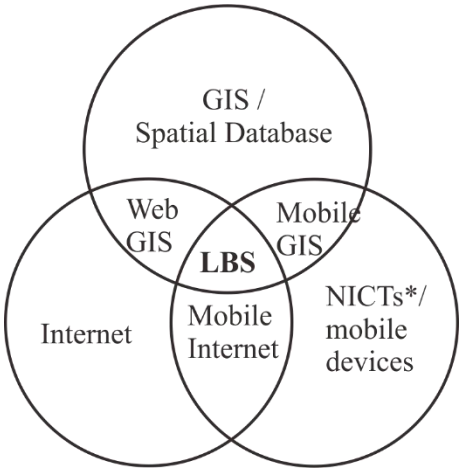
a) *Location Manager* (*API Maps*)

Menyediakan *tools/source* untuk LBS, *Application Programming Interface* (*API*) *Maps* menyediakan fasilitas untuk menampilkan, memanipulasi

maps/peta beserta fitur-fitur lainnya seperti tampilan satelit, *street* (jalan), maupun gabungannya. Paket ini berada pada *com.google.Android.maps*.

b) *Location Provider* (*API Location*)

Menyediakkan teknologi pencarian lokasi yang digunakan oleh *device*/perangkat. *API Location* berhubungan dengan data GPS (*Global Positioning System*) dan data lokasi *real-time*. *API Location* berada pada paket *Android* yaitu dalam paket *Android location*. Dengan *Location Manager*, kita dapat menentukan lokasi kita saat ini, *track* gerakan/perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan.



Gambar 2.1 Teknologi *Location Based Service* (safaat, 2015)

Location Based Service dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu *: Geographic information System*, *internet service* dan *Mobile Devices.* Teknologi *Location Based Service* berfokus pada bagaimana menentukan posisi dari peralatan yang digunakan oleh *user* atau disebut dengan metode *positioning.*

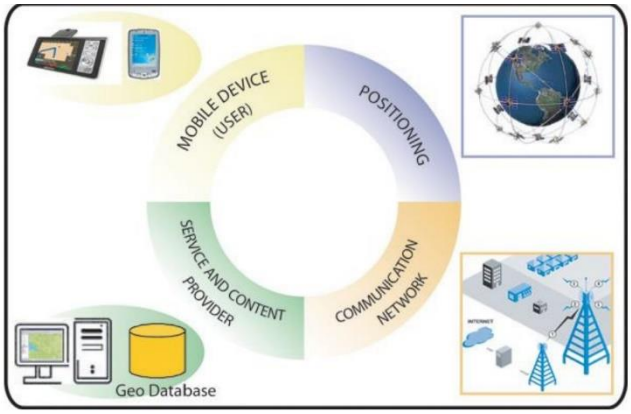
2.2.2 Jenis Layanan *Location Based Service*

Menurut (Anwar et al., 2014),secara garis besar *location based service* dibagi menjadi dua, yaitu :

1. *Pull Service* yaitu layanan yang diberikan jika ada permintaan dari pelanggan akan kebutuhan suatu informasi. Jenis layanan ini dapat dianalogikan seperti mengakses suatu web pada jaringan internet. Untuk *Pull Service* biasa dibagi lagi menjadi dua yaitu berdasakan layanan fungsional seperti memesan *ambulance* dengan menekan tombol pada *device* atau layanan *service* seperti mencari lokasi terdekat dari posisi pengguna.
2. *Push Service* yaitu layanan ini diberikan langsung oleh *service provider* tanpa menunggu permintaan dari pelanggan berupa informasi yang berkaitan dengan kebutuhan pelanggan.

2.2.3 Komponen Location Based Service

Dalam menggunakan layanan berbasis lokasi terdapat lima komponen dalam teknologi (Anwar et al., 2014). Berikut ini komponen dalam teknologi *Location Based Service* seperti yang ditunjukan pada Gambar 2.2 dibawah ini.



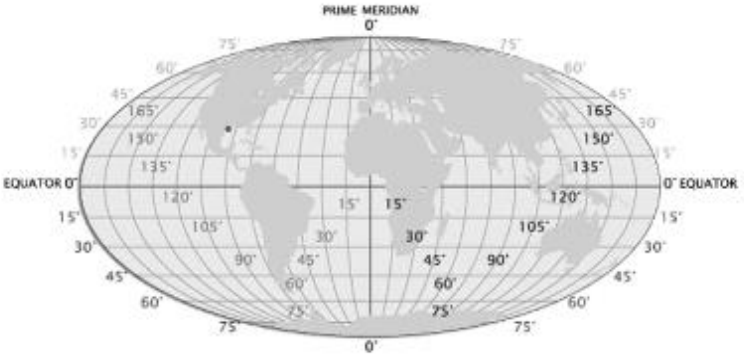
Gambar 2. 2 Komponen *Location Based Service* (Safaat, 2015)

Berikut ini komponen dalam teknologi *location based service* antara lain :

1. *Mobile Device* sebuah alat yang digunakan untuk meminta informasi yang dibutuhkan. Biasanya perangkat yang memungkinkan yaitu *Mobile Phones*, laptop dan perangkat lainnya yang mempunyai fasilitas navigasi.
2. *Communication Network* adalah jaringan selular yang mengirimkan dua pengguna dan permintaan layanan.
3. *Positioning Technology* untuk pengolahaan layanan biasanya posisi pengguna harus ditentukan. Posisi pengguna dapat diperoleh menggunakan jaringan komunikasi atau dengan menggunakan *Global Positioning System* (*GPS*).
4. *Service and Application Provider* adalah penyedia layanan pengguna selular yang bertanggung jawab untuk memproses layanan.
5. *Data and Content Provider* penyedia layanan informasi data yang dapat diminta oleh pengguna.

2.2.4 *Latitude* dan *Longitude*

Latitude dan *Longitude* adalah dua buah garis yang menentukan diperolehnya suatu nilai derajat dari suatu titik yang diukur.



Gambar 2. 3 *Latitude* dan *Longitude* (oloan Lubis & Salim, 2016)

2.2.4.1 *Latitude*

Latitude adalah garis yang melintang diantara kutub utara dan kutub selatan, yang menghubungkan antara sisi timur dan barat bagian bumi. Garis ini memiliki posisi membentangi bumi, sama halnya seperti garis *equator*, tetapi

dengan kondisi nilai tertentu. Garis lintang inilah yang dijadikan ukuran dalam mengukur sisi utara-selatan koordinat suatu titik di belahan bumi. *Latitude* dibedakan menjadi 2 wilayah, yaitu utara yang biasa disebut lintang utara dan selatan atau yang biasa disebut lintang selatan. Dimana nilai koordinat di bagian utara selalu positif dan nilai koordinat di bagian selatan adalah negatif. Berikut nilai-nilai yang dijadikan patokan ukuran garis lintang ini.

1. Garis paling atas (kutub utara) = 90 derajat
2. Garis paling tengah (equator) = 0 derajat
3. Garis paling bawah (kutub selatan) = -90 derajat

Dengan mempersamakan derajat ke dalam bentuk satuan kilometer (km) maka ukurannya seperti berikut ini :

1 derajat latitude = 111 km

1 menit latitude = 1.85 km

2.2.4.2 *Longitude*

Longitude adalah garis membujur yang menghubungkan antara sisi utara dan sisi selatan bumi (kutub). Garis bujur ini digunakan untuk mengukur sisi barat-timur koordinat suatu titik dibelahan bumi. Sama seperti *equator* pada *latitude* yang berada ditengah dan memiliki nilai 0 (nol) derajat, pada *longitude*, garis tengah yang bernilai 0 (nol) derajat disebut garis *prime meridian*. Sedangkan garis yang berada paling kiri memiliki nilai -90 derajat, dan yang paling kanan memiliki nilai 90 derajat. *Longitude* juga dibedakan menjadi 2 wilayah, yaitu bujur timur dan bujut barat, dimana koordinat yang berada di timur selalu bernilai negatif dan sebaliknya. Nilai satuan ukuran derajat menjadi kilometer pada *longitude* juga sama seperti *latitude.*

2.3 Google *Maps API*

Menurut (Agustina et al., 2016), dalam (Daud et al., 2022) menjelaskan bahwa *API* atau *application programming interface* adalah dokumen yang terdiri dari antarmuka, fungsi, kelas, struktur, dan sebagainya untuk membangun perangkat. Dengan *API* ini, *programmer* dapat dengan mudah membongkar dan mengembangkan perangkat lunak atau mengintegrasikannya dengan perangkat lunak lain. *API* dapat dianggap sebagai penghubung aplikasi ke aplikasi lain dan memungkinkan pemrogram untuk menggunakan *system function.*

2.4 *Global Positioning System* (*GPS*)

Menurut (Alfeno & Devi, 2017), *GPS* merupakan singkatan dari *Global Positioning System*, yang merupakan sistem navigasi dengan menggunakan teknologi satelit yang dapat menerima sinyal dari satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke bumi. Sinyal ini diterima oleh alat penerima (*receiver*) di permukaan, dimana *GPS receiver* ini akan mengumpulkan informasi dari satelit *GPS*.

Sebuah *GPS receiver* harus mengunci sinyal minimal tiga satelit untuk menghitung posisi 2D *(latitude dan longitude*) dan track pergerakan. Jika *GPS receiver* dapat menerima empat atau lebih satelit, maka dapat menghitung posisi 3D (*latitude*, *longitude dan altitude*).

2.5 *Android*

2.5.1 Pengertian *Android*

Menurut (Safaat,2015) dalam (Ratnasari et al., 2018) *Android* adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, middleware dan aplikasi. *Android* menawarkan sebuah lingkungan yang berbeda untuk pengembang. Setiap aplikasi memiliki tingkatan yang sama, tidak membedakan antara aplikasi inti dengan aplikasi pihak ketiga. Bahkan pengguna dapat menghapus aplikasi inti dan menggantinya dengan aplikasi pihak ketiga.

Pada awalnya sistem operasi ini dikembangkan oleh perusahaan *Android Inc*. lalu pada tahun 2005, perusahaan ini diakuisasi oleh *Google*.Pada tahun 2007, Google membentuk *Open Handset Alliance*(OHA) sebagai upaya untuk mengembangkan *Android*, yakni sebuah konsorium dari beberapa perusahaan yaitu *Texas Instruments, Broadcom Coorporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, NVDIA, Qualcomm, Samsung Electronics, Sprint Nextel,* dan *T-Mobile*. Tujuan utama dibentuknya OHA adalah untuk mengembangkan standar terbuka untuk perangkat *mobile*.

2.5.2 *Android* SDK (*Software Development Kit*)

Menurut (Safitri & Basuki, 2020) *Android* sdk adalah alat dari kode *Java* yang berjalan dengan lancar dan diizinkan untuk berjalan di perangkat *Android*. Pemrograman *Java* sendiri diperlukan untuk membuat program, sedangkan *Android* SDK diperlukan untuk menjalankan program di *Android*. Oleh karena itu, jika Anda ingin menggabungkan keduanya, Anda memerlukan aplikasi *Android* Studio. Hal ini agar *programmer* dapat menemukan *bug* pada aplikasi yang sedang mereka kembangkan.

2.5.3 *Android* JDK (*Java Development Kit*)

Java Development Kit (*JDK*) adalah sebuah produk yang dikembangkan oleh Oracle yang ditujukan untuk para *developer Java*. Sejak *Java* diperkenalkan, JDK merupakan *Java Software Development Kit* (*SDK*) yang paling sering digunakan.

Menurut (Kusniyati & Pangondian Sitanggang, 2016) *Java Development Kit* merupakan perangkat lunak yang digunakan untuk melakukan proses kompilasi dari kode *Java* menjadi *bytecode* yang dapat dimengerti dan dapat dijalankan oleh *Java Runtime Environtment.*

Java Development Kit merupakan sebuah pernagkat lunak yang berfungsi dalam manajemenisasi aplikasi *Java*. Hal tersebut dikarenakan Anda akan

menggunkan bahasa pemrograman *Java* dalam membuat aplikasi di *Android Studio*. Maka dari itu, *JDK* haruslah terpasang terlebih dahulu sebelum Anda mendownload *Android* Studio (Firly, 2018)

2.6 *Android* Studio

Menurut (Maiyana, 2018), "*Android* Studio merupakan sebuah *IDE* (*Integrated Development Environment*) untuk pengembangan aplikasi *Android*,aplikasi ini dipublikasikan oleh Google pada tanggal 16 mei 2013 dan tersedia secara gratis dibawah lisensi Apache 2.0, *Android* studio ini menggantikan *software* pengembangan *Android* sebelumnya yaitu *Eclipse*".

Sedangkan menurut (Safitri & Basuki, 2020) *Android Studio* adalah Lingkungan Pengembangan Terpadu (*Integrated Development Environment*/*IDE*) resmi untuk pengembangan aplikasi *Android*, yang didasarkan pada *IntelliJ IDEA*. Selain sebagai editor kode dan fitur *developer IntelliJ* yang andal, *Android* Studio menawarkan banyak fitur yang meningkatkan produktivitas anda dalam membuat aplikasi *Android*, seperti:

1. Sistem *build* berbasis *Gradle* yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan terpadu tempat Anda bisa mengembangkan aplikasi untuk semua perangkat *Android*
4. Terapkan Perubahan untuk melakukan push pada perubahan kode dan resource ke aplikasi yang sedang berjalan tanpa memulai ulang aplikasi
5. Template kode dan integrasi *GitHub* untuk membantu Anda membuat fitur aplikasi umum dan mengimpor kode sampel
6. *Framework* dan fitur pengujian yang lengkap
7. Fitur lint untuk merekam performa, kegunaan, kompatibilitas versi, dan masalah lainnya
8. Dukungan *C++* dan *NDK*
9. Dukungan bawaan untuk *Google Cloud Platform*, yang memudahkan

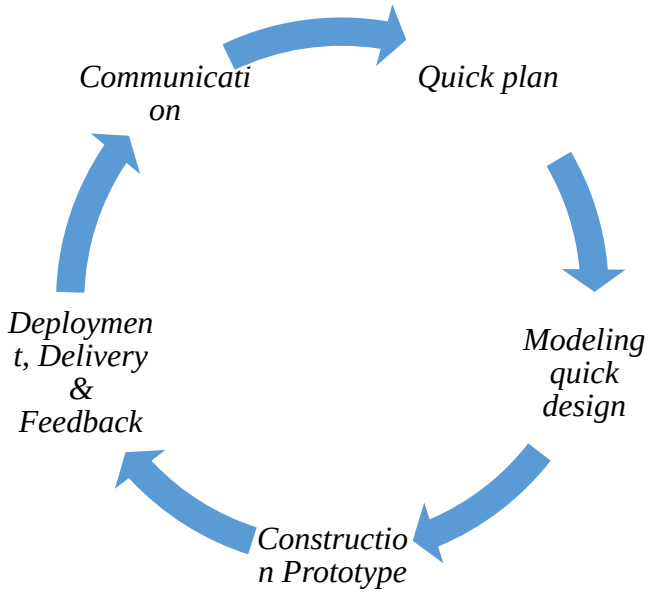
integrasi *Google Cloud Messaging* dan *App Engine*.

Dengan demikian dapat disimpulkan bahwa *Android* studio adalah sebuah *Integrated Development Environment (IDE)* yang digunakan untuk membuat dan mengembangkan sebuah aplikasi yang dapat dijalankan pada platform *Android*. *Android* studio berbasis *IntelliJ IDEA*,dimana *IDE* ini digunakan untuk bahasa pemrograman *Java*,untuk *layout* atau tampilannya menggunakan bahasa *XML*. Untuk *deploy* ke perangkat *Android*, *Android* studio telah terintegrasi dengan *Android Software Development Kit* (SDK).

2.7 *Java*

(Bambang Haryanto, 2011:2,Esensi-esensi Bahasa Pemrograman *Java*. Yogyakarta: Andi) dalam (Fitria, 2021) menjelaskan, "*Java* merupakan bahasa berorientasi objek untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, aplikasi untuk perangkat cerdas yang dapat berkomunikasi lewat internet/jaringan komunikasi. Melalui teknologi *Java*, kita dapat mengembangkan banyak aplikasi yang dapat digunakan pada lingkungan yang berbeda, seperti pada *desktop, Mobiile Internet* dan lainnya.

2.8 Metode *Prototype*



Gambar 2. 4 Metode *Prototype* (Pressman Roger, 2012)

(Pressman Roger, 2012) menguraikan bahwa dalam melakukan perancangan sstem yang akan dikembangkan dapat menggunakan metode *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 *prototype* :

1. *Communication*, yaitu analisis terhadap kebutuhan pengguna menggenai spesifikasi kebutuhan yang diinginkan.

2. *Quick Plan* yaitu pembuatan desain secara umum untuk selanjutnya dikembangkan kembali.
3. *Modeling Quick Design*, yaitu pada tahap ini pengembang membuat perancangnan sistem.
4. *Construction Prototype*, Setelah tahap pemodelan, maka pengembang melakukan pengkodean program.
5. *Deployment, Delivery & Feedback* tahap pengkodean program dibarengi oleh tahapan imlementasi dan pengujian sistem.

2.9 Pengujian *ISO* 25010

Pengujian *ISO* 25010 adalah bagian dari *Sistem and Software Quality Requirements and Evaluation* (*SQuaRE*) yang merupakan versi lanjutan dari *ISO* 91261. *ISO* 25010 telah direvisi secara teknis dengan menambahka beberapa struktur dan bagian dari standar model kualitas. Tujuan dari penggunaan kualitas ini adalah untuk mengukur sejauh mana produk atau sistem tersebut bisa digunakan oleh pengguna untuk memenuhi kebutuhan dalam mencapai tujuan yang diinginkan dengan efisiensi, efektivitas, kepuasan dalam konteks penggunaan yang spesifik, dan bebas dari resiko (Wattiheluw et al., 2019).

Menurut (Harun, 2018) *ISO* 25010 ini terdiri dari delapan karakteristik yang dibagi menjadi beberapa sifat static dan dinamis perangkat lunak dari sistem komputer seperti berikut :

1. *Functional Suitability*, merupakan sistem atau produk yang memberikan fungsional untuk memenuhi kebutuhan saat sistem atau produk tersebut digunakan pada keadaan tertentu.
2. *Reliability*, merupakan tingkat dimana suatu sistem atau produk dapat mempertahankan kinerjanya pada level tertentu ketika digunakan pada keadaan tertentu.

3. *Performance EfficIency*, merupakan tingkat dimana sistem atau produk menyediakan performa yang baik dengan sejumlah *resource* yang akan digunakan pada sistem atau produk.
4. *Usability*, merupakan tingkat dimana pada suatu sistem atau produk mudah dimengerti, mudah dipakai, dan menarik untuk digunakan.
5. *Security*, merupakan tingkat dimana pada suatu sistem atau produk menyediakan layanan untuk melindungi akses, penggunaan, modifikasi, pengrusakan, ataupun pengungkapan yang berbahaya.
6. *Compatibality*, merupakan kemampuan pada suatu komponen atau sistem untuk bertukar informasi.
7. *Maintainability*, merupakan tingkat dimana pada suatu sistem atau produk dapat dimodifikasi yang meliputi perbaikan, pengembangan untuk menyesuaikan dengan lingkungan, modifikasi pada kriteria, dan spesifikasi fungsi.
8. *Portability*, merupakan tingkat dimana pada suatu sistem atau produk dapat dipindahkan dari satu ruang ke ruang lainnya.

2.10 *Unified Modeling Languange* (*UML*)

Unified Modeling Language (*UML*) merupakan suatu metode pemodelan visual yang sering digunakan dalam perancangan dan pembuatan *software* yang berorientasi pada objek.

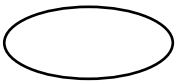
UML merupakan standar sebuah penulisan atau *blueprint* dimana didalamnya terdapat sebuah bisnis proses, penulisan kelas-kelas dalam Bahasa yang spesifik. (Prihandoyo, 2018).Terdapat beberapa diagram *UML* yang sering digunakan dalam pengembangan sebuah sistem, yaitu :

1. *Use Case Diagram*

   Use Case diagram adalah gambaran dari fungsional yang diharapkan dari suatu sistem, dan merepresentasikan interaksi antara aktor dan sistem. Dalam *Use Case* ada aktor yang di deskripsikan sebagai entitas manusia atau sebuah

sistem yang melakukan pekerjaan di sistem *Use Case* diagram biasanya berisi kasus pengguna, *actor*, ketergantungan, *generalisasi*, dan hubungan *asosiasi* (Prihandoyo, 2018). Pada **Tabel 2.2** merupakan contoh dari *Use Case* Diagram

Tabel 2.2 Simbol dan Keterangan *Use Case Diagram*

Simbol	Keterangan
 *Actor*	Mewakili peran orang, sistem yang lain atau alat ketika berkomunikasi dengan *use case*
Use case 	*Fungsionalitas* yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit dan aktor.
Association/Asosiasi 	Abstraksi dari penghubung actor dan *use case* yang berpartisipasi.
Extend/Extensi <<extend>> 	Menunjukan bahwa suatu use case merupakan tambahan *fungsional* dari *use case* lainnya jika suatu kondisi terpenuhi.
Include/ *Use case* <<*include*>> 	Menunjukan bahwa suatu *use case* seluhruhnya merupakan *fungsional* dari *use case* lainnya.

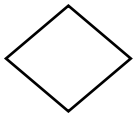
Tabel 2.3 Simbol dan Keterangan *Use Case Diagram* (lanjutan)

Simbol	Keterangan
Generalization/generalisasi 	Hubungan generalisasi dan spesialisasi antara dua buah *use case* yang mana fungsi yang satu lebih umum dari yang lainnya.

2. *Activity Diagram*

   Activity Diagram menurut (Rosa, 2016), dalam (Prihandoyo, 2018) merupakan suatu gambaran dari aliran – aliran aktivitas yang ada didalam sistem yang nantinya akan berjalan simbol – simbol *activity diagram* dapat dilihat pada **Tabel 2.4** berikut

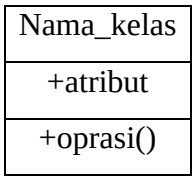
Tabel 2.4 Simbol *Activity Diagram*

Simbol	Keterangan
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas yang biasanya diawali dengan kata kerja.
Status awal 	Status awal sistem, sebuah diagram aktivitas bagaimana objek dibentuk atau diawali.
Status akhir 	Status akhir sistem, sebuah diagram aktivitas bagaimana objek dibentuk dan di akhiri
Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktifitas lebih dari satu.
Penggabungan/*join* 	Asosiasi penggabungan dimana lebih dari satu aktifitas digabungkan menjadi satu.

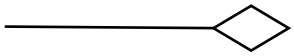
3. *Class Diagram*

Class Diagram menurut (Rosa, 2016), dalam (Prihandoyo, 2018) merupakan suatu gambaran dari struktur dan deskripsi dari *class, package* dan objek yang saling berhubungan seperti pewarisan, *asosiasi*, dan sebagainya. *Class Diagram* biasanya berisikan, kelas, antarmuka, kolaborasi, ketergantungan, *generalisasi*, dan hubungan asosiasi. Pada **Tabel 2.5** berikut adalah contoh dari *Class Diagram*.

Tabel 2.5 Simbol *Class Diagram*

Simbol	Keterangan
Kelas 	Himpunan dari objek – objek yang berbagi atribut serta oprasi yang sama
Nary Association 	Upaya untuk menghindari asosiasi lebih dari dua objek
Collaboration 	Deskripsi dari urutan aksi – aksi yang ditampilkan sistem yang menghasilkan
Realization 	Oprasi yang benar – benar dilakukan oleh suatu objek.
Association 	Apa yang menghubungkan antara objek satu dengan objek lainnya

Tabel 2.6 Simbol *Class Diagram* (lanjutan)

Simbol	Keterangan
Aggregation 	Relasi antar kelas dengan makna semua bagaian (*whole part*)
Dependency 	Hubungan dimana perubahan yang terjadi pada suatu element mandiri (*independant*) akan mempengaruhi element yang bergantung pada elemen yang tidak mandiri

4. *Sequence Diagram*

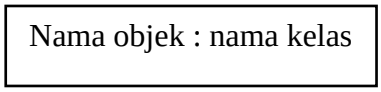
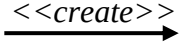
   Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *massage* yang dikirimkan dan diterima antar objek. Untuk menggambarkan diagram sekuen maka harus diketahui objek – objek yang terlibat dalam sebuah *use case* beserta metode – metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

   Berikut adalah simbol – simbol yang ada pada digram sekuen :

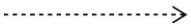
Tabel 2. 7 *Squence Diagram*

Simbol	Keterangan
Actor  Nama *Actor*	*Actor*, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tetapi aktor belum tentu merupakan orang, menggunakan kata benda

Tabel 2.8 *Squence Diagram* (lanjutan)

Simbol	Keterangan
Garis hidup/ *lifeline*	Menyatakan kehidupan suatu objek
Objek 	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan didalamnya
Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yaang dibuat
Pesan tipe *call* 	Menyatakan suatu objek menggail operasi / metode yang ada pada objek lain atau dirinya sendiri. Arah panah mengarah pada objek yang memiliki operasi / metode, karena ini menggil operasi / metode maka operasi /metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi

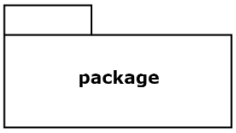
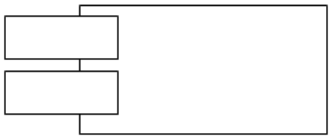
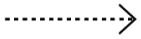
Tabel 2.9 *Squence Diagram* (lanjutan)

Simbol	Keterangan
Pesan tipe *return* *return* 	Menyatakan suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian keobjek
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*

5. *Component Diagram*

   Diagram komponen atau *component diagram* dibuat untuk menunjukan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem (Rosa, 2016). Simbol diagram komponen dapat dilihat pada **Tabel 2.10**

Tabel 2.10 Simbol *Component Diagram*

Simbol	Keterangan
Package 	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen
Component 	Merupakan komponen sistem
Dependency 	Ketergantungan atau *dependency* adalah kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai

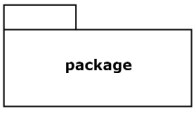
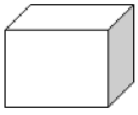
Tabel 2.11 Simbol *Component Diagram* (lanjutan)

Simbol	Keterangan
Interface/Antarmuka 	Antar muka atau *interface* merupakan antarmuka sama dengan *interface* pada pemograman berorientasi objek, yaitu sebagai antar muka komponen agar tidak mengakses lansung komponen
Link	Menggambarkan sebuah relasi antar komponen

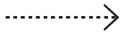
6. *Deployment diagram*

   Menurut (Rosa, 2016), diagram *deployment* menunjukan konfigurasi komponen dalam proses eksekusi dari sebuah aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal – hal seperti sistem tambahan dan sistem *client/server.* Simbol dapat dilihat pada **Tabel 2.12**

Tabel 2.12 Simbol *Deployment Diagram*

Simbol	Keterangan
Package 	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen
Node 	*node* biasa mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*sofware*), jika di dalam *node* disertakan komponen untuk mengkonsistenkan rancangan

Tabel 2. 13 Simbol *Deployment Diagram* (Lanjutan)

Simbol	Keterangan
Dependency 	Ketergantungan atau *dependency* adalah kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
Link	Menggambarkan sebuah relasi antar komponen

2.11 *My SQL*

Menurut (Risnandar et al., 2013) bahwa *MySQL* merupakan basis data yang bersifat *open source* sehingga banyak digunakan di dunia. Walaupun gratis, *My SQL* tetap berkualitas dan sudah cukup memberikan performa yang memadai.

Sedangkan menurut (Alesandro, 2021) *MySQL* digunakan untuk pencadangkan model data, pengembangan *SQL*, dan peralatan administrasi yang komprehensif untuk konfigurasi *server* basis data, dan administrasi pengguna.

2.12 XAMPP

Menurut Aryanto dalam (Kristania et al., 2017), "*XAMPP* merupakan sebuah aplikasi perangkat lunak pemrograman dan *database* yang di dalamnya terdapat berbagai macam aplikasi pemrograman seperti : *Apache HTTP server*, *MySQL*, *database*, Bahasa pemrograman *PHP* dan *perl*.".

Xampp merupakan perangkat lunak pemrograman dan database atau paket web server berbasis *open source* yang dapat dipasang pada beberapa sistem operasi yang ada seperti *Windows, Linux,* dan *Mac OS*. Yang mana didalamnya terdapat berbagai macam aplikasi pemrograman seperti : *Apache HTTP server, MySQL, database,* Bahasa Pemrograman *PHP* dan *Perl.*