BAB II
TINJAUAN PUSTAKA

2. 1. Perangkat Keras (*Hardware*)

Perangkat keras (*hardware*) adalah suatu perangkat komputer yang terdiri dari komponen-komponen elektronik berupa benda. Komponen Perangkat keras (*hardware*) pada sebuah sistem informasi yakni perangkat penyimpanan data, peralatan *input* dan peralatan *output*, serta peralatan komunikasi data. Peralatan *input* merupakan suatu alat yang digunakan untuk menerima *input* yang dimasukkan pada suatu sistem berupa signal *input*. Contoh peralatan *input* yakni *keyboard*, *scanner*, dan lain sebagainya. Sementara peralatan *output* merupakan suatu alat keluaran untuk menampilkan data yang telah diproses. Contoh peralatan *output* seperti *hard copy, soft copy*, dan lain sebagainya. Sedangkan peralatan komunikasi data merupakan alat yang dapat menyampaikan suatu informasi berupa *text* maupun gambar. Contoh peralatan komunikasi seperti terminal dan modem. (Kustina, Ketut Tanti, dkk, 2022:26)

2. 2. Perangkat Lunak (*Software*)

Perangkat lunak aplikasi merupakan *program* yang ditujukan untuk menyelesaikan suatu permasalahan dalam aplikasi yang tertentu yang sudah dibuat oleh pabrik pembuat perangkat lunak aplikasi. Program aplikasi dibuat dengan menggunakan perangkat lunak bahasa (*language software*). Ali (dalam Sudarso, 2022:5)

Software adalah sekumpulan data elektronik yang disimpan dan diatur oleh komputer. Data eletronik yang disimpan dapat berupa program atau instruksi yang akan menjalankan perintah. *Sofware* dapat diartikan juga sebagai segala jenis program yang digunakan untuk pengoperasian komputer dan peralatannya. (Indra, dalam Sudarso, 2022:5)

2. 3. *Internet*

Internet adalah rangkaian hubungan jaringan komputer yang dapat diakses secara umum diseluruh dunia, yang mengirimkan data dalam bentuk paket data berdasarkan standar *Internet Protocol* (IP). *Internet* adalah kumpulan jaringan dari jaringan-jaringan komputer dunia yang terdiri dari jutaan unit-unit kecil, seperti jaringan pendidikan, jaringan bisnis, jaringan pemerintahan dan lain-lain, yang secara bersamaan menyediakan layanan informasi seperti *e-mail*, *online chat*, *transfer file* dan saling keterhubungan (*linked*) antara satu halaman *web* dengan sumber halaman *web* yang lainnya. (Yuhefizar, 2008)

2. 4. Data

Data adalah suatu atribut yang melekat pada suatu objek tertentu, berfungsi sebagai informasi yang dapat dipertanggungjawabkan, dan diperoleh melalui suatu metode/instrumen pengumpulan data. (Herdiansyah, 2013:8)

Data adalah bahan mentah yang perlu dilakukan pengolahan sehingga menghasilkan informasi atau keterangan, baik *kualitatif* maupun *kuantitatif* yang menunjukkan fakta sehingga dapat memberi manfaat bagi peneliti atau memberi gambaran kepada peneliti tentang kondisi atau suatu keadaan. Sedangkan informasi adalah sekumpulan data yang sudah diolah sehingga menghasilkan suatu analisa untuk digunakan oleh pihak yang membutuhkan. Gunardi dan Widianto (dikutip Wahono dan Ali, 2021:227)

2. 5. Aplikasi

Aplikasi adalah sebuah perangkat lunak yang berisi sebuah *coding* atau perintah yang dimana bisa diubah sesuai dengan keinginan. Aplikasi adalah sebuah perangkat lunak yang dimana tujuannya adalah agar bisa melayani setiap aktivitas komputerisasi yang dilakukan oleh pengguna (Werstantia dan Syani, 2018).

Perangkat lunak aplikasi (*application software*) adalah program yang bisa dipakai oleh pemakai untuk melakukan tugas-tugas yang spesifik; misalnya untuk

membuat dokumen, memanipulasi foto, atau membuat laporan keuangan (Kadir, 2005).

Aplikasi adalah sebuah perangkat lunak (*software*) *computer*. Perangkat lunak adalah program *computer* yang terasosiasi dengan akomondasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaan (Hengky, 2012). Selain itu, aplikasi juga dapat didefinisikan sebagai penerapan, menyimpan sesuatu baik berupa data, permasalahan, pekerjaan kedalam suatu sarana ataupun media yang bisa digunakan untuk diterapkan menjadi sebuah bentuk yang baru. Maka dapat disimpulkan aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanankan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju.

2. 6. Pembayaran

Pengertian pembayaran menurut UU No.23 Pasal 1(1999:6) menyatakan bahwa Pembayaran mencakup seperangkat aturan, lembaga, dan mekanisme yang digunakan untuk melakukan pemindahan dana guna memenuhi suatu kewajiban yang timbul dari suatu kegiatan ekonomi.

Pengertian pembayaran menurut Hasibuan (2010:117) yaitu: Berpindahnya hak pemilikan atas sejumlah uang atau dan dari pembayar kepada penerimanya, baik langsung maupun melalui media jasa-jasa perbankan. Dari definisi diatas, penulis dapat menarik kesimpulan bahwa Pembayaran adalah mekanisme yang dilakukan untuk pemindahan mata uang menjadi barang, jasa atau informasi dari pembayar kepada penerima, baik langsung maupun melalui media jasa-jasa perbankan (Sujarwo, Rahmad Anton. 2019).

2. 7. Quick Response Code Indonesian Standard

Gambar 2. 1 QRIS

Menurut Andika Isma, dkk, dalam buku berjudul *E-Commerce* dan *Internet of Things (IOT)* (2023:90). Pembayaran menggunakan QR *code* yaitu kode yang diciptakan untuk membaca atau mentransfer data yang terdapat di dalamnya dengan cepat dan mudah. Pembeli dapat melakukan pembayaran dengan memindai kode QR yang telah disediakan penjual.

Bagian-Bagian QR *Code* :

a. *Finding Pattern*

Gambar 2. 2 *Finding Pattern*

Finding pattern adalah bagian dalam QR *code* yang berbentuk kotak dan jumlahnya ada tiga. Terdapat pola yang berfungsi untuk mendeteksi posisi dari QR *Code*.

b. *Alignment Pattern*

Gambar 2. 3 *Alignment Pattern*

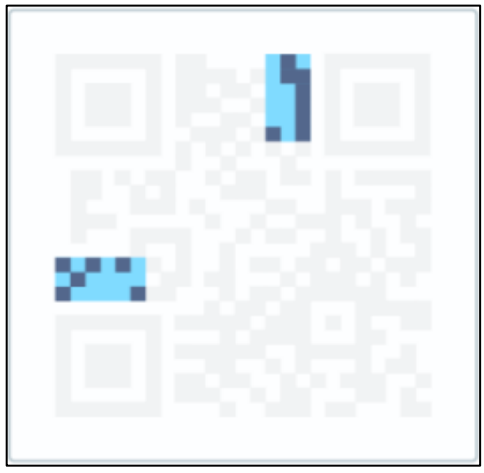
Alignment pattern merupakan pola yang digunakan untuk memperbaiki penyimpangan QR *Code* terutama distorsi *non linier*.

c. *Timing Pattern*

Gambar 2. 4 *Timing Pattern*

Timing pattern merupakan pola yang digunakan untuk konfigurasi data *grid* dan identifikasi koordinat pusat dengan QR *Code*, dibuat dalam bentuk modul hitam putih bergantian.

d. *Version Information*



Gambar 2. 5 *Version Information*

Version information adalah bagian yang memberi informasi versi *QR code*, versi terkecil adalah 1 (21 x 21) modul dan versi terbesar ada 40 (177 x 177) modul.

e. *Format Information*

Gambar 2. 6 *Format Information*

Format information merupakan informasi tentang *error correction level* dan *mask pattern.* Dengan *format information, scanner* akan lebih mudah

melakukan pemindaian *QR code* untuk menampilkan data yang dimuatnya pada pengguna.

f. *Data and Error Correction Keys*

Gambar 2. 7 Data and *Error Correction Keys*

Data merupakan struktur kode sebagai tempat dimana semua data disimpan. Terdapat juga *error correction keys* yang menjaga data tetap dapat dipindai meski kode rusak sebanyak 30%.

g. *Quiet Zone*

Gambar 2. 8 *Quiet Zone*

Quiet zone merupakan bagian kosong yang berada di area terluar QR *Code* memiliki kegunaan untuk menegaskan struktur dan membuatnya lebih mudah dipindai.

2. 8. Suara

Suara merupakan sesuatu yang terdengar (didengar) atau ditangkap oleh telinga. Suara terjadi karena adanya suatu getaran sehingga menciptakan suara yang dapat didengar oleh indera pendengaran manusia. (Fitri, Nadia Aisah. Dkk. 2022)

2. 9. Android

Menurut Nadia Firly, dalam bukunya yang berjudul *Create Your Own Android Application* (2018:1). Beberapa tahun belakangan ini, dunia dihebohkan dengan adanya *platform* baru yang kian menguasai pasar global. Saat ini pun, lebih dari setengah *persen* pengguna ponsel dunia telah menggunakan sistem operasi tersebut, *platform* tersebut adalah *android*.

Menurut Yudha Yudhanto dan Ardhi Wijayanto, dalam buku yang berjudul Mudah Membuat dan Berbisnis Aplikasi *Android* dengan *Android* Studio (2017:1) *Android* adalah sistem operasi yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. *Android* awalnya dikembangkan oleh *Android*, Inc, dengan dukungan finansial dari *Google* yang kemudian membelinya pada tahun 2005.

2. 10. Visual Studio Code

Visual Studio Code adalah kode *editor* sumber yang dikembangkan oleh *Microsoft* untuk *Windows*, Linux dan macOS. Ini termasuk dukungan untuk *debugging*, *control git* yang tertanam dan GitHub, penyorotan *syntax*, penyelesaian kode cerdas, *snippet*, dan *refactoring* kode. Ini sangat dapat disesuaikan, memungkinkan pengguna untuk mengubah tema, pintasan *keyboard*, preferensi, dan menginstal ekstensi yang menambah fungsionalitas tambahan (Agustini dan

Kurniawan, 2019). Teks editor ini secara langsung mendukung bahasa pemrograman *JavaScript*, *Typescript*, dan Node.js, serta bahasa pemrograman lainnya dengan bantuan plugin yang dapat dipasang *via marketplace Visual Studio Code* (seperti C++, C#, Python, Go, Java, dst).

2. 11. Flutter

Flutter adalah sebuah *framework open-source* atau SDK yang dikembangkan untuk membangun antarmuka (*Customer interface*/UI) aplikasi yang memiliki kinerja tinggi serta dapat dipublikasi ke *platform Android* dan iOS dari *codebase* tunggal. *Flutter* menggunakan bahasa pemrograman *dart* yang pastinya terasa familiar dengan bahasa pemrograman *java* atau *javascript*. *Dart* merupakan bahasa pemrograman yang dikembangkan oleh *Google* untuk kebutuhan umum (*general-purpose programming language*). *Dart* merupakan bahasa pemrograman tersebut termasuk ke dalam bahasa pemrograman bertipe dinamis. *Dart* mudah digunakan dalam pengembangan aplikasi *modern* dan memiliki implementasi berkinerja tinggi serta dapat digunakan sebelum dikompilasi. (Suhendro, Jauzaa Maylia, dkk. 2021)

2. 12. Kotlin

Kotlin adalah bahasa pemrograman *modern*, disajikan secara *statis* yang berjalan pada *platform Java Virtual Machine* (JVM). *Java Virtual Machine* (JVM) adalah mesin yang menyediakan lingkungan *runtime* untuk menjalankan kode dalam aplikasi Java. Kotlin dibuat karena terinspirasi dari bahasa pemrograman lain yang terlebih dahulu sudah ada seperti Java, C#, dan Javascript. Hal tersebut membuat pengembangnya berusaha untuk menjadikan kotlin sebagai sebuah bahasa pemrograman yang tidak terlalu rumit dan bisa dnegan mudah dipelajari. (Isnardi, dkk.2021)

2. 13. Firebase

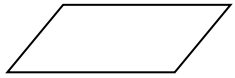
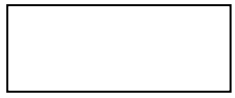
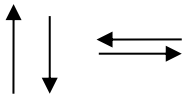
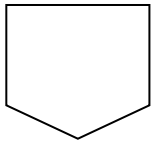
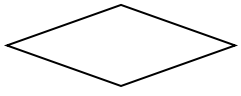
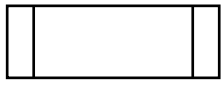
Firebase adalah suatu layanan dari *Google* yang digunakan untuk mempermudah para pengembang aplikasi dalam mengembangkan aplikasi. Penggunaan *firebase* diperlukannya akses *internet* dalam menjalankan aplikasi tersebut. *Firebase Realtime Database* memungkinkan pengguna untuk membuat aplikasi *kolaboratif* dan kaya fitur dengan menyediakan akses yang aman ke *database*, langsung dari kode sisi *client*. Data disimpan di *drive* lokal. Bahkan saat *offline* sekalipun, peristiwa *Realtime* terus berlangsung, sehingga pengguna akhir akan merasakan pengalaman yang *responsif*. Ketika koneksi perangkat pulih kembali, *Realtime Database* akan menyinkronkan perubahan data lokal dengan *update* jarak jauh yang terjadi selama *client offline*, sehingga setiap perbedaan akan otomatis digabungkan. *Realtime Database* adalah *database* NoSQL, sehingga memiliki pengoptimalan dan fungsionalitas yang berbeda dengan *database* terkait. (Suhendro, Jauzaa Maylia, dkk. 2021)

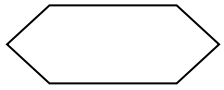
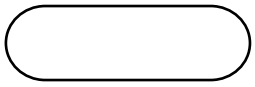
2. 14. Flowchart

Flowchart atau bagan alir merupakan kumpulan dari notasi diagram simbolik yang menunjukkan aliran data dan urutan operasi dalam sistem. Bagan alir (*flowchart*) merupakan metode teknik analisis yang dipergunakan untuk mendeskripsikan sejumlah aspek dari sistem informasi secara jelas, ringkas, dan logis (Mardi, 2014).

Flowchart adalah suatu jenis diagram yang merepresentasikan algoritma atau langkah-langkah instruksi yang berurutan dalam suatu sistem *Flowchart* adalah salah satu penyajian yang sistematis tentang proses dan logika dari kegiatan, penanganan suatu informasi atau penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program (Sari, 2017) Sedangkan bagan alir dokumen merupakan simbol-simbol standar yang digunakan oleh analis sistem untuk menggambarkan seperti apa bagan alir dokumen suatu sistem. Sistem biasanya menggunakan bagan alir (*flowchart*) untuk menggambarkan suatu sistem dan prosedur yang berjalan di dalamnya, menggunakan simbol seperti pada Tabel 2.1.

Tabel 2. 1 Simbol-simbol *Flowchart*

No.	Simbol	Keterangan
1.		Simbol *input* atau *output* (*input/output symbol)* digunakan untuk mewakili data *input/output.*
2.		Simbol proses digunakan untuk mewakili suatu proses .
3.		Simbol garis alir (*flow lines symbol*) digunakan untuk menunjukan arus dari proses.
4.		Simbol penghubung (*connector symbol*) digunakan untuk menunjukan sambungan dari bagan alir yang terputus di halaman yang masih sama.
5.		Simbol penghubung offline (*offline connector symbol*) digunakan untuk menunjukan sambungan dari bagan alir yang terputus di halaman berbeda.
6.		Simbol keputusan (*decision symbol*) digunakan untuk suatu penyeleksian kondisi di dalam program.
7.		Simbol proses terdefinisi (*predefined process symbol*) digunakan untuk

		menunjukan operasi yang rinciannya ditunjukkan di tempat lain.
8.		Simbol persiapan (*preparation symbol*) digunakan untuk memberi nilai awal suatu besaran .
9.		Simbol titik terminal (terminal point symbol) digunakan untuk menunjukan awal dan akir dari suatu proses.

Sumber: Sari (2017)

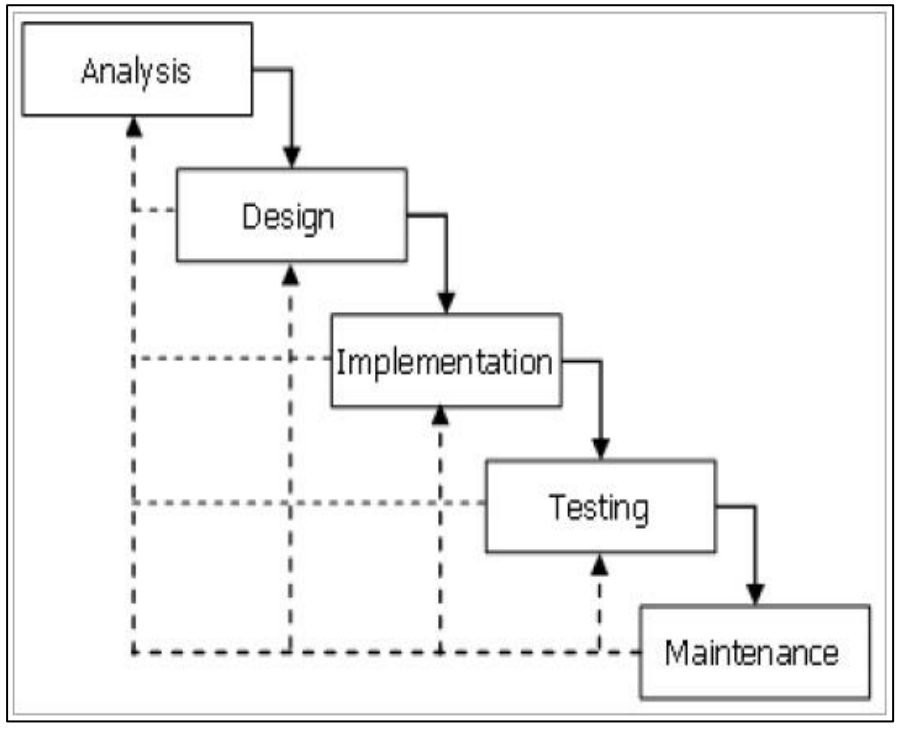
2. 15. Metode Pengembangan Sistem

Metode air terjun atau yang sering disebut metode *waterfall* sering dinamakan siklus hidup klasik (*classic life cycle)*, nama model ini sebenarnya adalah "*Linear Sequential Model*" dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna lalu berlanjut melalui tahapan-tahapan perencanaan (*planning*), permodelan (*medelling*), konstruksi (*contruction*), serta penyerahan sistem ke para pengguna (*deployment*), yang diakhiri dengan dukungan pada perangkat lunak lengkap yang dihasilkan.

2.15.1. Waterfall

Metode *waterfall* merupakan Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis,desain pengodean,pengujian dan tahap pendukung atau support" (Rosa dan Shalahuddin, 2017). Model *waterfall* pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai didalam *Software Engineering* (SE). Saat ini model *waterfall* merupakan model pengembangan perangkat lunak yang sering digunakan. Model pengembangan ini melalakukan pendekatan secara sistematis dan berurutan. Disebut *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan

berjalan berurutan. Model pengembangan ini bersifat linear dari tahap awal pengembangan sistem yaitu tahap perencanaan sampai tahap akhir pengembangan sistem yaitu tahap pemeliharaan. Tahapan dari metode *waterfall* dapat dilihat pada gambar dibawah ini :



Gambar 2. 9 Tahapan Metode *Waterfall*

a. Analisis Kebutuhan Perangkat Lunak

   Proses pengumpulan kebutuhan diakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh user. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

b. Design

   Design perangkat lunak adalah proses multi langkah yang fokus pada *design* pembuatan proram perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini

mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi *design* agar dapat diimplementasikan menjadi program pada tahap selanjutnya. *Design* perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

c. Implementation

Design harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan design yang telah dibuat pada tahap *design*.

d. Testing

Testing atau pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

e. Pendukung (*support*) atau Pemeliharaan (*maintenance*)

Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.15.2. Pengertian UML (*Unified Modelling Languange*)

UML (*Unified Modelling Language*) adalah salah satu standar bahasa yang banyak digunakan didunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasikan objek (Rosa Dan Shalahuddin, 2017). UML (*Unified Modeling*

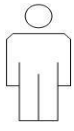
Language) merupakan pengganti dari metode analisis berorientasi *object* dan *design berorientasi object* (OOAD&D/*object oriented analysis and design*) yang dimunculkan sekitar akhir tahun 80-an dan awal tahun 90-an. UML merupakan gabungan dari metode Booch, Rumbaugh (OMT) dan Jacobson. Tetapi UML mencakup lebih luas daripada OOAD. Pada pertengahan saat pengembangan UML, dilakukan standarisasi proses dengan OMG (*Object Management Group*) dengan harapan UML bakal menjadi bahasa standar pemodelan pada masa yang akan datang (yang sekarang sudah banyak dipakai oleh berbagai kalangan) (Pratama, 2017).

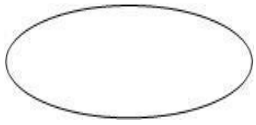
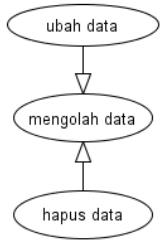
2.15.3. Pengertian *Use Case Diagram*

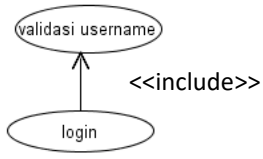
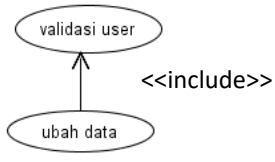
Use Case merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat (Rosa dan Shalahuddin, 2017). *Use case* diagram adalah gambaran grafis dari beberapa atau semua *actor, use case*, dan interaksi diantaranya yang memperkenalkan suatu sistem. *Use case* diagram tidak menjelaskan secara detil tentang penggunaan *use case,* tetapi hanya memberi gambaran singkat hubungan antara *use case,* aktor, dan sistem. Di dalam *use case* ini akan diketahui fungsi- fungsi apa saja yang berada pada sistem yang dibuat. (Pratama, 2017).

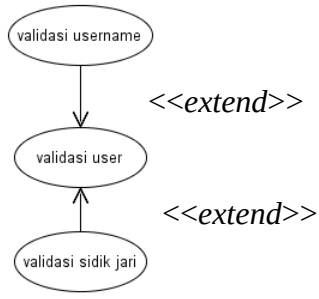
Berikut ini adalah simbol- simbol yang digunakan dalam *use case* diagram serta keterangannya seperti yang dijelaskan pada tabel 2.2.

Tabel 2. 2 Simbol *Use Case Diagram*

No	Simbol	Keterangan
1		Aktor, mewakili peran orang, 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, tapi aktor belum tentu

		merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
2		Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*.
3		*Association*, komunikasi antar aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* yang memiliki interaksi dengan actor.
4		*Generalization*, hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:  arah panah mengarah pada *use case* yang menjadi generalisasinya (umum).

5	<<include>>	Relasi tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini ada dua sudut pandang yang cukup besar mengenai include di *use case*: *a.* *Include* berarti *use case* yang ditambahkan akan selalu di panggil saat *use case* tambahan dijalankan, misalnya pada kasus berikut:  *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang di tambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:  kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.
	<<extend>>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang di tambahkan dapat berdiri

6		sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misalnya  arah panah mengarah pada *use case* yang ditambahkan; biasanya *use case* yang menjadi *extend*-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya.

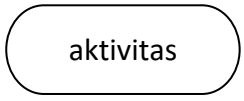
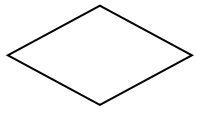
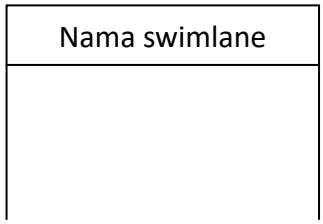
(*Sumber*: Rosa dan Shalahuddin (2016:156-158))

2.15.4. Pengertian Activity Diagram

Activity diagram adalah diagram yang menggambarkan sifat dinamis secara alamiah sebuah sistem dalam bentuk model aliran dan kontrol dari aktivitas ke aktivitas lainnya, Menurut Apol dan Radtyo dalam (Fauzi et al., 2019).

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol- simbol yang digunakan dalam *activity* diagram, Menurut (Rosa dan Shalahuddin 2017). Berikut ini adalah simbol- simbol yang digunakan dalam activity diagram serta keterangannya seperti yang dijelaskan pada tabel 2.3.

Tabel 2. 3 Simbol *Activity Diagram*

No	Simbol	Deskripsi
1.	Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.	Penggabungan / *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6.	*Swimlane*  atau	*Swimlane* memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

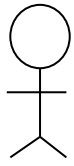
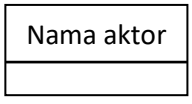
		

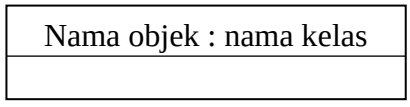
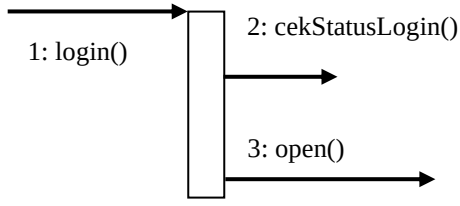
(*Sumber*: Rosa dan Shalahuddin (2016:162-163))

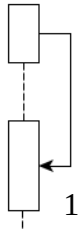
2.15.5. Pengertian Sequence Diagram

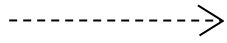
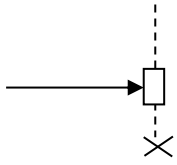
Menurut Rosa dan Shalahuddin (2016:165), "Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansikan menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat *scenario* yang ada pada *use case*".). Berikut ini adalah simbol- simbol yang digunakan dalam activity diagram serta keterangannya seperti yang dijelaskan pada tabel 2.4.

Tabel 2. 4 Simbol *Sequence Diagram*

No	Simbol	Deskripsi
1.	Aktor  Atau 	Orang, 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, tapi aktor belum tentu merupakan orang; biasanya dinyatakan

	tanpa waktu aktif	menggunakan kata benda di awal frase nama *actor*.
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, misalnya  maka cek Status Login () dan open() dilakukan di dalam metode login() Aktor tidak memiliki waktu aktif.

5.	pesan tipe *create* <<create>>	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
6.	Pesan tipe *call* 1 : nama_metode()	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  1: nama_metode() arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
7.	Pesan tipe *send* 1 : masukan	Menyatakan bahwa suatu objek mengirimkan data/ masukan/ informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.

8.	Pesan tipe *return* 1 : keluaran 	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.
9.	Pesan tipe *destrory*  <<destory>>	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy.*

(*Sumber*: Rosa dan Shalahuddin (2016:165-167))

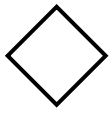
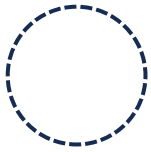
2.15.6. Pengertian *Class* Diagram

Class Diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan perancangan berorientasi objek". (Tohari (2014:83))

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Diagram kelas dibuat agar pembuat program atau programmer membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron (Rosa dan Shalahuddin (2013:141))

Tabel 2. 5 Simbol *Class* Diagram

No	Gambar	Nama	Keterangan
1.		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan

			struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2.		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3.		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4.		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu *actor*.
5.		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6.		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
7.		*Association*	Apa yang menghubungkan antara objek satu dengan lainnya.

2. 16. Teori Pengujian Sistem

Menurut Simarmata (2010) mengatakan pengujian adalah sebuah proses terhadap aplikasi atau program untuk menemukan segala kesalahan dan segala kemungkinan yang akan menimbulkan kesalahan sesuai dengan spesifikasi perangkat lunak yang telah ditentukan sebelum aplikasi tersebut diserahkan kepada pengguna.

Salah satu jenis metode untuk melakukan pengujian pada perangkat lunak yaitu *black box testing*.

Black box testing berfokus pada apakah unit program memenuhi kebutuhan (*requirement*) yang disebutkan dalam spesifikasi. Pada *black box testing*, cara pengujian dilakukan dengan menjalankan atau mengeksekusi unit atau modul, kemudian diamati apakah hasil dari unit ini sesuai dengan proses yang diinginkan.

Teknik yang digunakan dalam *black box testing* antara lain:

a. Digunakan untuk menguji fungsi-fungsi khusus dari perangkat lunak.
b. Kebenaran perangkat lunak yang diuji hanya dilihat berdasarkan keluaran (*output*) yang dihasilkan.
c. Kemampuan program dalam memenuhi kebutuhan pemakai dapat diukur sekaligus dapat diketahui kesalahan-kesalahannya.