BAB II
LANDASAN TEORI

2.1 Kajian Pustaka

Anwar (2015) membuat aplikasi yang memberikan informasi wisata beserta penjelasan singkat dan menampilkan peta rute terdekat menuju lokasi wisata yang akan dituju melalui Google Maps APIs dalam membantu wisatawan menemukan lokasi wisata di kota Semarang.

Model analisisnya menggunakan *Use Case, Activity, Class Diagram*. Sistem dirancang menggunakan pemodelan UML dan metodologi SDLC. Sistem ini di implementasikan menggunakan *script* PHP serta java pada aplikasi android, untuk *user* dibangun menggunakan *eclipse Juno, Java* perangkat lunak membuktikan sistem ini mampu untuk memberikan informasi terkait suatu objek wisata serta beberapa fitur yang memanfaatkan *location based service.*

Penelitian Fadhillah (2013) bertujuan untuk menampilkan informasi pariwisata berupa lokasi serta menampilkan daftar objek wisata lengkap dengan estimasi waktu tempuh ke lokasi wisata. Sistem juga dapat menampilkan informasi yang berkaitan dengan pariwisata antara lain informasi restoran, hotel, ATM dan SPBU. Untuk menyimpan data wisata, sistem ini menggunakan bahasa pemrograman PHP dan MySQL.

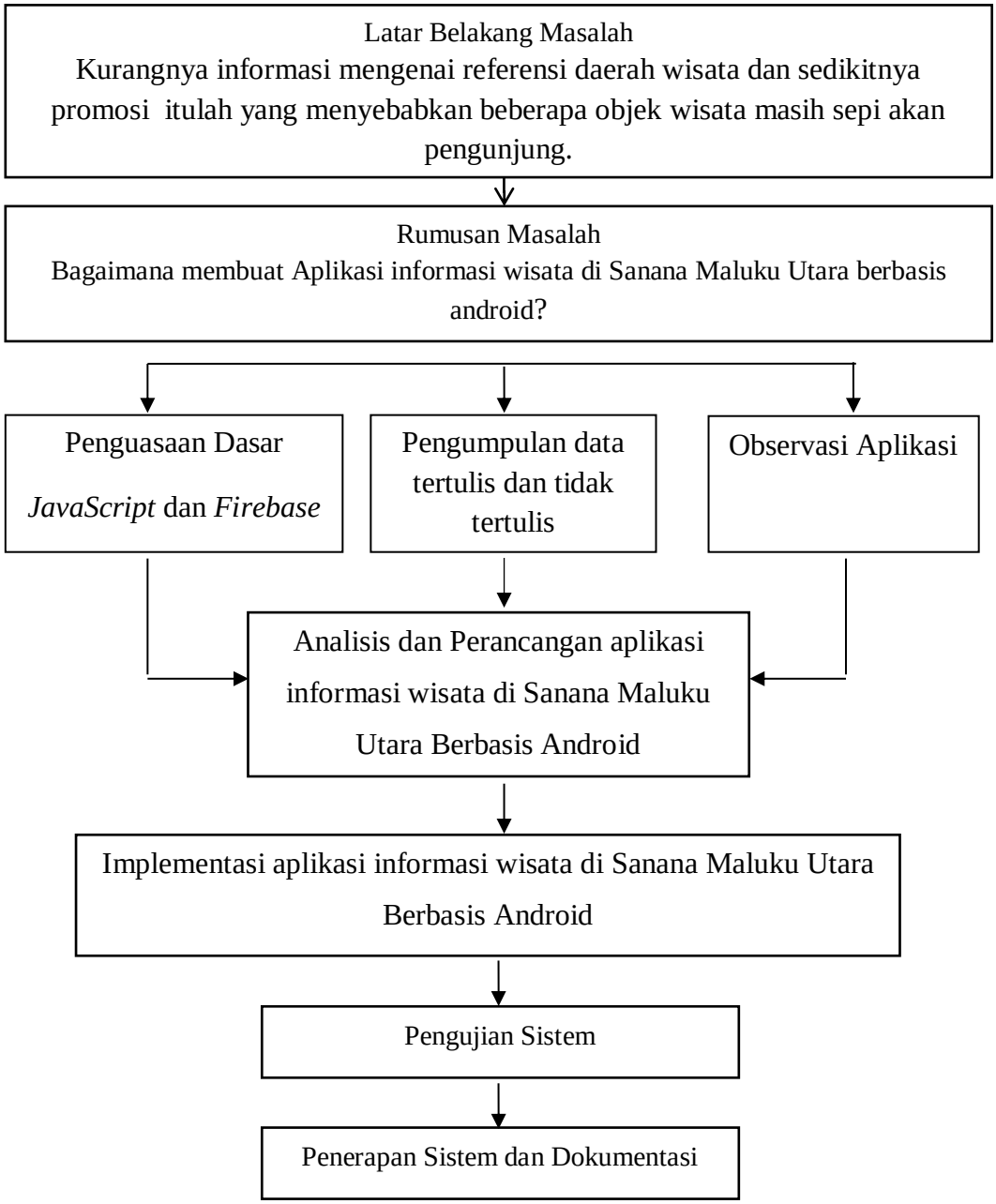
Penelitian Khasanah (2018) membuat aplikasi pencarian wisata di Pemalang dan untuk memudahkan para wisatawan menuju ke tempat wisata yang diinginkan. Pembuatan aplikasi ini merupakan solusi yang terbaik untuk memecahkan permasalahan-permasalahan yang di hadapi oleh para wisatawan, dengan aplikasi ini para wisatawan dapat memilih dan mengetahui lokasi atau rute tempat wisata yang akan dikunjungi. Aplikasi ini bisa dijalankan di *smartphone* android dengan minimal OS android 4.2.2 *(Jellybean)* dan maksimal OS android 5.1 *(Lolipop).*

Penelitian Iqbal Fauzi (2012) membuat perancangan sistem informasi objek wisata secara realtime berbasis mobile android, aplikasi lokasi wisata kuningan berbasiskan platform Android merupakan aplikasi pemetaan yang berguna untuk

menunjukkan lokasi objek wisata dan kuliner yang berada di Kabupaten Kuningan, pembuatan aplikasi menggunakan *eclipse.*

2.2 Kerangka Pemikiran

Tahapan kerangka pemikiran dalam pembuatan aplikasi informasi wisata di Sanana Maluku Utara disajikan pada Gambar 2.1



Gambar 2.1 Kerangka Pemikiran

Penjelasan dari kerangka pemikiran tersebut adalah :

1. Latar Belakang Masalah

   Kurangnya informasi mengenai referensi daerah wisata dan sedikitnya promosi itulah yang menyebabkan beberapa objek wisata masih sepi akan pengunjung.

2. Rumusan Masalah

   Bagaimana membuat Aplikasi informasi wisata di Sanana Maluku Utara?

3. Penguasaan *Javascript* dan *Firebase*

   Tahap untuk mempelajari dasar-dasar *Javascript* dan *Firebase* agar lebih menguasai program-program yang akan digunakan untuk membangun aplikasi.

4. Pengumpulan data

   Tahap pengumpulan data pada penelitian ini melalui observasi, wawancara dan studi literatur. Pengumpulan data bertujuan untuk mengetahui permasalahan dan kebutuhan informasi mengenai aplikasi informasi wisata di Sanana Maluku Utara berbasis android.

5. Observasi Aplikasi informasi wisata di Sanana Maluku Utara Berbasis Android

   Merupakan tahap pengamatan contoh aplikasi yang telah ada, jurnal, buku, maupun karya ilmiah untuk kajian yang dapat dijadikan referensi untuk pembangunan aplikasi.

6. Analisis dan Perancangan aplikasi informasi wisata di Sanana Maluku Utara berbasis android.

7. Implementasi aplikasi informasi wisata di Sanana Maluku Utara berbasis android.

   Implementasi sistem adalah langkah-langkah atau prosedur-prosedur yang dilakukan dalam menyelesaikan desain aplikasi yang telah disetujui, untuk menginstal, menguji dan memulai sistem baru atau sistem yang diperbaiki. Lingkungan implementasi aplikasi ini meliputi kebutuhan perangkat lunak, perangkat keras, *form* program yang sesuai, *script* yang digunakan, pemrograman, pengujian program dan pengujian aplikasi yang telah

dirancang sesuai dengan kebutuhan dalam pembuatan aplikasi informasi wisata di Sanana Maluku Utara berbasis android ini.

8. Pengujian Sistem

   Pengujian sistem adalah prosedur yang dilakukan untuk menyelesaikan rancangan sistem yang telah disetujui, menguji sistem, menginstal serta memulai penggunaan sistem baru atau sistem yang telah diperbaiki. Dalam pengujian sistem ini menggunakan metode *Five view.*

2.3 Landasan Teori

2.3.1 Pengertian Pariwisata

Secara etimologi, kata pariwisata berasal dari bahasa Sansekerta yang terdiri dari dua suku kata, yaitu pari‟ dan wisata‟. Pari yang berarti berputar atau keliling, sedangkan wisata adalah perjalanan. Pariwisata adalah istilah yang diberikan apabila seorang wisatawan melakukan perjalanan itu sendiri atau dengan kata lain aktivitas dan kejadian yang terjadi ketika seorang pengunjung melakukan perjalanan (Sutrisno, 1998).

Pariwisata merupakan kegiatan yang memanfaatkan kekayaan alam dan lingkungan hidup yang khas, seperti hasil budaya, peninggalan sejarah, pemandangan alam yang indah dan iklim yang nyaman.

2.3.2 Pengertian Android

Android adalah sebuah *platform* untuk sebuah perangkat *mobile* berbasis *linux* yang mencakup sistem operasi, *middle ware* dan aplikasi. Android menyediakan *platform* tebuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya *google inc*. membeli *android inc.*yang merupakan pendatang baru yang menciptakan piranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan android, dibentuklah *open handset alliance,* konsorsium dari 34 perusahaan piranti keras, piranti lunak dan telekomunikasi, termasuk *google, HTC, intel, motorola, qualcomm, t-mobile,* dan nvidia. android bersama Open Handset Allliance menyatakan mendukung pengembangan *open source* pada perangkat *mobile*. Di lain pihak, *google* merilis kode-kode Android di bawah

lisensi *apache*, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler. (Safaat, 2015).

2.3.3 Android Studio

Android Studio adalah *Integrated Development Environment* (IDE) untuk mengembangkan aplikasi *Android. Android Studio* berbasis pada *"IntelliJ IDEA"* Java-IDE dari Jetbrains dan diperkenalkan oleh *Google*. *Android Studio* ini diumumkan pada Mei 2013, (Hohensee, 2014)

Android Studio direncanakan akan menggantikan *Eclipse* sebagai IDE resmi untuk mengembangkan aplikasi pada *platform Android. Android Studio* memiliki beberapa fitur baru dibandingkan dengan *Eclipse,* diantaranya adalah

a. Menggunakan *Gradle-based build* sistem yang fleksibel.
b. Bisa melakukan *build* pada beberapa APK
c. *Layout editor yang* lebih bagus.
d. *Built-insupport* untuk Google Cloud Platform, sehingga mudah untuk integrasi dengan *Google Cloud Messaging dan App Engine.*
e. *Import library* langsung dari *Maven repository*

2.3.4 Firebase

Firebase adalah penyedia layanan *cloud* dengan *back-end* sebagai servis yang berbasis di *San Fransisco, California. Firebase* membuat sejumlah produk untuk pengembangan aplikasi *mobile* ataupun *web*. *Firebase* di dirikan oleh Andrew Lee dan James Tamplin pada tahun 2011 dan diluncurkan dengan *cloud database* secara realtime di tahun 2012 (Evangelist, 2015).

Produk utama dari Firebase yakni suatu *database* yang menyediakan API untuk memungkinkan pengembang menyimpan dan mensinkronisasi data lewat *multiple client*. Perusahaan ini diakusisi oleh *Google* pada Oktober 2014.

Firebase memiliki banyak *library* yang memungkinkan untuk mengintegrasikan layanan ini dengan Android, iOS, *Javascript*, *Java*, *Objective-C* dan *Node.JS. Database Firebase* juga bersifat bisa diakses lewat REST API dan data binding untuk beberapa *framework Java script* seperti halnya Angular JS, React JS, Ember.JS, dan Back bone.JS. REST API tersebut menggunakan protokol *Server-Sentn Event* dengan membuat koneksi HTTP untuk menerima

push *notification* dari server. Pengembang juga bisa menggunakan *database* ini untuk mengamankan data mereka menggunakan server *firebase* dengan *rules* yang ada.

2.3.5 Android SDK

Menurut Safaat (2011), Android SDK adalah *tools* API *(Application Programming Interface)* yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan, namun bukan merupakan aplikasi bawaan *Handphone/Smartphone*. Beberapa fitur-fitur Android yang paling penting adalah:

a. *Framework* aplikasi yang mendukung penggantian komponen dan *reusable.*
b. Mesin *Virtual Dalvik* dioptimalkan untuk perangkat mobile.
c. *Integrated browser* berdasarkan *engine open source WebKit*.
d. Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, grafis 3D berdasarka spesifikasi opengl ES 1,0 (Opsional akselerasi *hardware*).
e. *SQLite* untuk penyimpanan data *(database)*.
f. Media *Support* yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM *Telephony* (tergantung *hardware*).
g. Bluetooth, EDGE, 3G, dan WiFi (tergantung *hardware*)
h. Kamera, GPS, kompas, dan accelerometer (tergantung *hardware*).
i. Lingkungan *development* yang lengkap dab kaya termasuk perangkat *emulator, tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE *Eclipse*

2.3.6 JDK (*Java Development Kit*)

Menurut Jubilee (2015), JDK adalah singkatan dari *Java Development Kit* yaitu *software* yang digunakan untuk membangun aplikasi-aplikasi java. Tanpa JDK kita tidak akan bisa membangun atau membuat berbagai macam aplikasi java. JDK berisi sekumpulan *command line tool* untuk menciptakan program java.

JDK wajib terinstall pada komputer yang akan melakukan proses pembuatan aplikasi berbasis java. Berikut adalah beberapa komponen utama JDK :

1.Kompulator (javac)2. *Interpreter* program java (java)

3. *Applet viewer (appletviewer)*

4. *Debugger* (jdb)

5. *Class file disassembler (javap)*

6. *Java Archive (jar)*

7. *Documentation generator (javadoc)*

8. *Applet* demo

9. Kode sumber Java API

JDK berisi *Java Runtime Environment* (JRE) dan semua alat yang diperlukan untuk mengkompilasi aplikasi Java. Hal ini diperlukan untuk mengkompilasi dan menjalankan Java, yang diperlukan untuk menjalankan android.

2.3.7 Node JS

Menurut Iqbal dkk. (2012) *Node JS* adalah sistem perangkat lunak yang didesain untuk pengembangan aplikasi web. Aplikasi ini ditulis dalam bahasa *javascript*, menggunakan basis *event* dan *asynchrounous I/O*. Tidak seperti kebanyakan bahasa *javascript* yang dijalankan pada peramban, *node.js* dieksekusi sebagai aplikasi *server*. Aplikasi ini terdiri dari V8 *javascript engine* buatan *google* dan beberapa modul bawaan yang terintegrasi. *node.js* sebagian besar diimplementasikan dalam C dan C++, berfokus pada kinerja dan penggunaan memori yang rendah. *node.js* bertujuan untuk menyokong lama berjalanya proses *server*. Proses node tidak bergantung pada *multithreadding* untuk mendukung pelaksaan *business logic*, hal ini berdasar pada model *event asynchronous I/O*.

2.3.8 Android Virtual Devices

Android Virtual Device merupakan emulator untuk menjalankan program aplikasi Android yang kita buat. AVD ini selanjutnya digunakan sebagai tempat untuk test dan menjalankan aplikasi Android tanpa harus menggunakan perangkat Android yang sebenarnya. Sebelum menggunakan AVD harus menentukan karakteristiknya, misalkan dalam menentukan versi Android, jenis dan ukuran layar dan besarnya memori (Maiyana, 2018).

2.3.9 Pengertian OOP (**Objek Oriented Program**)

Sukamto dan Shalahuddin (2014) mendefinisikan bahwa, *Object Oriented Programming* (OOP) adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Berdasarkan pengertian yang ada dapat di simpulkan bahwa *Object Oriented Programing* (OOP) merupakan suatu strategi atau cara baru untuk membuat program atau merancang sistem dengan memperhatikan objek .

2.3.10 Pengertian UML

Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement,* membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. pada UML terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Berikut ini penjelasan singkat dari pembagian kategori tersebut (Sukamto dan Shalahudin, 2014).

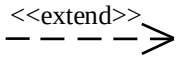
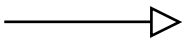
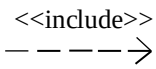
1. *Structure diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. *Structure* diagram terdiri dari *class diagram*, *object diagram*, *component diagram*, *composite structure diagram*, *package diagram* dan *deployment diagram*.
2. *Behavior diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. *Behavior diagram* terdiri dari *use case diagram*, *activity diagram*, *state machine system*.
3. *Interaction diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar sub sistem pada suatu sistem. *Interaction diagram* terdiri dari *sequence diagram communication diagram, timing diagram*, *interaction overview diagram*.

2.3.11 Pengertian *Use Case Diagram*

Menurut Sukamto dan Shalahudin (2014), *use case* atau *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. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Simbol *use case diagram* disajikan pada Tabel 2.1.

Tabel 2.1. Simbol *Use Case Diagram* (Sukamto dan Shalahuddin, 2014).

NO	SIMBOL	NAMA	KETERANGAN
1		*Use case*	Fungsional yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan kata kerja di awal di awal frase nama *use case*.
2		*Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanyanya dinyatakan menggunakan kata benda di awal frase nama aktor.
3		*Association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4		*Extend*	Relasi *use case* tambahan kesebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
5		*Generalization*	Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6		*Include*	Relasi *use case* tambahan ke sebuah *use case* di mana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

2.3.12 Pengertian *Class Diagram*

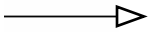
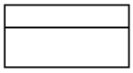
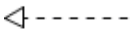
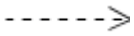
Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun system, (Sukamto dan Shalahudin, 2014).

Kelas memiliki apa yang disebut atribut dan *method* atau operasi. Berikut penjelasan atribut dan *method*:

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau method adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Simbol *class diagram* disajikan pada Tabel 2.2.

Tabel 2.2. Simbol *Class Diagram* (Sukamto dan Shalahuddin, 2014).

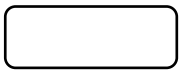
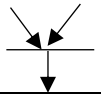
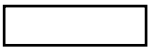
NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*Nary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

2.3.13 Pengertian *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *work flow* (aliran kerja) atau aktivitas dari sebuah sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak, (Sukamto dan Shalahudin, 2014)

Yang perlu di perhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Simbol *activity diagram* disajikan pada Tabel 2.3.

Tabel 2.3 Simbol *Activity Diagram* (Sukamto dan Shalahuddin, 2014).

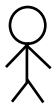
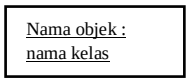
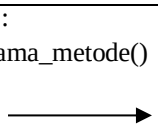
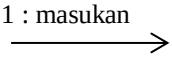
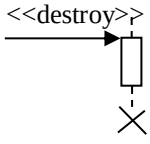
No	Gambar	Nama	Keterangan
1		*Start Point*	Diletakkan pada objek kiri atas dan merupakan awal aktivitas
2		*End Point*	Akhir aktivitas
3		*Activities*	Menggambarkan suatu proses atau kegiatan bisnis
4		*Fork*	Digunakan untuk menunjukkan kegiatan yang dilakukan secara pararel atau digunakan untuk menggabungkan dua kegiatan menjadi satu
5		*Join*	Digunakan untuk menunjukkan dekomposisi
6		*Decision Points*	Menggambarkan pilihan untuk pengambilan keputusan true atau false
7		*Swimlane*	Pembagian activity diagram dalam sebuah urutan yang sama

2.3.14 Pengertian *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dengan *message* yang dikirimkan dan diterima antar objek, (Sukamto dan Shalahudin, 2014).

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 diinstansiasi menjadi objek itu. *Sequence diagram* mempunyai simbol-simbol yang ditunjukkan pada Tabel 2.4.

Tabel 2.4. Simbol *Sequence Diagram* (Sukamto dan Shalahuddin, 2014).

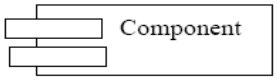
NO	SIMBOL	NAMA	KETERANGAN
1		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang;
2		*Lifeline*	Menyatakan kehidupan suatu objek.
3		Objek	Menyatakan objek yang berinteraksi pesan.
4		Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi.
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 pada objek lain atau dirinya sendiri.
7		Pesan tipe send	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.
8		Pesan tipe *destroy*	Menyatakan suatu objek mengakhiri hidup objek lain arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*.

2.3.15 Pengertian *Component Diagram*

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada didalam system, (Sukamto dan Shalahudin, 2014).

Hubungan antara *component* dan *class*, *Component* adalah implementasi *software* dan sebuah *class*. *Class* mewakili abstraksi dari serangkaian *attribute* dan *operation*. Hal terpenting yang perlu diingat tentang *class* dan *component* adalah sebuah *component* bisa jadi merupakan implementasi dari lebih dari sebuah *class*. *Component diagram* mempunyai simbol-simbol yang ditunjukkan pada Tabel 2.5.

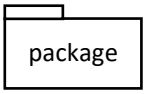
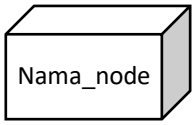
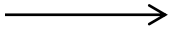
Tabel 2.5. Simbol *Component Diagram* (Sukamto dan Shalahudin, 2014).

SIMBOL	NAMA	KETERANGAN
	Komponen	Sebuah komponen melambangkan sebuah entitas software dalam sebuah sistem. Sebuah komponen dinotasikan sebagai sebuah kotak segiempat dengan dua kotak kecil tambahan yang menempel disebelah kirinya.
	Depedency	Sebuah Dependency digunakan untuk menotasikan relasi antara dua komponen. Notasinya adalah tanda panah putus-putus yang diarahkan kepada komponen tempat sebuah komponen itu bergantung.

2.3.16 Pengertian *Deployment Diagram*

Menurut (Sukamto dan Shalahuddin, 2014) *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram *deployment* juga dapat digunakan untuk memodelkan sistem tambahan yang menggambarkan rancangan *device*, *node*, dan *hardware*. Sistem *client*/*server*, sistem terdistribusi murni, rekayasa ulang aplikasi.Berikut merupakan tabel mengenai simbol *deployment diagram* yang ditunjukkan pada Tabel 2.6.

Tabel 2.6 Simbol-simbol *Deployment*

NO	SIMBOL	NAMA	KETERANGAN
1.		*Package*	*Package* merupakan sebuah bungkusan dari satu atau lebih *node.*
2.		*Node*	Biasanya mengacu pada perangkat keras (*hardware*), perangkat lunak yang tidak dibuat sendiri (*software*), jika didalam *node* disertakan komponen untuk mengkonsistenkan rancangan maka komponen yang diikutsertakan harus sesuai dengan komponen yang telah didefinisikan sebelum pada diagram komponen.
3.		*Dependency*	Kebergantungan antar *node*, arah panah mengarah pada *node* yang dipakai.
4.		*Link*	Relasi antar *node.*

2.3.17 Metode *Five View*

Pengujian perangkat lunak ini menggunakan metode pengujian *Five View*, pengujian *Five View* berfokus pada penilaian deskriptif.

Menurut (Tripathy dan Naik, 2011) Pengujian *Five View* adalah pengujian yang sifatnya deskriptif dimana *software* yang diuji dinilai melalui lima sudut pandang atau kategori yang berbeda melalui penilaian dari *expertise* dari masing-masing sudut pandang. Lima sudut pandang tersebut adalah sebagai berikut:

1) User View

Kualitas menyangkut sejauh mana produk memenuhi kebutuhan dan harapan pengguna dan apakah suatu produk cocok untuk digunakan. Pendapat ini bersifat sangat personal. Sebuah produk berkualitas baik jika memuaskan sebagian besar pengguna, Hal ini berguna untuk mengidentifikasi fitur dari produk yang pengguna anggap penting. Pandangan ini dapat mencangkup banyak unsur subjek, seperti kegunaan, kendala, dan ke efisiensasian.

2) Manufacturing View

Pandangan ini berkaitan dengan faktor dalam industri manufaktor, apakah produk memenuhi persyaratan atau tidak setiap penyimpanan dari persyaratan yang dinilai mengurangi kualitas produk. Konsep proses memainkan peran kunci. Produk yang dibuat harus orisinil sehingga biaya berkurang, misal biaya pembangunan dan biaya pemeliharaan.

Kesesuaian dengan persyaratan dan spesifikasi menyebabkan keraguan dalam produk tapi beberapa berpendapat bahwa keseragaman tersebut tidak menjamin kualitas. Kualitas produk dapat secara bertahap ditingkatkan dengan memperbaiki proses.

3) Transcendental View

Kualitas menurut pandangan ini adalah sesuatu yang dapat dikenali melalui pengalaman tapi tidak dapat selalu digambarkan. Objek atau *software* yang bagus itu menonjol dan dapat dengan mudah dikenali.

4) Value-based View

Value-based View merupakan penggabungan dari dua konsep yaitu keunggulan dan kelayakan. Kualitas adalah ukuran dari keunggulan dan nilai adalah ukuran layak. Beberapa banyak pengguna bersedia membayar untuk tingkat kualitas tertentu. Kualitas tidak berarti jika produk memenuhi nilai ekonomi, Pandangan berbasis nilai antara biaya dan kualitas.

5) Product View

Jika sebuah produk diproduksi dengan sifat internal (misalnya bahan dan tindakan) yang baik, maka produk akan memiliki sifat eksternal atau *output* yang baik dan dapat dieksplorasi hubungan antara sifat internal dan kualitas eksternal.