BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Ada beberapa tinjauan pustaka sebagai dasar dari penelitian ini. Tinjauan pustaka yang digunakan meliputi penelitian dari Firdhaus, dkk. (2017), Christianti, dkk. (2015), Isnawati, dkk. (2016), Ida, dkk. (2018).

Penelitan pertama dari Firdaus, dkk. (2017) dengan judul Sistem Informasi Penggunaan Ruang Kuliah Pada Universitas Wahid Hasyim Berbasis Web membahas tentang kurang efektifnya pencarian ruang oleh mahasiswa yang tentunya akan memakan waktu yang lama sehingga dibuatlah sistem informasi penggunaan ruang sehingga jadwal perkuliahan dapat tersusun secara akurat. Penelitian ini menghasilkan sebuah sistem informasi berupa fasilitas pencarian ruangan dengan menggunakan website. Dengan adanya sistem informasi ruangan kampus ini dapat mempermudah mahasiswa atau dosen dalam menemukan letak suatu ruangan dan juga mencari ruang kelas yang sedang kosong atau tidak terpakai dengan cepat di Universitas Wahid Hasyim. Sistem ini diharapkan bisa membantu meningkatkan pelayanan terhadap dosen dan mahasiswa di Universitas Wahid Hasyim menjadi lebih baik lagi. Rancangan penelitian ini memiliki tampilan menu yaitu menu utama, daftar ruang dan menu cari jadwal.

Penelitian kedua dari Christianti, dkk. (2015) dengan judul Analisis Dan Perancangan Aplikasi Penyusunan Jadwal Mengajar Sesuai Data Kesediaan Mengajar Dosen yang membahas tentang proses penyusunan jadwal mengajar dosen masih dilakukan secara manual. Hal ini mengakibatkan proses penyusunan jadwal memakan waktu cukup lama, karena sekretaris Jurusan Teknik Informatika perlu menanyakan kesediaan mengajar dosen dan waktu yang disediakan, membandingkan jadwal mengajar setiap dosen sehingga tidak terjadi bentrokan dengan jadwal mengajar dosen lain dan juga kebentrokan penggunaan ruangan.

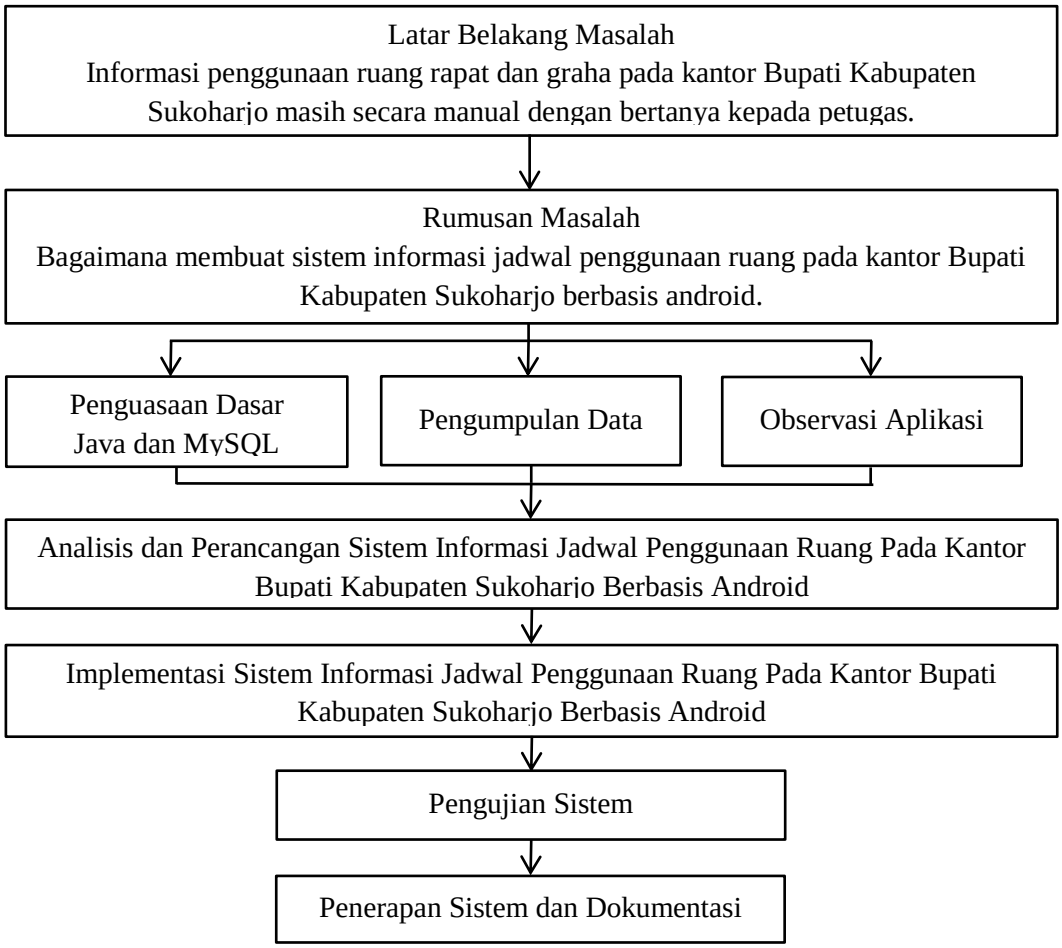
Rancangan penelitian ini memiliki tampilan menu berupa tampilan data utama fakultas, data jurusan, data mata kuliah, data proyek pendidikan, data dosen, data ruangan, dan data utama menu slot.

Penelitian ketiga dari Isnawaty, dkk. (2016) dengan judul Rancang Bangun Aplikasi Penjadwalan Ruang *Meeting* Hotel Menggunakan *Algoritma Multiple Feedback Queue* (MFQ) Berbasis Android Menggunakan Layanan SMS "Studi Kasus Hotel Plaza Inn Kendari" membahas tentang sistem pemesanan jadwal ruang meeting yang masih manual. Hal ini menyebabkan jadwal meeting yang bertabrakan sehingga petugas hotel harus memeriksa semua jadwal ruangan apabila ada pemesanan ruangan meeting yang baru ataupun adanya perubahan jadwal meeting yang sudah ditambahkan sebelumnya, hal ini menjadi tidak efisien dari segi waktu. Rancangan penelitian ini memiliki tampilan menu berupa menu cek ketersediaan ruangan dan jadwal, menu ketersediaan jadwal, menu menambahkan pemesanan ruangan dan menu rincian pemesanan.

Penelitian keempat dari Ida, dkk. (2018) yang membahas tentang *Webqual* 4.0 yang digunakan dalam penelitian ini sebenarnya merupakan kerangka untuk mengukur kualitas website. Dalam penelitian ini, Webqual 4.0 diadopsi dan kemudian disesuaikan untuk digunakan sebagai kerangka untuk mengukur faktor kualitas aplikasi berbasis Android. Hasil penelitian menunjukkan bahwa Webqual 4.0 masih cukup valid dan reliabel untuk mengukur kualitas layanan teknologi informasi dan komunikasi selain website. Hal ini dikarenakan adanya persamaan karakteristik antara website dan aplikasi mobile berbasis Android, yaitu sebagai jendela penghubung antara pengguna dengan organisasi.

2.2 Kerangka Pemikiran

Berikut ini adalah tahapan kerangka pemikiran dalam membuat sistem informasi jadwal penggunaan ruang pada Kantor Bupati Kabupten Sukoharjo Berbasis *Android* dapat dilihat pada Gambar 2.1 berikut:



Gambar 2.1 Kerangka Pemikiran

Penjelasan dari kerangka pemikiran tersebut adalah :

1. Latar Belakang Masalah

Pemberitahuan informasi mengenai agenda di kantor Bupati Kabupaten Sukoharjo seperti waktu pelaksaan kegiatan serta ruang yang akan digunakan yaitu ruang rapat atau ruang graha saat ini masih manual, sehingga apabila kesulitan mendapatkan suatu informasi menyebabkan mekanisme pencarian ruang butuh waktu yang cukup lama.

2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan maka rumusan masalahnya adalah bagaimana membuat sistem informasi jadwal penggunaan ruang pada kantor Bupati Kabupaten Sukoharjo berbasis *android*.

3. Penguasaan dasar *Java* dan MySQL

Penguasaan dasar *java* dan MySQL merupakan suatu kunci terciptanya sistem informasi jadwal penggunaan ruang pada kantor Bupati Kabupaten Sukoharjo berbasis *android* tersebut, karena pembuatan sistem ini menggunakan bahasa pemrograman *java* dan menggunakan database MySQL sebagai media penyimpan datanya.

4. Pengumpulan data

Tahap pengumpulan data pada penelitian ini melalui observasi, wawancara dan studi literatur. Pengumpulan data bertujuan untuk mengetahui permasalahan dan kebutuhan suatu instansi.

5. Observasi aplikasi

Observasi aplikasi bertujuan untuk mencari gambaran sistem yang akan dibuat dengan melihat beberapa aplikasi yang sudah ada.

6. Analisis dan perancangan sistem

Analisis dan perancangan sistem informasi jadwal penggunaan ruang pada kantor Bupati Kabupaten Sukoharjo bertujuan untuk mengetahui kelemahan serta kekurangan dari sistem yang telah berjalan saat ini, serta membuat perancangan sistem baru yang diharapkan dapat memberikan suatu informasi dengan efektif. Perancangan sistem pada penelitian ini menggunakan UML (*Unified Modeling Language*).

7. Implementasi Sistem

Membuat sistem informasi jadwal penggunaan ruang pada kantor Bupati Kabupaten Sukoharjo menggunakan bahasa pemrograman *java* dan *database* MySQL sebagai media penyimpanan datanya.

8. Pengujian Sistem

Pengujian sistem merupakan tahap setelah sistem berhasil dibuat. Pengujian sistem bertujuan untuk mengetahui kelebihan, kelemahan serta mengetahui kelayakan suatu sistem untuk digunakan. Pengujian sistem pada penelitian ini menggunakan metode *WebQual*.

9. Penerapan Sistem dan Dokumentasi

Tahap penerapan sistem adalah sistem yang telah dibuat telah lolos uji aplikasi dan telah digunakan pada DISKOMINFO Sukoharjo untuk membantu membagikan suatu informasi agar efektif, serta membuat dokumentasi sistem informasi jadwal penggunaan ruang di kantor Bupati Kabupaten Sukoharjo.

2.3 Landasan Teori

2.3.1 Sistem

Menurut Abdul (2014), sistem adalah sekumpulan elemen yang saling terkait atau terpadu yang dimaksudkan untuk mencapai suatu tujuan. Tujuan utama sistem yang umum ada tiga macam yaitu:

1. Mendukung fungsi kepengurusan manajemen.
2. Mendukung pengambilan keputusan manajemen.
3. Mendukung kegiatan operasi perusahaan.

2.3.2 Android

Menurut Akhmad (2015), a*ndroid* merupakan sebuah sistem operasi telepon seluler dan komputer tablet layar sentuh (*touchscreen*) yang berbasis Linux. Namun seiring perkembangannya *Android* berubah menjadi platform yang begitu cepat dalam melakukan inovasi. Hal ini tidak lepas dari pengembang utama dibelakangnya yaitu Google. Google-lah yang mengakusisi *Android*, kemudian membuatkan sebuah platform. Platform *Android* terdiri dari sistem operasi berbasis Linux, sebuah GUI (*Graphic User Interface*), sebuah web browser dan aplikasi *end-user* yang dapat di *download* dan juga para pengembang bisa dengan leluasa berkarkaya serta menciptakan aplikasi yang terbaik dan terbuka untuk digunakan oleh berbagai macam perangkat.

1. *Operating* sistem yang digunakan:

a. *KitKat* (*Android* versi 4.4)

Awalnya *Android* versi ini di isukan bernama *Key Lime Pie*. Namun pada tanggal Oktober 2013 Google merilis kitkat sebagai generasi *android* berikutnya. *Android* versi ini memiliki banyak fitur & semakin memanjakan

para pengguna *android*. Diantaranya: *Immersive mode,* Akses kontak langsung dari aplikasi telepon, *google now launcher*, dan pastinya memiliki *interface* UI yang baru.

b. *Lollipop* (*Android* versi 5.0)

Android 5.0 merupakan versi paling baru dari sistem operasi *Android* 5.0 sendiri dianggap membawa *update* yang fantastis, banyak perubahan yang disertakan Google di dalamnya.

2. Pemrograman yang digunakan:

a. *Android* SDK (*Software Development Kit*)

Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform *Android* menggunakan bahasa pemrograman java. *Android* merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang direlease oleh Google (Safaat, 2014).

b. AVD (*Android Virtual device*)

AVD yaitu semacam *emulator* untuk menjalankan *virtual*. Jadi tanpa menggunakan *phone*, seorang *progammer* bisa merasakan membuat program. Tetapi untuk yang ingin berkecimpung di dunia *coding* sebaiknya minimal punya *phone*, karena bisa melakukan tes aplikasi yang sedang dibuat langsung karena jika menggunakan AVD akan memakan memori ram komputer, sehingga akan berjalan berat.

c. JDK (*Java Development Kit*)

JDK merupakan sebuah produk yang dikembangkan oleh Oracle yang ditujukan untuk para *developer java*. JDK dilengkapi dengan banyak komponen untuk melakukan pemrograman. JDK juga berisi paket Java Runtime Environment (JRE) yang lengkap, biasanya disebut *private runtime* dari JRE reguler dan dilengkapi dengan konten tambahan, yaitu terdiri atas *Java Virtual Machine* dan semua *class library* yang ada di *environment* produk dan juga *library* tambahan yang berguna untuk *developer* (Susilo, 2015).

d. *Web Service*

Web service adalah sebuah *software* yang dirancang untuk mendukung *interoperabilitas* interaksi mesin ke mesin melalui sebuah jaringan. *Web service* secara teknis memiliki mekanisme interaksi antar sistem sebagai penunjang *interoperabilitas*. *Web service* memiliki layanan terbuka untuk kepentingan integrasi data dan kolaborasi informasi yang bisa diakses melalui internet oleh berbagai pihak menggunakan teknologi yang dimiliki oleh masing-masing pengguna. Data yang didapatkan dari *web service* dikirimkan dalam format standar misalnya XML atau JSON (*Javascript Object Notation*). Dalam penelitian ini dipilih JSON. Kelebihan utama JSON dibandingkan dengan XML adalah dari sisi ukuran *file*, JSON memiliki ukuran yang lebih kecil dibandingkan dengan XML. Ukuran *file* yang kecil penting untuk *web service* yang akan dibuat karena nantinya data akan diakses oleh aplikasi *mobile* yang membutuhkan respon cepat (Surendra, 2014).

e. JSON (*JavaScript Object Notation*)

JSON adalah format pertukaran data yang ringan, mudah dibaca dan ditulis, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman *JavaScript*. JSON adalah format teks yang tidak bergantung pada bahasa pemprograman apapun sehingga dapat digunakan di bahasa pemrograman C, C++, C#, Java, *JavaScript*, *Perl*, *Python*, dll. Oleh karena sifat-sifat tersebut, menjadikan JSON sangat ideal sebagai bahasa pertukaran data. JSON memiliki dua buah struktur yang diwakili dalam bentuk objek dan *array* (Rosid, 2016).

2.3.3 Android Studio

Menurut Juansyah (2015), Android studio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi Android dan bersifat *open source* atau gratis. Peluncuran Android Studio ini diumumkan oleh Google pada 16 mei 2013 pada *event* Google I/O *Conference* untuk tahun 2013. Sejak saat itu, Android Studio mengantikan Eclipse sebagai IDE resmi untuk mengembangkan aplikasi Android. Android studio sendiri dikembangkan

berdasarkan *IntelliJ* IDEA yang mirip dengan Eclipse disertai dengan ADT plugin (Android Development Tools). Android studio memiliki fitur:

a. Projek berbasis pada *Gradle Build*
b. *Refactory* dan pembenahan bug yang cepat
c. *Tools* baru yang bernama "Lint" dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.
d. Mendukung *Proguard And App-signing* untuk keamanan.
e. Memiliki GUI aplikasi android lebih mudah.
f. Didukung oleh Google *Cloud Platfrom* untuk setiap aplikasi yang dikembangkan.

2.3.4 Analisis Berorientasi Objek

Analisis berorientasi objek atau *Object Oriented Analysis* (OOA) adalah tahapan untuk menganalisis spesifikasi atau kebutuhan akan sistem yang akan dibangun dengan konsep berorientasi objek, apakah benar kebutuhan yang ada dapat diimplementasikan menjadi sebuah sistem berorientasi objek. Analisis ini sebaiknya dilakukan oleh orang-orang yang benar-benar memahami implementasi sistem yang berbasis atau berorientasi objek, karena tanpa pemahaman itu maka sistem yang dihasilkan bisa jadi tidak realistis jika diimplementasikan dengan berbasis objek. OOA biasanya menggunakan kartu CRC (*Component, Resposibility, Collaboration*) untuk membangun kelas-kelas yang akan digunakan atau menggunakan UML (*Unified Modeling Language*) pada bagian diagram *use case*, diagram kelas, dan diagram objek (Rosa & Shalahuddin, 2016).

2.3.5 Perancangan Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientasi objek

meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini sangat beragam (aplikasi bisnis, *real-time*, *utility* dan sebagainya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi.

Saat ini sudah banyak bahasa pemrograman berorientasi objek. Banyak orang berfikir bahwa pemrograman berorientasi objek identik dengan bahasa Java. Memang bahasa Java merupakan bahasa yang paling konsisten dalam mengimplementasikan paradigma pemrograman berorientasi objek. Bahasa pemrograman yang mendukung pemrograman berorientasi objek antara lain: bahasa pemrograman *Smalltalk*, *Eiffel*, C++, PHP dan Java (Rosa & Shalahuddin, 2016).

2.3.6 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQLAB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Relational Database Management System (RDBMS). MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structured Query Language*) (Ramadhani, Anis, & Masruro, 2013).

2.3.7 *Unified Modeling Language* (UML)

Unified Modeling Language adalah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan kebutuhan, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek. UML hanya berfungsi untuk melakukan pemodelan. Jadi, penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek (Rosa dan Shalahuddin, 2013).

Ada 2 tujuan penggunaan UML yaitu sebagai berikut:

1. Memodelkan suatu sistem (bukan hanya perangkat lunak) yang menggunakan konsep berorientasi objek.
2. Menciptakan suatu bahasa pemodelan yang dapat digunakan baik oleh manusia maupun mesin.

Ada 4 (empat) prinsip dasar dari pemrograman berorientasi obyek yang menjadi dasar kemunculan UML, yaitu abstraksi, enkapsulasi, modularitas dan hirarki (Haviluddin, 2011). Berikut dijelaskan satu persatu:

1. Abstraksi memfokuskan perhatian pada karakteristik obyek yang paling penting dan paling dominan yang bisa digunakan untuk membedakan obyek tersebut dari obyek lainnya.
2. Enkapsulasi menyembunyikan banyak hal yang terdapat dalam obyek yang tidak perlu diketahui oleh obyek lain. Dalam praktek pemrograman, enkapsulasi diwujudkan dengan membuat suatu kelas *interface* yang akan dipanggil oleh obyek lain, sementara didalam obyek yang dipanggil terdapat kelas lain yang mengimplementasikan apa yang terdapat dalam kelas *interface*.
3. Modularitas membagi sistem yang rumit menjadi bagian-bagian yang lebih kecil yang bisa mempermudah *developer* memahami dan mengelola obyek tersebut.

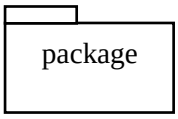
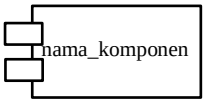
4. Hirarki berhubungan dengan abstraksi dan modularitas, yaitu pembagian berdasarkan urutan dan pengelompokkan tertentu. Misalnya untuk menentukan obyek mana yang berada pada kelompok yang sama, obyek mana yang merupakan komponen dari obyek yang memiliki hirarki lebih tinggi. Semakin rendah hirarki obyek berarti semakin jauh abstraksi.

Perancangan sistem dengan alat bantu perancangan *Unified Modeling Language* (UML) ada beberapa tahapan yang akan dilakukan, yaitu sebagai berikut:

1. *Component Diagram*

Diagram komponen *atau component* diagram dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. (Rosa & Shalahuddin, 2016) simbol *component* diagram dapat dilihat pada Tabel 2.1.

Tabel 2.1 Simbol *Component Diagram*.

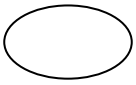
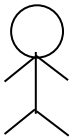
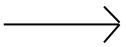
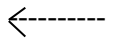
NO	GAMBAR	NAMA	KETERANGAN
1		*Package*	*Package* merupakan sebuah bungkusan dari satu atau lebih komponen
2		*Component*	Komponen sistem
3		*Dependency*	Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai
4		*Interface*	Sama dengan konsep *interface* pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen
5		*Link*	Relasi antar komponen

2. *Use Case Diagram*

Menurut Kusnita (2016), *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi

tersebut. Simbol-simbol yang digunakan dalam *use case* diagram seperti pada Tabel 2.2 berikut:

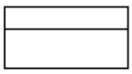
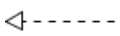
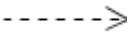
Tabel 2.2Simbol *Use Case*

No	Gambar	Nama	Keterangan
1.		*Use Case*	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja.
2.		*Actor*	*Actor* merupakan orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan *actor,* harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target system
3.		*Association*	*Association* antara *actor* dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data.
4.		*Generalization*	*Generalization* merupakan *association* antara *actor* dan *use case* yang menggunakan panah terbuka untuk mengidentifikasikan bila aktor berinteraksi secara pasif dengan sistem.
5.		*Include*	*Include* merupakan pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
6.		*Extend*	*Extend* merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

3. *Class Diagram*

Class Diagram merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), relasi *Assosiation*, attribut (*attributes*), operasi (*operation/method*) dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau attribut (Kusnita, 2016). Simbol-simbol yang digunakan dalam *class diagram* antara lain seperti pada Tabel 2.3 berikut:

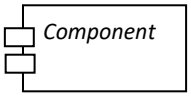
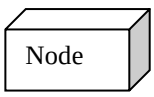
Tabel 2.3 Simbol *Class Diagram*

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 mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

4. *Deployment Diagram*

Menurut Kusnita (2016), *Deployment diagram* digunakan untuk menggambarkan detail bagaimana komponen disusun di infrastruktur sistem. Simbol-simbol yang digunakan dalam *deployment diagram* antara lain seperti pada Tabel 2.4 berikut:

Tabel 2.4 Simbol *Deployment Diagram*

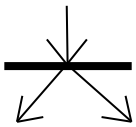
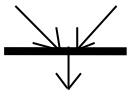
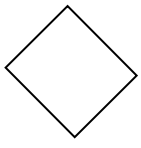
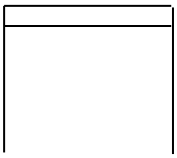
No	Gambar	Nama	Keterangan
1.		*Component*	Pada *deployment diagram*, komponen-komponen yang ada diletakkan didalam node untuk memastikan keberadaan posisi mereka.
2.		*Node*	*Node* menggambarkan bagian-bagian hardware dalam sebuah sistem. Notasi untuk node digambarkan sebagai sebuah kubus 3 dimensi.
3.		*Association*	Sebuah *association* digambarkan sebagai sebuah garis yang menghubungkan dua node yang mengindikasikan jalur komunikasi antara element-element hardware.

5. *Activity Diagram*

Menurut Kusnita (2016), *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis, diagram ini menunjukkan langkah – langkah dalam proses kerja sistem yang kita buat. Menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses serta memperlihatkan urutan aktifitas proses pada sistem.

Simbol-simbol *activity diagram* antara lain seperti pada Tabel 2.5 berikut:

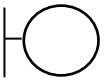
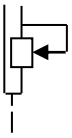
Tabel 2.5 Simbol *Activity Diagram*

No	Gambar	Nama	Keterangan
1.		*Start Point*	*Start Point* diletakkan pada pojok kiri atas dan merupakan awal aktivitas.
2.		*End Point*	*End Point* merupakan akhir aktivitas.
3.		*Activities*	*Activities* menggambarkan suatu proses/ kegiatan bisnis.
4.		*Fork*	*Fork*/ percabangan, digunakan untuk menunj ukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
5.		*Join*	*Join* / penggabungan atau rake digunakan untuk menunjukkan adanya dekomposisi
6.		*Decision Point*	*Decision Point* menggambarkan pilihan untuk pengambilan keputusan *true* atau *false.*
7.		*Swimlane*	*Swimlane* merupakan pembagian *activity diagram* untuk menunjukkan siapa dan melakukan apa.

6. *Sequence Diagram*

Menurut Kusnita (2016), *Sequence Diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* antara lain seperti pada Tabel 2.6 berikut:

Tabel 2.6 Simbol *Sequence Diagram*

No	Gambar	Nama	Keterangan
1.		*Entity Class*	*Entity Class* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
2.		*Boundary Class*	*Boundary Class* berisi kumpulan kelas yang menjadi *interfaces* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan form entry dan form cetak.
3.		*Control Class*	*Control class* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek
4.		*Message*	*Message*, merupakan simbol mengirim pesan antar *class.*
5.		*Recursive*	*Recursive* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
6.		*Activation*	*Activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi.
7.		*Lifeline*	*Lifeline* garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation.*

2.3.8 Pengujian WebQual

Menurut (Rosa & Shalahuddin, 2016) pengujian perangkat lunak adalah sebuah elemen topik yang memiliki cakupan luas dan sering dikaitkan dengan verifikasi (*verification*) dan validasi (*validation*). Verifikasi mengacu pada sekumpulan aktifitas yang menjamin bahwa perangkat lunak mengimplementasikan dengan benar sebuah fungsi yang spesifik. Validasi mengacu pada sekumpulan aktifitas yang berbeda yang menjamin bahwa perangkat lunak yang dibangun dapat ditelusuri sesuai dengan kebutuhan pelanggan (*customer*). Penelitian ini menggunakan model pengujian *WebQual*.

Menurut (Iman, 2012), *WebQual* merupakan salah satu metode atau teknik pengukuran kualitas *website* berdasarkan persepsi pengguna akhir. Metode ini merupakan pengembangan dari SERVQUAL yang banyak digunakan sebelumnya pada pengukuran kualitas jasa. *WebQual* sudah mulai dikembangkan sejak tahun 1998 dan telah mengalami beberapa interaksi dalam penyusunan dimensi dan butir pertanyaannya.

Menurut teori *WebQual* pada jurnal yang disusun oleh (Anwariningsih, 2011), terdapat tiga dimensi yang mewakili kualitas suatu *website*, yaitu kegunaan (*usability*), kualitas informasi (*information quality*) dan interaksi layanan (*service interaction*).

Beberapa tahapan yang harus dilakukan untuk melakukan pengujian *webqual* antara lain:

2.3.8.1 Uji Instrumen

Menurut (Prakosad, 2017), uji instrumen digunakan untuk mengetahui deskripsi mengenai variabel-variabel dalam penelitian, uji instrumen terdiri dari uji validitas dan uji reliabilitas.

2.3.8.2 Uji Validitas

Menurut (Arikunto, 2010) validitas adalah suatu ukuran yang menunjukkan tingkat kehandalan dan kesahihan suatu instrumen. Instrumen yang valid berarti alat ukur yang digunakan untuk mendapatkan data itu valid. Menurut (Sugiyono, 2010) valid berarti instrumen tersebut dapat digunakan untuk mengukur apa yang hendak diukur. Suatu kuesioner dinyatakan valid apabila r

hitung > r tabel, sebaliknya kuesioner dapat dinyatakan tidak valid apabila r hitung < r tabel. Pengujian validitas ini mengguakan batasan r tabel dengan signifikansi 0,05 dan uji 2 sisi.

2.3.8.3 Uji Reliabilitas

Menurut (Arikunto, 2010) reliabilitas adalah sesuatu instrumen yang dapat dipercaya untuk digunakan sebagai alat pengumpul data karena instrumen tersebut sudah baik. Instrumen yang reliabel atau dapat dipercaya akan menghasilkan data yang dapat dipercaya, instrumen yang reliabel mengandung arti bahwa instrumen tersebut harus baik sehingga mampu mengungkap data yang bisa dipercaya.

Berdasarkan (Iman, 2012) terdapat aturan praktis yang dapat diterapkan terkait dengan nilai alpha, jika alpha >0,9 berarti reliabilitas model sangat bagus, alpha >0,8 berarti reliabilitas model bagus, alpha > 0,7 artinya reliabilitas model bisa diterima, alpha >0,6 berarti reliabilitas model layak, alpha >0,5 berarti reliabilitas model kurang bagus, dan alpha <0,5 berarti reliabilitas model tidak dapat diterima.

2.3.8.4 Uji F (Uji Simultan)

Menurut (Ghozali, 2011), uji F digunakan untuk menunjukkan apakah semua variabel independent tang dimasukkan dalam model mempunyai pengaruh secara bersama-sama terhadap variabel dependent. Dasar penerimaan atau penolakan hipotesis dapat dilihat dengan membandingkan F hitug dengan F tabel, jika F hitung > F tabel maka Ho ditolak dan Ha diterima.

2.3.8.5 Uji T (Uji Parsial)

Menurut (Ghozali, 2011), uji T pada dasarnya menunjukkan seberapa jauh pengaruh satu variabel independent secara individual dalam menerangkan variabel dependent. Penerimaan atau penolakan hipotesis dilakukan dengan kriteria sebagai berikut:

1. Jika nilai *sig*<0,05 atau t hitung > t tabel maka terdapat pengaruh variabel X (variabel bebas) terhadap variabel Y (variabel terikat).
2. Jika nilai *sig*>0,05 atau t hitung < t tabel maka tidak terdapat pengaruh variabel X (variabel bebas) terhadap variabel Y (variabel terikat).

2.3.9 Pengujian *Black Box*

Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi *input* yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Menurut Rosa A.S dan M. Shalahuddin (2011), pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut:

a. Fungsi-fungsi yang tidak benar atau hilang.
b. Kesalahan interface.
c. Kesalahan dalam struktur data atau akses eksternal.
d. Kesalahan kinerja.
e. Inisialisasi dan kesalahan terminasi.