BAB II
TINJAUAN PUSTAKA

2.1 Studi Literatur

Pada bagian ini mengemukakan tentang referensi dari beberapa penelitian terkait dengan topik bahasan yang penulis angkat dalam penelitian ini. Penelitian yang digunakan sebagai bahan studi literatur dalam penelitian ini, antara lain :

1. Ariq (2017), melakukan penelitian yang berjudul "Aplikasi Antar Jemput Laundry Menggunakan *Google Maps API* Berbasis Android (Studi Kasus: Laundry CV. Ratu Sabrina)". Penelitian tersebut bertujuan untuk membangun sistem yang dapat membantu mengelola pelayanan laundry khususnya pada bagian antar jemput laundry pada CV. Ratu Sabrina dengan aplikasi tersebut pengantaran dapat dilakukan dengan lebih mudah dan efektif dengan memanfaatkan fitur *google Maps API* sebagai penanda lokasi tujuan pengantaran, Keterkaitan antara penelitian tersebut dengan penelitian ini adalah dalam pengambilan barang serta penggunaan *google Maps API* sebagai alat bantu untuk memetakan lokasi pengguna pada penelitian tersebut.
2. Suciyono dan Anwar (2019), melakukan penelitian dengan judul "Pencarian Wisata Kuliner Terdekat Dengan Menggunakan Metode *LBS* (*Location Based Service*) Berbasis Web Mobile Di Kota Tasikmalaya". Penelitian tersebut bertujuan untuk merancang aplikasi yang dapat memberikan solusi permasalahan pencarian lokasi kuliner di Tasikmalaya. Keterkaitan antara penelitian tersebut dengan penelitian ini adalah dalam penerapan metode *Location Based Service* (*LBS*) yang digunakan untuk pencarian lokasi.
3. Sri Sulistiowati (2020), melakukan penelitian dengan judul "Sistem Pemesanan Jasa Perbaikan Komputer Dengan *Location Based Services (LBS)* Berbasis *Android*", Penelitian tersebut bertujuan untuk membangun sistem pemesanan jasa teknisi secara online agar memudahkan pengguna jasa untuk mendapatkan teknisi lebih cepat sekaligus sebagai tempat bagi seseorang yang mempunyai kemampuan dalam memperbaiki komputer tetapi tidak memiliki toko. Dengan aplikasi ini masyarakat tidak perlu mendatangi tempat penyedia jasa perbaikan untuk memperbaiki komputer cukup menunggu di

rumahnya dan teknisi akan menuju lokasi tujuan. Keterkaitan antara penelitian tersebut dengan penelitian ini yaitu dalam hal penggunaan *Location Based Services*, dimana dalam hal ini pencarian lokasi teknisi akan ditentukan berdasarkan jarak terdekat antara customer dan teknisi yang sedang aktif.

2.2 Rongsokan

Menurut Kamus Besar Bahasa Indonesia (KBBI Online), "Rongsokan adalah suatu benda yang sudah rusak sama sekali (tentang barang)". Dapat diartikan bahwa rongsokan atau barang rongsok merupakan suatu benda yang daya gunanya sudah tidak maksimal lagi untuk digunakan atau dapat dikatakan barang yang sudah rusak, akan tetapi barang rongsok ini dapat di daur ulang kembali seperti kertas-kertas seperti koran, gelas plastik, botol plastik, botol kaca, besi, beling, kardus, aluminium, seng, tembaga, aki motor dan mobil. Barang rongsok yang sudah rusak ini dapat dijual kembali ke tukang rongsok dengan cara ditimbang atau dihitung berat barangnya.

2.3 Tukang Rongsok

Menurut Ali Lukman (1991), dapat disimpulkan bahwa tukang rongsok ataupun pemulung merupakan sebuah profesi yang dimana pekerjaannya berkeliling dari satu tempat ke tempat lain untuk mencari rongsokan atau barang-barang bekas yang sudah tidak dipakai dan membelinya. Sebagaimana yang kita ketahui rongsokan adalah barang-barang atau benda-benda yang sudah tidak terpakai atau barang yang sudah rusak. Dapat dikatakan bahwa tukang rongsok membeli barang yang daya gunanya tidak maksimal lagi atau tidak berfungsi, kotor, berdebu, atau mengalami kerusakan akan tetapi kebanyakan barang tersebut dapat di daur ulang kembali dengan cara diperbaiki ataupun diolah kembali. Setelah itu barang-barang bekas tersebut dijual kembali ke penampung barang bekas.

2.4 *Location Based Service*

Location Based Service (LBS) atau layanan berbasis lokasi adalah istilah umum yang digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang digunakan. Menurut Mahmoud (2004), *LBS* adalah sebuah layanan yang digunakan untuk mengetahui posisi dari pengguna,

kemudian menggunakan informasi tersebut untuk menyediakan jasa dan aplikasi yang personal. Dua unsur utama *LBS* adalah:

1. *Location Manager (API Maps)*

   Menyediakan tools/source untuk *LBS*, Application Programming Interface (API) Maps menyediakan fasilitas untuk menampilkan, memanipulasi peta beserta fitur-fitur lainnya seperti tampilan satelit, jalan, maupun gabungannya. Paket ini berada pada com.google.android.maps.

2. *Location Providers (API Location)*

   Pengguna dapat menentukan lokasinya, melacak gerakan/perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan. Safaat (2011).

Secara garis besar, *LBS* dapat dibagi menjadi dua, yaitu:

1. *Pull Service:* Layanan hanya diberikan ketika ada permintaan dari pengguna.
2. *Push Service:* Layanan diberikan langsung oleh service provider tanpa menunggu permintaan dari pengguna.

Dalam Layanan Berbasis Lokasi terdapat lima komponen penting yaitu meliputi:

1. *Mobile Devices*, merupakan suatu alat yang digunakan oleh pengguna untuk meminta informasi yang dibutuhkan. Informasi dapat diberikan dalam bentuk suara, gambar, dan text.
2. *Communication Network*, komponen ini mengirim data pengguna dan informasi yang diminta dari Mobile terminal ke Service Provider kemudian mengirimkan kembali informasi yang diminta ke pengguna. Communication network dapat berupa jaringan seluler *(GSM, CDMA), Wireless Local Area Network (WLAN),* atau *Wireless Wide Area Network (WWAN).*
3. *Positioning Component*, digunakan untuk memproses suatu layanan maka posisi pengguna harus diketahui.
4. *Service* dan *Aplication Provider*, penyedia layanan menawarkan berbagai macam layanan kepada pengguna dan bertanggung jawab untuk memproses informasi yang diminta oleh pengguna.

5. Data dan *Content Provider*, penyedia layanan tidak selalu menyimpan semua data yang dibutuhkan yang bisa diakses oleh pengguna. Untuk itu, data dapat diminta dari data dan *content provider*.

2.5 Aplikasi

Menurut Sri Sulistiowati (2005), Aplikasi adalah program yang memiliki aktivitas pemrosesan perintah yang diperlukan untuk melaksanakan permintaan pengguna dengan tujuan tertentu. Sedangkan menurut Pramana (2003), aplikasi adalah suatu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas seperti sistem perniagaan, game pelayanan masyarakat, periklanan, atau semua proses y (Rongsokan)ang hampir dilakukan manusia.

Berdasarkan dua pendapat ahli tersebut, maka dapat disimpulkan bahwa aplikasi merupakan sebuah program perangkat lunak yang dibuat dengan tujuan memenuhi kebutuhan dari aktivitas yang dilakukan oleh pengguna serta mempermudah pekerjaan dari pengguna tersebut. Pada penelitian ini Aplikasi akan menjadi alat bantu untuk menghubungkan antara tukang rongsok dan pemilik rongsok.

2.6 *Google Maps API*

Google Maps API adalah sebuah layanan (*service*) yang diberikan oleh *Google* kepada para pengguna untuk memanfaatkan *Google Map* dalam mengembangkan aplikasi. *Google Maps API* menyediakan beberapa fitur untuk memanipulasi peta, dan menambah konten melalui berbagai jenis *services* yang dimiliki, serta mengizinkan kepada pengguna untuk membangun aplikasi *enterprise* di dalam websitenya, *Google Maps API* adalah suatu *library* yang berbentuk *JavaScript* (Kindarto, 2008).

Google Maps API akan penulis gunakan untuk menampilkan peta pada aplikasi yang penulis buat. Adapun *Direction API* salah satu *service* dari Google yang akan penulis gunakan untuk memperoleh data jarak secara dinamis yang terdapat pada *Google Maps.*

2.6.1 *Direction API*

Directions API adalah salah satu layanan dari *google* yang memudahkan kita (*developer*) untuk mencari rute dan navigasi dari satu tempat ke tempat tertentu.

Kita tinggal memasukkan *latitude* dan *longitude* posisi berangkat dan juga *latitude longitude* posisi tujuan. Keunggulan dari *API* ini adalah dia mudah digunakan, kita hanya tinggal melakukan *HTTP Request* untuk memanggil *Google Directions API*.

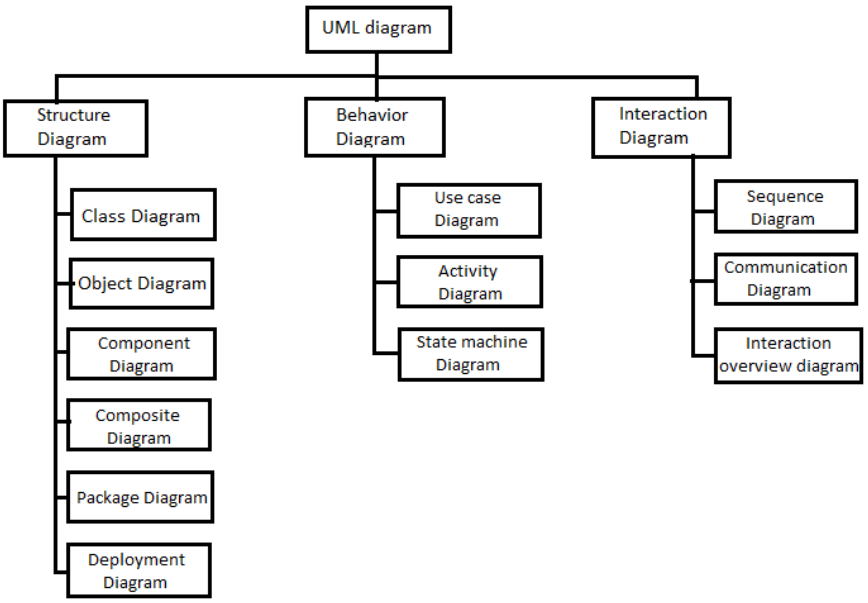
2.7 Alat Bantu Perancangan Sistem

Dalam penelitian ini, proses perancangan dilakukan dengan menggunakan beberapa alat bantu yang dapat diuraikan sebagai berikut :

2.7.1 *Unified Modeling Language* (*UML*)

Menurut Rosa dan Shalahuddin (2013:133), *Unified Modeling Language* adalah salah satu standar bahasa visual yang banyak digunakan di dunia industri untuk mengidentifikasi requirement, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. Sedangkan pandangan Whitten dan Bentley (2007, p. 371), *Unified Modeling Language* (*UML*) merupakan kumpulan dari model yang digunakan untuk menjelaskan sistem dari perangkat lunak sebagai objek.

Menurut Sukamto dan Shalahuddin (2013:140), dalam *UML* terdapat 13 *Diagram* yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam *Diagram* tersebut dapat dilihat pada Gambar 2.1 dibawah ini :



Gambar 2.1 Klasifikasi *Diagram UML*

Berikut penjelasan singkat dari Gambar 2.1.

1. *Structure Diagram,* yaitu kumpulan *Diagram-Diagram* yang menggambarkan struktur statis dari sistem yang dimodelkan.
2. *Behavior Diagram,* yaitu kumpulan *Diagram-Diagram* yang menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi di dalam sistem.
3. *Interaction Diagram,* yaitu kumpulan *Diagram-Diagram* yang menggambarkan interaksi sistem dengan sistem lain ataupun interaksi antar subsistem dalam sebuah sistem.

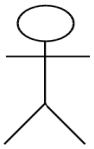
Secara garis besar, beberapa *Diagram* utama sudah dapat menggambarkan keseluruhan sistem. *Diagram* tersebut antara lain *use case Diagram, class Diagram, sequence Diagram,* dan *Activity Diagram.*

2.7.2 *Use case*

Menurut Sukamto dan Shalahuddin (2013:155), *Diagram use case* merupakan pemodelan untuk kelakuan 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. Berikut adalah simbol-simbol yang ada

pada *Diagram use case*.

Tabel 2.1 Simbol *Diagram Use case*

No.		Notasi	Nama	Deskripsi
1.			*Use case*	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*.
2.			*Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya 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>> - - - - - - - ->	*Extended* (Ekstensi)	Relasi *use case* tambahan ke sebuah *use case* dan *use case* yang ditambahkan dapat berdiri sendiri walau

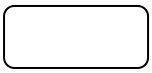
No.		Notasi	Nama	Deskripsi
				tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
5.		<<include>> < - - - - - - - -	*Include* (Menggunakan)	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. *Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan.
6.			*Generalization* (Generalisasi)	Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah *use case* dimana fungsi yang satu merupakan fungsi yang lebih umum dari lainnya. Arah panah

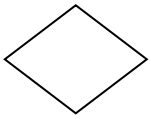
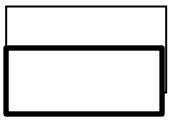
No.		Notasi	Nama	Deskripsi
				mengarah pada *use case* yang menjadi generalisasinya (umum)

2.7.3 *Activity Diagram*

Activity Diagram menggambarkan *workflow* (aliran kerja) proses bisnis dan urutan aktivitas dalam sebuah proses. *Activity Diagram* sangat mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Membuat *Activity Diagram* pada awal pemodelan proses cukup menguntungkan untuk membantu memahami keseluruhan proses. *Activity Diagram* juga bermanfaat untuk menggambarkan *parallel behavior* atau menggambarkan interaksi antara beberapa *use case.* Berikut merupakan simbol notasi *Activity Diagram* pada Tabel 2.2 (Rosa A. S dan M. Salahuddin, 2013).

Tabel 2.2 Simbol *Activity Diagram*

No	**Notasi**	**Nama**	**Deskripsi**
1		Status Awal (*initial node)*	Status awal aktivitas, sebuah *Diagram* aktivitas memiliki status awal.
2		Status Akhir (*final node)*	Status akhir yang dilakukan sistem.
3		Aktivitas (*activity)*	Aktivitas yang dilakukan oleh sistem, biasanya diawali oleh kata kerja.

No	Notasi	Nama	Deskripsi
4		Percabangan (*decision)*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
5		Penggabungan (*join)*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
6		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.7.4 *Class Diagram*

Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (Sukamto dan Shalahuddin, 2013:141). *Class Diagram* memiliki apa yang disebut atribut dan operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut:

1. *Private* (-), hanya dapat digunakan oleh *class* yang memilikinya
2. *Public* (+), dapat digunakan oleh *class* lain.
3. *Protected* (#), hanya dapat dipanggil oleh *class* yang bersangkutan jumlah suatu anak yang mewarisinya.

Nilai kardinalitas atau *multiplicity* sebuah *class* menunjukkan jumlah suatu objek yang dapat berhubungan dengan objek lain. Berikut nilai kardinalitas atau *multiplicity* pada Tabel 2.4 (Tohari, Hamim 2014) dan notasi *class Diagram* pada Tabel 2.3 (Rosa A. S dan M. Salahuddin, 2013).

Tabel 2.3 Jenis-jenis *Multiplicity*

No	Indikator	Keterangan
1	0 .. 1	Nol atau satu
2	1	Hanya satu
3	0 .. *	Nol atau lebih
4	1 .. *	Satu atau lebih

Berikut adalah simbol-simbol yang ada pada *class Diagram.*

Tabel 2.4 Simbol *Class Diagram*

No	Notasi	Nama	Deskripsi
1	**Kelas** +atribut +metode()	*Class*	Menggambarkan konsep dasar pemodelan sistem.
2	————	Asosiasi (*Association*)	Sebuah garis solid antara dua *class*, ditarik dari *class* sumber ke *class* target lebih spesifik, digunakan dalam struktur pewarisan.
3	- - - - - - - - ->	Ketergantungan (*Dependency)*	Relasi antara dua elemen jika perubahan definisi sebuah elemen (*supplier* atau sumber) dapat menyebabkan perubahan pada elemen lainnya (*Client* atau target).

2.8 Teknologi Pendukung

2.8.1 *PHP*

Welling & Thomson (2005) mengungkapkan bahwa PHP atau hypertext preprocessor merupakan bahasa pemrograman server-side script yang dirancang untuk pengembangan web. Selanjutnya Arief (2011), mengungkapkan bahwa PHP

merupakan sebuah bahasa server-side –scripting yang menyatu dengan HTML yang berfungsi untuk membuat halaman web yang dinamis. PHP merupakan server-side-scripting sehingga sintaks dan perintah-perintah dalam PHP akan dieksekusi di server, kemudian hasilnya akan dikirimkan ke browser dalam format HTML.

Berdasarkan kedua pendapat ahli tersebut, dapat disimpulkan bahwa PHP merupakan bahasa yang bersifat server-side-scripting yang berguna dalam pembuatan web dinamis yang sintaks dan perintahnya dieksekusi di dalam server untuk kemudian dikirimkan hasilnya ke web browser dalam bentuk format HTML.

2.8.2 *MySQL*

Menurut Anhar (2010) "*MySql* merupakan *software* yang tergolong *database* server yang bersifat *Open Source*. *Open Source* menyatakan bahwa software ini dilengkapi dengan *source code* (kode yang dipakai membuat *MySql*). Selain tentu saja entuk *executable*-nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi dan bisa diperoleh dengan cara mengunduh di internet secara gratis".

MySQL menggunakan *SQL* sebagai bahasa dasar untuk mengakses *database*-nya. Sistem *database MySQL* memiliki sistem sekuritas dengan tiga verifikasi yaitu *username*, *password*, dan *host*. Verifikasi *host* memungkinkan untuk membuka sekuriti di '*localhost'*, tetapi tertutup bagi *host* lain (bekerja di lokal komputer). Sistem sekuriti ini ada di dalam *database MySQL* dan pada tabel *user*. Proteksi juga dapat dilakukan terhadap *database*, tabel hingga kolom secara terpisah. Sebenarnya, *MySQL* merupakan turunan salah satu konsep utama dalam *database* sejak lama yaitu *SQL* (*Structured Query Language*). *SQL* adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Sebagai *database server*, *MySQL* dapat dikatakan lebih unggul dibandingkan *database server* lainnya dalam *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh *single user*, kecepatan *query MySQL* bisa sepuluh kali lebih cepat dari *PostgreSQL* dan lima kali lebih cepat dibandingkan *Interbase*. Saat ini *MySQL* tersedia dibawah izin *open source*, tetapi juga ada izin untuk penggunaan secara komersial.

Keunggulan dari *MySQL* adalah :

1. Bersifat *open source*.
2. Sistem yang digunakan oleh perangkat lunak ini tidak memberatkan kerja dari server, karena dapat bekerja di *background*.
3. Mempunyai koneksi yang stabil dan kecepatan yang tinggi.

2.8.3 *Javascript*

Javascript menurut Sunyoto (2007) adalah bahasa scripting yang populer di internet dan dapat bekerja di sebagian besar browser populer seperti *Internet Explorer (IE), Mozilla Firefox, Netscape* dan *Opera*. Kode *Javascript* dapat disisipkan dalam halaman web menggunakan tag *SCRIPT*. Beberapa hal tentang *Javascript*:

1. *Javascript* didesain untuk menambah interaktif suatu *web*.
2. *Javascript* merupakan sebuah bahasa *scripting*.
3. Bahasa *scripting* merupakan bahasa pemrograman yang ringan.
4. *Javascript* berisi baris kode yang dijalankan di komputer (*web browser*).
5. *Javascript* biasanya disisipkan (*embedded*) dalam halaman *HTML*.
6. *Javascript* adalah bahasa interpreter (yang berarti skrip dieksekusi tanpa proses kompilasi).

2.8.4 *HTML*

Menurut Kuswayatno (2006), *HTML* merupakan halaman yang berada pada suatu situs internet atau web. *HTML* merupakan metode yang menautkan (*link*) satu dokumen ke dokumen lain melalui teks. Menurut Deris Setiawan, *HTML* merupakan *framework* internet, hampir semua situs web yang ada menggunakan *HTML* untuk menampilkan teks, grafik, suara, dan animasinya. Sedangkan menurut Oktavian (2010), *HTML* adalah suatu bahasa yang dikenali oleh web browser untuk menampilkan informasi dengan lebih menarik dibandingkan dengan tulisan teks biasa (*plain text*).

HTML adalah sebuah bahasa markah yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah penjelajah web Internet dan pemformatan hiperteks sederhana yang ditulis dalam berkas format *ASCII* agar dapat menghasilkan tampilan wujud yang terintegrasi. Dengan

kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan dalam format *ASCII* normal sehingga menjadi halaman web dengan perintah-perintah *HTML*.

Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan *SGML (Standard Generalized Markup Language), HTML* adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web. *HTML* saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh *World Wide Web Consortium (W3C). HTML* dibuat oleh kolaborasi Caillau TIM dengan Berners-lee Robert ketika mereka bekerja di CERN pada tahun 1989 (CERN adalah lembaga penelitian fisika energi tinggi di Jenewa).

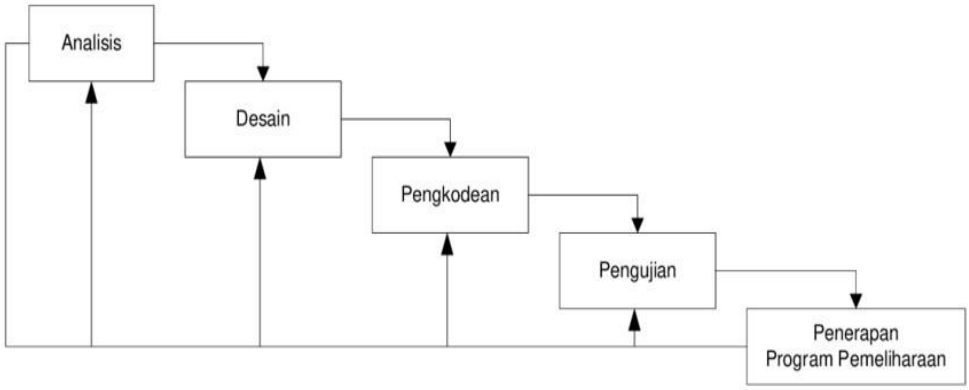
2.8.5 *Bootstrap*

Bootstrap merupakan sebuah *framework CSS* yang memudahkan pengembang untuk membangun website yang menarik dan responsif. *Bootstrap* adalah *CSS* tetapi dibentuk dengan *LESS*, sebuah *pre-prosessor* yang memberi fleksibilitas dari penggunaan CSS biasa. Bootstrap dapat dikembangkan dengan tambahan lainnya karena ini cukup fleksibel terhadap pekerjaan web yang mengutamakan desain (Otto, 2020). Selanjutnya Fauzi (2008), mengungkapkan bahwa "*Bootstrap* merupakan suatu metode berbasis komputer yang sangat potensial untuk dipergunakan pada masalah ketidakstabilan dan keakuratan, khususnya dalam menentukan interval konfidesi".

Berdasarkan kedua pendapat ahli tersebut, dapat disimpulkan bahwa *bootstrap* merupakan sebuah *framework CSS* yang dirancang untuk membantu *developer* dalam melakukan desain terhadap rancangan website yang akan dibangun agar dapat lebih menarik, interaktif dan responsive.

2.9 Model *Waterfall*

Model *Waterfall* merupakan model yang bersifat sistematis dan termasuk dalam model klasik. Menurut Ian Sommerville (2011), Model *Waterfall* memiliki lima tahapan, yaitu analisis (*analysis*), perancangan (*design*), pengkodean (*coding*), pengujian (*testing*) dan pemeliharaan (*maintenance*). Tahapan model *Waterfall* dapat dilihat pada Gambar 2.6.



Gambar 2.2 Tahapan Model *Waterfall*

Penjelasan tahapan-tahapan model *Waterfall* tersebut yaitu:

1. Analysis (Analisis)

   Tahapan ini merupakan proses analisa terhadap sistem yang sedang berjalan dengan tujuan untuk mendapatkan jawaban mengenai pengguna sistem, cara kerja sistem dan waktu penggunaan sistem, sehingga kebutuhan yang diperlukan untuk sistem baru akan didapatkan.

2. Design (Perancangan)

   Perancangan merupakan proses penentuan cara kerja sistem dalam hal perancangan antarmuka, database, dan perancangan alur program. Perancangan diperlukan untuk menggambarkan sistem baru dengan tujuan memenuhi kebutuhan pengguna.

3. Coding (pengkodean)

   Tahapan coding merupakan implementasi rancangan sistem yang dibentuk menjadi suatu kode program untuk pembuatan sistem.

4. Testing (Pengujian)

   Pengujian program dilakukan untuk mengetahui kesesuaian sistem berjalan sesuai prosedur atau tidak dan memastikan sistem terhindar dari error yang terjadi. Testing juga dilakukan untuk memastikan kevalidan dalam proses input sehingga dapat menghasilkan output yang sesuai.

5. Maintenance (Pemeliharaan)

   Tahapan ini yaitu pemeliharaan dan pengembangan sistem yang berguna untuk melihat kemampuannya, mengecek jika masih ada ditemukan error atau ada penambahan fitur-fitur yang belum ada pada sistem tersebut. Pengembangan diperlukan ketika adanya perubahan dari pengguna seperti

ketika ada pergantian sistem operasi, atau perangkat lainnya.

2.10 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan suatu teknik yang digunakan menguji apakah sebuah perangkat lunak yang dihasilkan telah sesuai dengan yang diharapkan atau belum. Menurut Pressman (2002) pengujian adalah proses eksekusi suatu program untuk menemukan kesalahan sebelum digunakan oleh pengguna akhir (end-user). Pengujian dalam penelitian ini dilakukan dengan menggunakan metode *Black Box.*

2.10.1 Pengujian *Black Box*

Menurut Pressman (2010), *Black Box* testing juga disebut pengujian tingkah laku, memusat pada kebutuhan fungsional perangkat lunak. Teknik pengujian *Black Box* memungkinkan memperoleh serangkaian kondisi masukan yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Selanjutnya Shalahuddin dan Rosa (2011) mengungkapkan bahwa *Black Box* testing merupakan pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Blackbox Testing cenderung untuk menemukan hal-hal berikut ini:

1. Fungsi yang tidak benar atau tidak ada.
2. Kesalahan antarmuka (interface errors).
3. Kesalahan pada struktur data dan akses basis data.
4. Kesalahan performansi (performance errors).
5. Kesalahan inisialisasi dan terminasi.

Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:

1. Bagaimana fungsi-fungsi diuji agar dapat dinyatakan valid?
2. Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
3. Apakah sistem sensitif pada input-input tertentu?
4. Bagaimana sekumpulan data dapat diisolasi?
5. Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?

Berdasarkan kedua pendapat ahli di atas, dapat disimpulkan bahwa pengujian *Black Box* merupakan pengujian perangkat lunak dari segi fungsional

suatu program atau aplikasi yang bertujuan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari program atau aplikasi tersebut telah sesuai dengan spesifikasi yang dibutuhkan.

2.10.2 Pengujian *System Usability Scale* (*SUS*)

Usability adalah analisa kualitatif yang menentukan seberapa mudah user menggunakan antarmuka suatu aplikasi (Nielsen, 2012). Suatu aplikasi disebut *usable* jika fungsi-fungsinya dapat dijalankan secara efektif, efisien, dan memuaskan (Nielsen, 1993). Secara umum Usability merupakan tingkat kualitas dari suatu sistem yang mudah digunakan, mudah dipelajari dan mendorong pengguna untuk menggunakan sistem sebagai alat bantu dalam menyelesaikan suatu tugas atau pekerjaan, hal ini berpengaruh pada *user experience* (UX). Pengujian ini dilakukan dengan harapan mencapai tujuan dimana pengguna mendapatkan pengalaman yang baik selama menggunakan website. Pengujian Usability dijadikan sebagai tolak ukur kualitas pengalaman pengguna ketika berinteraksi dengan suatu sistem seperti website, android maupun aplikasi perangkat lunak lainnya yang dioperasikan oleh pengguna.

System Usability Scale (SUS) merupakan kuesioner yang dapat digunakan untuk mengukur usability sistem komputer menurut sudut pandang subyektif pengguna (Brooke, 2013). SUS diciptakan oleh John Brooke pada tahun 1986 dan dahulu digunakan untuk menguji sistem elektronik kantor. *System Usability Scale* (SUS) berisi 10 pertanyaan dimana partisipan diberikan pilihan skala 1 sampai 5 untuk dijawab berdasarkan pada seberapa banyak mereka setuju dengan setiap pernyataan tersebut terhadap produk atau fitur yang diuji. Nilai 1 berarti sangat tidak setuju dan 5 berarti sangat setuju dengan pernyataan tersebut. Menurut Brooke (1996) Skor SUS berkisar dari 0 hingga 100.