BAB 2
LANDASAN TEORI

2.1 Grammar Tenses

Grammar atau tata bahasa merupakan himpunan dari aturan-aturan yang terstruktur dan yang mengatur susunan kalimat, frase, dan kata dalam bahasa apapun. Dalam bahasa inggris, penggunaan grammar akan selalu berhubungan dengan *Tenses*. Tenses merupakan perubahan bentuk kata kerja untuk mengungkapkan suatu peristiwa yang terkait erat dengan waktu kejadian. Secara umum Tenses dibagi menjadi 3 bagian yaitu *Past Tense* (waktu lampau), *Present Tense* (waktu saat ini), dan *Future Tense* (waktu yang akan datang) [9].

2.1.1 Past Tense

Past Tense merupakan sebuah *Tenses* yang digunakan berdasarkan sesuatu yang telah terjadi di waktu lampau. Past Tense dibagi menjadi 4 bagian yaitu sebagai berikut [10]:

1. *Simple Past Tense*

   Simple Past Tense merupakan bagian Past Tense yang paling sederhana dan digunakan untuk menyatakan peristiwa atau kegiatan yang berlangsung di masa lalu.

2. *Past Continuous Tense*

   Past Continuous Tense merupakan bentuk Tenses yang digunakan untuk menyatakan peristiwa atau kegiatan yang berlangsung di masa lalu ketika ada peristiwa lain terjadi.

3. *Past Perfect Tense*

   Past Perfect Tense merupakan bentuk Tenses yang digunakan untuk menyatakan peristiwa atau kegiatan yang berlangsung sebelum peristiwa lampau terjadi.

4. *Past Perfect Continuous Tense*

Past Perfect Continuous Tense merupakan bentuk Tenses yang digunakan untuk menyatakan peristiwa atau kegiatan yang telah berlangsung sebelum peristiwa lain terjadi di masa lampau.

Tabel 2.1 Rumus dan Contoh dari Tipe-Tipe Past Tense

Tenses	Rumus	Contoh Kalimat
Simple Past Tense	Subject + Verb (v2) or Irregular Verb	*I played basketball yesterday*
Past Continuous Tense	Subject + was/were + Verb (+ing)	*I was playing basketball when Mary came*
Past Perfect Tense	Subject + had + Verb (v3)	*I had played basketball before Mary came*
Past Perfect Continuous Tense	Subject + had + been + Verb (+ing)	*I had been playing basketball when Mary came*

2.1.2 Present Tense

Present Tense merupakan sebuah *Tenses* yang digunakan berdasarkan sesuatu yang saat ini sedang dilakukan. Present Tense dibagi menjadi 4 bagian yaitu sebagai berikut [10]:

1. *Simple Present Tense*

   Simple Present Tense merupakan bagian Present Tense yang paling sederhana dan digunakan untuk menyatakan kebiasaan, kebenaran, atau kondisi saat ini.

2. *Present Continuous Tense*

   Present Continous Tense merupakan bentuk Tenses yang digunakan untuk menyatakan peristiwa atau kegiatan yang sedang berlangsung saat dibicarakan. Bisa sekarang, hari ini, atau sekitar saat ini.

3. *Present Perfect Tense*

   Present Perfect Tense merupakan bentuk Tenses yang digunakan untuk menyatakan peristiwa atau kegiatan yang berlangsung di waktu lampau dan masih ada hubungannya dengan saat ini.

4. *Present Perfect Continuous Tense*

   Present Perfect Continuous Tense merupakan bentuk Tenses yang digunakan untuk menyatakan peristiwa atau kegiatan yang berlangsung di waktu lampau dan masih berlangsung, sudah selesai, atau diulang-ulang.

Tabel 2.2 Rumus dan Contoh dari Tipe-Tipe Present Tense

Tenses	Rumus	Contoh Kalimat
Simple Present Tense	Subject + Verb (v1) + es/es	*I play basketball every week*
Present Continuous Tense	Subject + is/am/are + Verb (+ing)	*I'm playing basketball*
Present Perfect Tense	Subject + has/have + Verb (v3)	*I have played basketball*
Present Perfect Continuous Tense	Subject + has/have + been + Verb (+ing)	*I have been playing basketball for 3 hours*

2.1.3 Future Tense

Future Tense merupakan sebuah *Tenses* yang digunakan berdasarkan suatu perbuatan yang belum terjadi, dan masih akan dilakukan oleh subyeknya. Future Tense dibagi menjadi 4 bagian yaitu sebagai berikut [10]:

1. *Simple Future Tense*

   Simple Future Tense merupakan bagian Future Tense yang paling sederhana dan digunakan untuk menyatakan peristiwa atau kegiatan yang akan terjadi di waktu yang akan datang.

2. *Future Continuous Tense*

   Future Continuous Tense merupakan bentuk Tenses yang digunakan untuk menyatakan peristiwa atau kegiatan yang akan terjadi di waktu tertentu di waktu yang akan datang.

3. *Future Perfect Tense*

Future Perfect Tense merupakan bentuk Tenses yang digunakan untuk menyatakan peristiwa atau kegiatan yang akan diselesaikan di waktu tertentu di waktu yang akan datang.

4. *Future Perfect Continuous Tense*

   Future Perfect Continuous Tense merupakan bentuk Tenses yang digunakan untuk menyatakan peristiwa atau kegiatan yang dimulai pada waktu tertentu dan masih berlangsung ketika peristiwa lain terjadi di waktu tertentu di waktu yang akan datang.

Tabel 2.3 Rumus dan Contoh dari Tipe-Tipe Future Tense

Tenses	Rumus	Contoh Kalimat
Simple Future Tense	Subject + will/shall + Verb (v1)	*I will play basketball next week*
Future Continuous Tense	Subject + will be/shall be + Verb (+ing)	*I will be playing basketball this afternoon*
Future Perfect Tense	Subject + will have + Verb (v3)	*I will have played basketball by tomorrow*
Future Perfect Continuous Tense	Subject + will have + been + Verb (+ing)	*I will have been playing basketball for 20 minutes*

2.2 WhatsApp

WhatsApp adalah aplikasi pesan untuk smartphone yang mampu berjalan lintas *platform* seperti *Apple iOS, Blackberry, Android, Nokia,* dan *Windows Phone.* Seperti layanan *instant messanger* lainnya, *WhatsApp* menggunakan paket data internet dan menggunakan koneksi data *mobile* serta WiFi untuk melangsungkan komunikasi data, dengan begitu seseorang dapat melakukan obrolan online, berbagi file dan bertukar foto [11]. WhatsApp pertama kali didirikan oleh Jan Koum dan Brian Acton di Santa Clara, California pada tahun 2009. Pada bulan Februari tahun 2014, perusahaan Facebook membeli saham WhatsApp sebesar 19 miliar dollar Amerika (USD) [12].

Menurut data statistik website statista menunjukan bahwa jumlah pengguna WhatsApp aktif bulanan di seluruh dunia per-Maret 2020. Pada bulan tersebut, website statista mengumumkan lebih dari 2 miliar pengguna bulanan, naik mencapai 500 juta pengguna pada Desember 2017. Di Amerika Serikat, pengguna WhatsApp berjumlah 68,1 juta pengguna pada tahun 2019 dan akan diperkirakan akan tumbuh menjadi 85,8 juta pengguna pada tahun 2023 [13].

Untuk mendukung pengembangan chatbot, peneliti akan menggunakan API yaitu WhatsApp Business API. WhatsApp Business API menyediakan API resmi yang dapat digunakan oleh pengembang untuk membuat bot, namun API tersebut tidak lah gratis dan harus mengeluarkan biaya [14]. Pada penelitian kali ini, peneliti menggunakan WhatsApp Business API dari pihak ketiga.

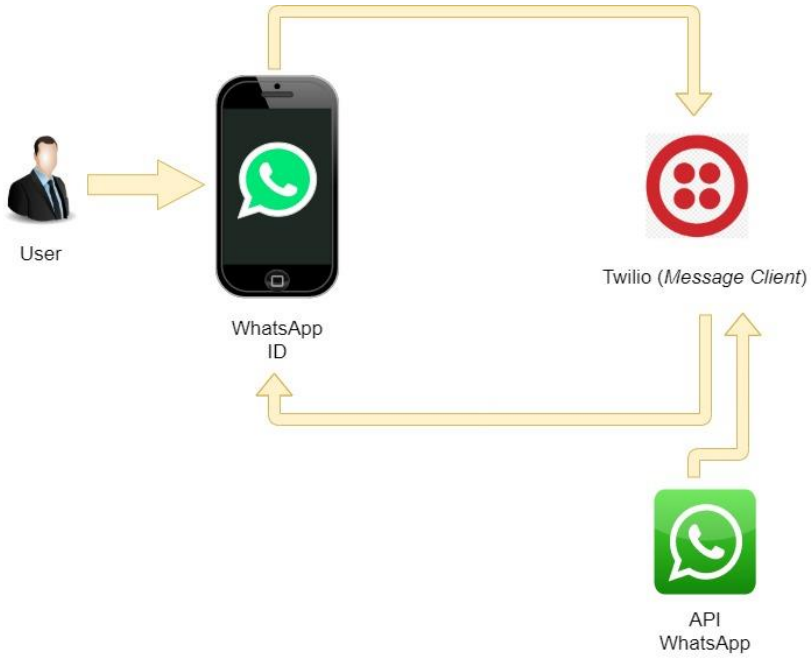
Gambar 2.1 Logo WhatsApp

2.2.1 WhatsApp Business API

WhatsApp Business API adalah sebuah API yang dikembangkan untuk membangun aplikasi bot pada platform WhatsApp khususnya dipergunakan dalam dunia bisnis, baik itu bisnis skala kecil, menengah, maupun dalam skala besar [15]. WhatsApp Business API mengizinkan perusahaan atau organisasi untuk berkomunikasi dengan banyak audience mereka di aplikasi WhatsApp yang sudah terpasang [16]. Dalam pengajuan untuk mendapatkan akses WhatsApp Business API, pengguna akan diminta nama perusahaan atau organisasinya lalu melakukan verifikasi melalui Facebook Business Manager [17].

Pada penelitian kali ini, peneliti akan dibantu oleh platform Twilio. Twilio adalah platform komunikasi yang dapat dimanfaatkan oleh pengembang untuk memberikan kapabilitas pengiriman teks pesan, pesan suara, dan pesan video bagi perangkat lunak. Twilio juga menyediakan sebuah akun WhatsApp yang dapat digunakan sebagai media komunikasi antara user dengan bot. Setiap pesan yang

akan dikirimkan dari user akun tersebut akan diteruskan ke backend oleh Twilio dan ketika backend memberikan balasan maka Twilio akan mengarahkan balasan tersebut ke user yang melakukan chat sebelumnya [14].



Gambar 2.2 Alur Kerja WhatsApp Business API

2.3 Chatbot

Chatbot merupakan sebuah pengembangan aplikasi komputer yang dirancang untuk dapat berinteraksi dengan manusia melalui pesan teks maupun suara. Chatbot telah dipersiapkan dengan kecerdasan buatan dan pemrosesan bahasa alami (NLP) yang membuatnya menjadi aplikasi komputer yang cerdas dan dapat menjawab pertanyaan dari manusia [18]. Chatbot terdiri dari 2 komponen utama yaitu *Chat* dan *Bot*, Chat yang dapat diartikan pembicaraan dan Bot adalah suatu program yang mengandung sejumlah data, jika diberikan masukan maka akan memberikan jawaban [19]. Chatbot pertama kali dikembangkan pada tahun 1966 dengan nama bot “ELIZA”, bot ELIZA ini digunakan untuk meniru terapis dengan menggunakan aturan pemahaman bahasa yang sederhana [20]. Chatbot terdiri dari 3 kombinasi yang membentuk sebuah chatbot, diantara lainnya adalah sebagai berikut [18]:

1. *User Interface*

   User Interface merupakan tampilan antar muka dalam chatbot yang merupakan jembatan antara chatbot dan user saling berinteraksi. User Interface dapat memberikan pengalaman yang baik kepada user ketika berinteraksi dengan chatbot melalui aplikasi pesan berbasis text.

2. *Artificial Intelligence*

   Atau biasa disebut AI akan membuat aplikasi paham dalam setiap interaksi yang terjadi dengan user, dikarenakan chatbot merupakan salah satu pengembangan aplikasi sistem cerdas antara manusia dengan komputer.

3. *Integrasi*

   Integrasi dengan sistem lainnya akan menambah kekayaan fitur yang terdapat di dalam suatu chatbot. Dengan mengintegritasikan chatbot ke sistem yang lain dapat menyediakan informasi tambahan. Dengan begitu chatbot mampu memberikan informasi yang lebih kaya kepada user.

2.4 Application Programming Interface (API)

Application Programming Interface atau biasa disebut API merupakan interface (sarana atau media dari dua sistem yang terpisah untuk berkomunikasi) yang mengimplementasikan aplikasi yang memungkinkan aplikasi lain untuk berkomunikasi dan saling bertukar data dengannya [21]. Dengan kata lain, API juga merupakan sekumpulan perintah, fungsi, komponen, dan protokol yang telah disediakan oleh sistem operasi atau bahasa pemrograman tertentu yang dapat digunakan oleh programmer saat membangun perangkat lunak. Dalam API terdapat fungsi atau perintah untuk menggantikan sebuah bahasa yang digunakan dalam *system calls* dengan bahasa yang lebih terstruktur dan mudah dimengerti oleh programmer [22].

2.4.1 API Grammarly

Grammarly memiliki API yang berfungsi untuk melakukan pengecekan dan pengkoreksian grammar bahasa inggris. Grammarly juga menggunakan *Natural Language Processing* (NLP) dan *Machine Learning* dalam pemrosesan pengecekan grammar bahasa inggris dan untuk kepentingan dalam pengembangan aplikasi

tersebut [23]. Namun sayangnya, Grammarly tidak menyediakan API secara resmi sehingga pengembang tidak bisa mendapatkan dan menggunakan API Grammarly dikarenakan akan menimbulkan resiko dalam aturan bisnis Grammarly itu sendiri [24]. Pada penelitian kali ini, peneliti akan menggunakan API Grammarly yang berasal dari salah satu pihak ketiga yang diunggah di situs GitHub.

Gambar 2.3 Logo Grammarly

2.5 Algoritma String Matching Knuth Morris-Pratt

Algoritma Knuth Morris Pratt (KMP) dikembangkan oleh D. E. Knuth, Bersama dengan J. H. Morris dan V. R. Pratt. Untuk pencarian string dengan algoritma Brute Force , setiap kali ditemukan ketidakcocokan dengan teks, maka pattern akan digeser satu karakter ke kanan. Berbeda dengan algoritma Brute Force, informasi yang digunakan masih dipelihara untuk melakukan jumlah pergeseran. Algoritma menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh, tidak hanya satu karakter [25].

Perbandingan antara algoritma brute force dengan algoritma KMP ditunjukkkan dengan perpindahan posisi pattern terhadap posisi teks. Dimana dalam hal pencocokan string, pattern yang dicocokkan berawal dari kiri teks.

Kompleksitas algoritma pencocokan string dihitung dari jumlah operasi perbandingan yang dilakukan. Kompleksitas waktu terbaik dari algoritma brute force adalah $O(n)$. Kasus terbaik terjadi jika pada operasi perbandingan, setiap huruf pattern yang dicocokkan dengan awal dari teks adalah sama. Kompleksitas waktu terburuk dari brute force adalah $O(mn)$ [25].

Algoritma Knuth-Morris-Praat (KMP) merupakan proses pencocokan string. Bila terjadi ketidak cocokan pada saat pattern sejajar dengan teks [i...i + n – 1], kita bisa menganggap ketidak cocokan pertama terjadi diantara teks [i + j] dan

pattern [j], dengan <j<n. Berarti, teks [i…i + j] = pattern [0…j + 1] dan a = teks [i + j] tidak sama dengan b = pattern [j]

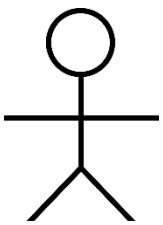
2.6 UML (Unified Modeling Language)

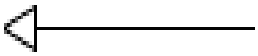
Unified Modeling Language atau biasa disebut UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak [26]. UML memiliki diagram-diagram yang digunakan dalam aplikasi berorientasi objek yaitu sebagai berikut:

2.6.1 Use Case Diagram

Use Case Diagram merupakan pemodelan untuk melakukan *behavior* sistem informasi yang akan dibuat dan digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut [26]. Berikut adalah simbol-simbol yang ada pada diagram *Use Case* [27]:

Tabel 2.4 Simbol-simbol Diagram *Use Case*

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling tertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja diawal *frase* nama *Use Case*
Aktor / *actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar 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 diawal *frase* nama aktor
Asosiasi / *association*	Komunikasi antara aktor dan *use case* yang berpartisifasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Ekstensi / *extend* <<extend>> 	Relasi *use case* tambahan ke sebuah *use case* dinamakan *use case* yang ditambahkan dapat berdiri sendir walau tanpa use case tambahan itu; mirip dengan prinsip intherince pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan
Generalisasi / *generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.

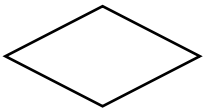
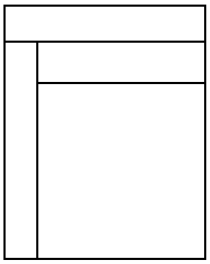
Sumber : Rosa dan Shalahuddin (2014 : 156)

2.6.2 Activity Diagram

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan oleh aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Berikut adalah simbol-simbol yang ada pada diagram aktivitas [27]:

Tabel 2.5 Simbol-simbol Diagram Aktivitas

Simbol	Deskripsi

Status Awal 	Status Awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

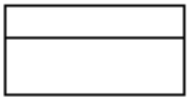
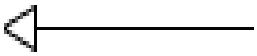
Sumber : Rosa dan Shalahuddin (2014 : 162)

2.6.3 Class Diagram

Class Diagram menggambarkan sturktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Berikut adalah simbol-simbol yang ada pada diagram kelas [27]:

Tabel 2.6 Simbol-simbol Diagram Kelas

Simbol	Deskripsi

Kelas 	Kelas pada struktur sistem
Antarmuka / *interface*  nama_interface	Sama dengan konsep interface dalam pemrograman berorientasi objek
Asosiasi / *association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *Directed Association* 	Relasi antar kelas dengan makna kelas satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi
Kebergantungan / *Dependency* 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi / *Aggregation* 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

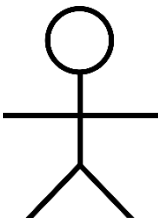
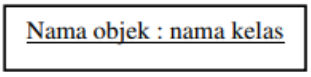
Sumber : Rosa dan Shalahuddin (2013 : 146)

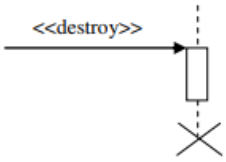
2.6.4 Sequence Diagram

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Diagram sekuen harus diketahui objek-objek yang terlibat dalam

sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek tersebut. Berikut adalah simbol-simbol yang ada pada diagram sekuen [27]:

Tabel 2.7 Simbol-simbol Diagram Sekuen

Simbol	Deskripsi
Aktor 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar 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 diawal *frase* nama aktor
Garis hidup / *lifeline*	Menyatakan kehidupan suatu objek
Objek 	Menyatakan objek yang berinteraksi pesan
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya
Pesan tipe *create* <<create>> 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat

Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri. 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
Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe *return* 1: keluaran 	Menyatakan suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*

Sumber : Rosa dan Shalahuddin (2014 : 165)

2.7 Pengujian *Alpha*

Pengujian Alpha merupakan salah satu strategi dalam pengujian perangkat lunak yang paling banyak digunakan dalam pengembangan perangkat lunak.

Pengujian ini dilakukan pada sisi pengembang perangkat lunak. Pengujian Alpha merupakan pengujian yang dilakukan oleh pengembang atau pembuat sistem untuk mengetahui trouble shooting dari produk yang dikembangkan. Pengujian Alpha berfungsi sebagai bentuk pengujian penerimaan internal (pihak pengembang) [28].

2.7.1 Metode Black Box

Metode Black Box merupakan salah satu teknik pengujian perangkat lunak yang berfokus pada spesifikasi fungsional dari perangkat lunak. Pengujian Blackbox bekerja dengan mengabaikan struktur kontrol sehingga perhatiannya difokuskan pada informasi utama. Pengujian Blackbox memungkinkan pengembang perangkat lunak untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program [29]. Berikut adalah keuntungan dan kekurangan dalam menggunakan metode Blackbox [29]:

2.7.1.1 Keuntungan

1. Penguji tidak perlu memiliki pengetahuan tentang bahasa pemrograman tertentu
2. Pengujian dilakukan dari sudut pandang pengguna, ini membantu untuk mengungkapkan ambiguitas atau inkonsistensi dalam spesifikasi persyaratan
3. *Programmer* dan *tester* keduanya saling bergantung satu sama lain

2.7.1.2 Kekurangan

1. Uji kasus sulit di desain tanpa spesifikasi yang jelas
2. Kemungkinan memiliki pengulangan tes yang sudah dilakukan oleh programmer
3. Beberapa bagian *back end* tidak diuji sama sekali

2.8 Perangakat Lunak Pendukung

Perangkat lunak pendukung merupakan beberapa perangkat lunak seperti aplikasi, bahasa pemrograman, dan lainnya. Perangkat lunak pendukung ini digunakan dalam pembangunan perangkat lunak pada penelitian ini. Berikut

merupakan beberapa perangkat lunak pendukung yang diperlukan pada penelitian ini.

2.8.1 HTML

HTML adalah singkatan dari *Hypertext Markup Language* yang dikembangkan pertama kali pada tahun 2004 oleh organisasi bernama WHAT Working Group (WHATWG). Anggota dari organisasi tersebut merupakan gabungan dari apple, mozilla, dan opera. HTML berasal dari kata Hypertext yang berarti membuat text menjadi link karena hakikatnya sebuah website adalah dokumen yang mengandung banyak link untuk menghubungkan antara dokumen satu dengan dokumen lainnya. Kata selanjutnya yaitu Markup Language yang berarti menggunakan tanda untuk menandai bagian-bagian dari teks agar memiliki tampilan/fungsi tertentu. Pada tahun 2008, HTML versi pertama diperkenalkan. Di tahun yang sama pula browser Mozilla menjadi browser pertama yang mensupport HTML, kemudian disusul oleh chrome, safari, dan internet explorer. Struktur dasar HTML setidaknya memiliki 4 unsur tag yaitu tag DTD atau DOCTYPE, tag HTML, tag HEAD, dan tag BODY [30].

2.8.2 CSS

CSS adalah singkatan dari *Cascading Style Sheets* yang merupakan sebuah dokumen berisi aturan untuk memisahkan isi dengan layout dalam halaman-halaman web yang dibuat. CSS memperkenalkan template yang berupa *style* untuk dibuat dan mengizinkan penulisan kode yang mudah pada halaman web yang dirancang. CSS juga mampu menciptakan tampilan halaman yang sama pada layar dengan resolusi yang berbeda. Cara kerja CSS sangat mudah karena hanya membutuhkan style sebagai penentu format atribut pada halaman web yang dibuat. Terdapat 3 cara untuk menuliskan CSS yaitu [30] :

1. *Inline*

   Inline berarti kode CSS yang dituliskan berada pada atribut HTML yang akan diberi *style*.
2. *Embedded Class*

Embedded Class berarti kode CSS dituliskan dengan klausa *class* pada atribut HTML dan *style* CSS berada pada tag head dan diberi tag *style* untuk penulisan kode CSS tersebut.

3. *Linked Style Sheet*

   Linked Style Sheet berarti kode CSS ditulis pada file terpisah dengan ekstensi file .css, kemudian file CSS tersebut dipanggil di dalam file HTML yang ingin diberi *style*.

2.8.3 Javascript

Javascript merupakan bahasa skript populer yang digunakan untuk menciptakan halaman web yang dapat berinteraksi dengan *user* dan dapat merespon *event* yang terjadi pada halaman. *Javascript* juga merupakan perekat yang dapat menyatukan beberapa halaman web. *Javascript* dibangun secara langsung ke dalam browser seperti Microsoft Internet Explore, Mozilla Firefox, Google Chrome, dan Opera. Dalam sintaksis, *Javascript* mirip dengan bahasa C, Perl, dan Java.

Javascript berperan sebagai bahasa pemrograman yang memiliki konstruksi-konstruksi dasar seperti variabel dan tipe data, look control, statemen *if/else*, statemen *switch*, fungsi, dan objek. *Javascript* juga dapat dipakai untuk perhitungan aritmatik, pemanipulasian tanggal dan waktu, pemodifikasian array, string, dan objek. *Javascript* dapat dikombinasikan dengan bahasa pemrograman lainnya seperti HTML, CSS, dan PHP [31].

2.8.4 Node.JS

Node.JS merupakan sistem perangkat lunak yang didesain untuk pengembangan aplikasi web. Node.JS dapat juga disebut sebagai *runtime environment*. Aplikasi ini ditulis dalam campuran bahasa C++ dan Javascript, mempunyai model *event driven* dan *asynchrounous* I/O. Node.JS juga digunakan sebagai aplikasi server, dikarenakan Node.JS dapat berjalan di server dan terdapat dukungan dari V8 *Engine* buatan Google dan beberapa modul bawaan yang terintegrasi seperti modul http, modul filesystem, modul security, dan modul yang lainnya [32].

2.8.5 PHP

PHP atau Hypertext Preprocessor merupakan salah satu bahasa pemrograman berbasis web yang ditulis oleh dan untuk pengembang web. PHP pertama kali dikembangkan oleh Rasmus Lerdorf pada akhir tahun 1994. PHP dibangun dengan konsep berorientasi objek (OO) secara penuh seperti halnya bahasa pemrograman seperti Java dan C++ [33]. Berikut adalah keunggulan dalam menggunakan PHP yaitu sebagai berikut [33]:

1. Gratis, PHP dapat diunduh dan dipergunakan secara gratis.
2. PHP berlisensi GNU General Public License (GPL).
3. Performa dari PHP sangat handal dan efisien.
4. PHP mendukung hampir semua perangkat basis data seperti MySQL, Oracle, PostgreSQL, Informix, Interbase, Sybase, MariaDB, dan SQLite.
5. PHP dapat dijalankan hampir semua sistem operasi seperti Linux, Unix, Windows, Mac OS, FreeBSD, Sun Solaris, dan dapat berjalan di sistem operasi Android.
6. Perintah-perintah PHP sangat mudah untuk dipelajari.

2.8.6 MySQL

MySQL adalah program basis data yang mampu mengirim dan menerima data dengan sangat cepat dan multi user. MySQL pertama kali dirintis oleh seorang programmer basis data bernama Michael Widenius. MySQL dirilis pertama kali secara internal pada 23 Mei 1995. MySQL dapat menangani data dengan volume yang besar namun membutuhkan resource yang besar juga. Terdapat 2 lisensi pada DBMS ini yaitu free software dan shareware. Adapun keunggulan MySQL disbanding basis data lain, diantaranya sebagai berikut :

1. MySQL merupakan server tercepat.
2. MySQL mempunyai performa yang tinggi namun sederhana.
3. MySQL bersifat open source dan free sehingga siapapun dapat menggunakannya.
4. MySQL dapat diakses melalui protocol *Open Database Connectivity* (ODBC) buatan Microsoft sehingga MySQL dapat diakses oleh banyak software.

5. MySQL dapat berjalan di berbagai operating system seperti *linux, windows, solaris,* dan lain-lain.