BAB II
TINJAUAN PUSTAKA

2.1 Penyakit Tuberkulosis

Penularan penyakit tuberkulosis disebabkan oleh bakteri *Mycobaciterium tuberculosis*. Bakteri *Mycobacterium* memiliki berbagai spesies diantaranya *M. tuberculosis, M. africanum, M. bovis, M. Leprae*. Bakteri TBC disebut sebagai Bakteri Taham Asam (BTA) dikarenakan dapat hidup di lingkungan asam. Selain *Mycobacterium tuberculosis*, terdapat kelompok bakteri *Mycobacterium* yang dapat menggangu saluran pernafasan yang dikenal sebagai MOTT (*Mycobacterium Other Than Tuberculosis*). Penegakan diagnosis dan pengobatan TBC dapat terganggu dengan adanya kelompok bakteri MOTT (Pusat Data Dan Informasi Kementerian Kesehatan RI, 2018).

Pasien TBC paru akan mengalami gejala utama batuk berdahak selama 2 minggu atau lebih. Selain gejala batuk berdahak, terdapat gejalah tambahan yang dapat dialami seperti sesak nafas, dahak bercampur darah, batuk darah, badan lemas, berat badan menurun, nafsu makan menurun, malaise, demam meriang selama lebih dari satu bulan, dan berkeringat malam hari tanpa kegiatan fisik. Pada pasien dengan HIV positif, batuk sering kali bukan merupakan gejala TBC yang khas, sehingga gejala batuk tidak harus selalu selama 2 minggu atau lebih (Pusat Data Dan Informasi Kementerian Kesehatan RI, 2018).

Lama pengobatan tuberkulosis paru berlangsung selama minimal 6 bulan atau dapat bertambah dipengaruhi hasil akhir pengobatan. Proses pengobatan dibagi menjadi 2 fase yaitu fase intensif dan fase lanjutan. Fase intensif adalah fase pengobatan awal yang mana pasien harus mengkonsumsi obat TBC setiap hari dalam seminggu selama 2 bulan. Obat yang dikonsumsi pada tahap intensif diantaranya *rifampisin, isoniasid, pirasinamid*, dan *etambutol*. Fase lanjutan adalah fase setelah pasien dinyatakan berhasil melakukan fase intensif. Pada fase lanjutan, pasien mengkonsumsi obat TBC 3 kali dalam seminggu selama 4 bulan.

Obat yang dikonsum pada tahap lanjutan diantara lain *rifampisin*, *isoniasid*, *pirasinamid* (Andri et al., 2020).

2.2 Puskesmas Karya Mulia

Puskesmas Karya Mulia resmi dinyatakan beroperasi pada tanggal 7 Agustus 1996. Puskesmas Karya Mulia berlokasi di Jalan Ampera Kecamatan Pontianak Kota. Cakupan wilayah kerja dari Puskesmas Karya Mulia berdasarkan perubahan tahun 2020, adalah wilayah kelurahan Sungai Bangkong dengan luas daerah ± 2 Km2 dan wilayah Sungai Jawi ± 2,2 Km2 yang terdiri dari 111 RT/21 RW (UPT Puskesmas Karya Mulia, 2020).

Berdasarkan data rekap pasien TBC Puskesmas Karya Mulia pada tahun 2020, total pasien TBC yang berobat pada Puskesmas Karya Mulia sebanyak 36 pasien. Dari total 36 pasien yang melalui proses pengobatan, didapati sebanyak 33 pasien berhasil sembuh dengan kategori BTA positif maupun ronsen positif. 3 pasien lainnya adalah pasien yang mengalami gagal pengboatan. Pasien yang mengalami gagal pengobatan diantaranya 2 pasien meninggal selama proses pengobatan dan 1 pasien pindah pengobatan sehingga tidak bisa dilakukan pemantauan pengobatan oleh Puskesmas Karya Mulia. (UPT Puskesmas Karya Mulia, 2020).

Selain aktif dalam melakukan pemantauan dan pengobatan, Puskesmas Karya Mulia juga melakukan sebuah program inovasi untuk meningkatan angka kesembuhan pada kasus TBC. Salah satu diantaranya adalah “KEPO TB”. KEPO TB adalah kegiatan edukasi yang dilakukan untuk memberikan pemahaman terhadap masyarakat agar mau melakukan *screening* dan melakukan pemeriksaan TBC secara dini. Kegiatan KEPO TBC dilakukan dengan metode pelayanan di dalam gedung dan luar gedung. Kegiatan di dalam gedung dengan membuat pojok KEPO TBC yang menyediakan informasi terkait dengan TBC. Selain itu, dalam proses pengobatan TBC Puskesmas Karya Mulia memberikan sebuah buku pemantauan pengobatan yang diisi oleh pasien setiap kali melakukan aktivitas minum obat (UPT Puskesmas Karya Mulia, 2020).

2.2.1 Pengobatan Pasien TB di Puskesmas Karya Mulia

Puskesmas Karya Mulia melakukan standarisasi operasional yang telah ditetapkan puskesmasnya untuk menangani pasien-pasien penderita TBC dimulai dari pasien dinyatakan menderita penyakit TBC sampai ke pembimbingan pengobatan pasien menuju kesembuhan. Pada operasionalnya, pasien yang memiliki keluhan penyakit datang ke Puskesmas Karya Mulia untuk berobat. Dokter melakukan pemeriksaan kepada pasien dan mengambil langkah mengenai penyakit pasien. Pasien yang didapati menderita penyakit TBC, akan dilakukan pendataan untuk dilakukannya pemantauan selama masa pengobatan.

Kepatuhan dan kerjasama antara pihak puskesmas, pasien, dan orang terdekat pasien mendukung keberhasilan dari kesembuhan pasien TBC. Pasien TBC dapat dinyatakan gagal dalam melakukan pengobatan jika tidak menaati instruksi konsumsi obat yang telah diberikan oleh dokter. Jika hal ini terjadi, pasien TBC akan melakukan pengobatan dari awal sesuai instruksi yang diberikan dokter.

2.3 Teknologi Pendukung Penelitian

2.3.1 Android

Android adalah sebuah sistem operasi yang yang digunakan pada sebuah perangkat bergerak dan populer digunakan pada sebuah ponsel cerdas. Tidak hanya untuk ponsel cerdas, android merupakan platform pemrograman yang dikembangkan untuk berbagai perangkat seluler, contohnya tablet. Berbagai vendor pengembang ponsel cerdas menggunakan sistem operasi andorid, sehingga android dapat berjalan di berbagai macam perangkat cerdas. Pada pengembangan perangkat lunak, android menyertakan sebuah paket yang digunakan untuk penulisan kode dan perakitan modul perangkat lunak dalam pengembangan aplikasi android. Selain paket pengembangan, terdapat sebuah pasar yang digunakan untuk mendistribusikan aplikasi yang telah dikembangkan. Lengkapnya fasilitas yang tersedia, menciptakan sebuah ekosistemnya tersendiri

yang mendukung pengembangan dari aplikasi android (Herlinah & Musliadi, 2019).

Pada awal pengembangan android, andorid pertama kali dikembangkan oleh perusahaan Android Inc. Pada tahun 2005, sistem operasi android dibeli oleh Google dan menjadikan sistem operasi bersifat "*Open Source*" yang memungkinkan untuk siapa saja menggunakannya dengan gratis dan kebebasan menggunakan kode sumber yang digunakan dalam pengembangan sistem operasi tersebut (Herlinah & Musliadi, 2019).

Terdapat berbagai alasan bagi seorang developer untuk mengembangkan aplikasi berbasis android. Diantaranya menjawab kebutuhan bisnis, membangun layanan baru, membuat bisnis baru, dan menyediakan *game* serta jenis materi lainnya untuk pengguna. Selain itu, aplikasi android menjangkau sebagian besar pengguna perangkat seluler dalam berbagai bidang (Herlinah & Musliadi, 2019).

2.3.2 Android Studio

Android studio adalah sebuah *integrated development environment* (IDE), lingkungan pengembangan perangkat lunak terpadu yang digunakan untuk mengembangkan sebuah aplikasi android, berdasarkan IntelliJ IDEA. Selain menjadr editor kode IntelliJ dan pengembang yang berdaya guna, terdapat berbagai banyak fitur yang dapat digunakan untuk meningkatkan produktivitas developer dalam membuat aplikasi andorid, antara lain:

- Sistem versi berbasis *gradle* yang fleksibel.
- Emulator yang cepat dan kaya fitur.
- Lingkungan yang menyatu untuk pengembangan bagi semua perangkat android.
- *Instant run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru.

- Kode *templet* dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.

- Memiliki alat pengujian dan kerangkat kerja yang ekstensif.

Dengan beragammya fitur yang diberikan oleh Google *Cloud Platform*, akan memudahkan pengintegrasian Google *Cloud Messaging* dan *App Engine* (Herlinah & Musliadi, 2019).

2.3.3 Java

Java merupakan bahasa pemrograman berorientasi objek yang dikenalkan pada tahun 1995 oleh sebuah perusaan bernama Sun Microsystem Inc yang pada saat itu dipimpin oleh James Gosling. Penciptaan bahasa pemrograman java berawal dari keinginan perusahaan Sun Microsystem untuk membuat sebuah bahasa pemrograman yang dapat berjalan di berbagai *device* dengan tanpa terikat oleh platform yang digunakan oleh *device* tersebut. Pada tahun 1991, dibentuk sebuah proyek yang anggotanya dipelopori oleh Patrick Naughton, James Gosling, Mike Sheridan, dan Bill Joy. Pada awalnya, bahasa pemrograman java diberi nama "Oak" (Maiyana, 2018).

Java adalah bahasa pemrograman *multiplatform* dan *multidevice*. Hal ini memungkinan program yang dibuat dalam bahasa java dapat dijalankan hampir di semua komputer dan perangkat lain yang mendukung Java, dengan melakukan sedikit perubahan atau tidak sama sekali dalam kodenya. Aplikasi java dikomputasikan dan dijalankan dengan menggunakan *Java Virtual Machine*. Selain itu, aplikasi java ditulis menggunakan konsep OOP atau *Object Oriented Programming* (Irsan, 2015).

Java menyediakan berbagai *library* yang dapat digunakan oleh developer dalam mengembangkan aplikasi java. *Library* ini memudahkan seorang developer untuk memilih dan menggunakan *library* apa saja yang dibutuhkannya dalam proses pengembangan. Ketersediaan *library* java semakin beragam seiring waktu yang didukung oleh karya komunitas java (Irsan, 2015).

2.3.4 Web Server

Web server pertama kali diciptakan pada tahun 1980an dan menjadi tulang belakang dari *world wide web* (www). Seorang pengguna (*client*) yang menggunakan *browser* seperti Mozilla Firebox, Google Chrome, dan program *browser* lainnya, akan mengirim sebuah permintaan kepada web server. Web server menerima permintaan dari client melalui *browser* dan memproses permintaan yang kemudian memberikan hasil dari pemrosesan permintaan berupa data yang dikirimkan kembali kepada *browser* (Nurmiati, 2012).

Data yang dikirim oleh web server memiliki sebuah standarisasi format berupa SGML (*Standar General Markup Language*). *Browser* yang menerima data format ini akan menampilkan sesuai dengan kemampuan dan fitur *browser* tesebut. *Browser* yang hanya memiliki kemampuan untuk menampilkan teks, tidak dapat menampilkan data kiriman gambar yang diterima. Jika tersedia, *browser* berbasis teks ini akan menampilkan sebuah teks alternatif dari data gambar yang diterima (Nurmiati, 2012).

Web server berkomunikasi dengan client melalui *browser* mempunyai sebuah protokol tersendiri yang dinamakan HTTP (*Hypertext Transfer Protocol*). Protokol ini memungkinkan terjadinya komunikasi yang lebih mudah dimengerti diantara web server dengan *client*. Seperti yang telah dijelaskan sebelumnya, *world wide web* menggunakan format data SGML. Pada saat ini, format data HTML (*Hypertext Markup Language*) lebih digemari oleh pengguna internet dikarenakan kesederhanaan dan kemudahan dalam mempelajari (Nurmiati, 2012).

Penetapan standarisasi dari web server dikeluarkan oleh berbagai organisasi seperti W3C (*World Wide Web Consortium*), IETF (*Internet Engineering Task Force*), dan beberapa organisasi lainnya. Sampai saat ini, sudah lebih dari 110 spesifikasi yang telah dikeluarkan oleh W3C (W3C *Recommendations*). Berikut adalah standarasi web server antara lain (Nurmiati, 2012):

1. Spesifikasi HTML, CSS, DOM dan XHTML (W3C).
2. Spesifikasi Javascript (ECMA).

3. URL, HTTP (IETF) dalam bentuk dokumen RFC.

2.3.5 Web *Service*

Web *service* merupakan suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. Web *service* digunakan sebagai suatu fasilitas yang disediakan oleh suatu *website* untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan-layanan (*service*) yang disediakan oleh suatu *site* yang menyediakan web *service*. Web *service* menyimpan data informasi dalam format pengiriman seperti XML ataupun JSON sehingga data ini dapat diakses oleh sistem lain walaupun berbeda platform, sistem operasi, maupun bahasa *compiler*. Web *service* bertujuan untuk meningkatan kolaborasi antar pemrograman dan perusahaan, yang memungkinkan sebuah fungsi di dalam web *service* dapat dipinjam oleh aplikasi lain tanpa perlu mengetahui detail pemrograman yang terdapat di dalamnya. Beberapa alasan mengapa digunakannya web *service* adalah sebagai berikut:

1. Web *service* dapat digunakan untuk menstransformasikan satu atau beberapa bisnis *logic* atau *class* dan objek yang terpisah dalam satu ruang lingkup yang menjadi satu, sehingga tingkat keamanan dapat ditangani dengan baik.

2. Web *service* memiliki kemudahan dalam proses *deployment*-nya, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. Web *service* cukup diunggah ke web server dan siap diakses oleh pihak-pihak yang telah diberikan otoritasi.

3. Web *service* berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian web *service* tidak memerlukan konfigurasi khusus di sisi *firewall*.

2.3.6 PHP

Bahasa pemrograman adalah sebuah paket bahasa yang digunakan untuk membentuk sebuah bahasa turunan, bahasa turunan ini dapat berupa bahasa pemrograman, atau dapat juga berupa hasil akhir yang disebut dengan istilah aplikasi pemrograman. Menurut cara prosesnya, bahasa pemrograman dikategorikan menjadi bahasa *compiler* dan interpreter. PHP (*Hypertext Preprocessor*) adalah sebuah bahasa pemrograman berbentuk *scripting*, yang mana sistem kerja dari bahasa pemrograman ini menghasilkan sebuah program yang dijalankan sebagai interpreter bukan sebagai *compiler* (Nugroho, 2018).

Bahasa pemrograman *compiler* adalah bahasa yang akan mengubah *script* program ke dalam *source code*. *Source code* diubah menjadi bentuk *object code* yang menghasilkan *file* yang lebih kecil dari *file* mentah sebelumnya. *Object code* akan berubah menjadi sebuah program yang dapat dijalankan tanpa adanya program bantu pembuatnya. Hasil dari bahasa pemrograman *compiler* berbentuk sebuah program yang berstatus sebagai program .exe yang dapat dieksekusi tanpa adanya bantuan program pembuatnya (Nugroho, 2018).

Pada bahasa pemrograman interpreter, *script* yang telah ditulis pada saat menjalankannya tidak harus diubah ke dalam bentuk *source code*. Sehingga, pada saat menjalankan bentuk program kode dasar secara langsung akan dijalankan tanpa melalui proses pengubahan dalam bentuk *source code*. Ketika program memiliki kesalahan, program akan tetap berjalan tanpa harus memperhatikan kesalahan yang ada. Salah satu kekurangan dari bentuk bahasa pemrograman interpreter adalah program pembuatnya harus tersedia dan berjalan saat mengeksekusi program yang dibuat, sehingga hasil dari program ini bukan merupakan program yang dapat dieksekusi secara mandiri tanpa menggunakan program pembuatnya (Nugroho, 2018).

Seperti pada bahasa pemrograman lainnya, PHP memiliki beberapa aturan penulisan yang harus diketahui untuk memulai dan mengakhiri sebuah program PHP, sehingga dengan memenuhi beberapa aturan yang ada dan dapat mengerjakan program yang dihadapi. Untuk menuliskan dan memulai kode PHP,

harus memulainya dengan tanda <?php, setelah tanda tersebut dapat melanjutkan dengan kode program isi didalamnya. Untuk mengakhiri kode program yang dibuat, dapat menutupnya dengan tanda ?> (Nugroho, 2018).

2.3.7 RESTful API

REST (*Representational State Transfer*) adalah teknik di dalam arsitektur *software* yang memungkinkan untuk melakukan sistem terdistribusi pada *world wide web*. Teknik REST tidak memerlukan parsing XML ataupun JSON dan *header* pesan dari penyedia layanan sehingga dapat mengurangi penggunaan *bandwidth*. RESTful Web API atau RESTful Web *Service* merupakan web *service* yang pada implementasinya menggunakan protokol http yang menerapkan prinsip-prinsip pada REST. *Service* yang dapat digunakan terhadap http yang menggunakan prinsip REST antara lain *method POST, GET, PUT, PATCH* atau *DELETE*. Untuk membuat sebuah RESTful Web API, tersedia berbagai *library online* ataupun *library framework* yang dapat digunakan seperti *Simple*-REST, *Sanctum,* dan *Passport* yang dapat disesuaikan dengan kebutuhan. Berikut ini adalah penjelasan tengan *method* pada REST (Sandoval, 2009):

1. *POST* : *request* yang digunakan untuk menambah atau membentuk sebuah data baru pada tabel *database*.
2. *GET* : *request* yang digunakan untuk mendapatkan kumpulan data dari *database*.
3. *PUT* : *request* yang digunakan untuk memperbaruhi seluruh data kolom pada baris tabel *database* terpilih.
4. *PATCH* : *request* yang digunakan untuk memperbaruhi sebagian data kolom pada baris tabel *database* terpilih.
5. *DELETE*: *request* yang digunakan untuk menghapus data baris pada *database*.

Setiap *request* yang dilakukan, RESTful Web API akan memberikan sebuah kode respon dan pesan balik dalam bentuk format yang disepakati seperti XML

ataupun JSON. Seorang pengembang dapat menentukan kode responnya secara mandiri sesuai kebutuhan. Respon kode yang umum digunakan seperti 200 untuk proses berjalan tanpa masalah, 201 data berhasil tercipta, 204 request tidak berisi data, 404 untuk data tidak ditemukan, 405 untuk *method* tidak diizinkan, dan 409 terjadi sebuah konflik data.

2.3.8 JSON

JSON (JavaScript *Object Notation*) adalah format pertukaran data yang ringan, mudah untuk dibaca dan ditulis serta mudah diterjemahkan oleh mesin. JSON adalah format teks yang benar-benar independen namun menggunakan konvensi yang familiar bagi pemrogram yang termasuk dalam keluarga Bahasa C, termasuk C, C ++, C #, Java, JavaScript, Perl, dan Python. Oleh karena sifat tersebut menjadikan JSON sebagai bahasa pertukaran data yang ideal. JSON terbuat dari dua struktur utama, yaitu kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*structure*), kamus (*dictionary*), tabel *hash* (*hash table*), daftar berkunci (*keyed list*), *associative array*, dan daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*). Struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data ini.

2.3.9 MySQL

MySQL merupakan *database* server yang memiliki kemampuan untuk memanajemen *database* dengan baik. MySQL terhitung merupakan *database* yang paling digemari dan paling banyak digunakan jika dibandingan dengan *database* lainnya. Selain MySQL, terdapat berbagai jenis *database* server yang memiliki kemampuyan yang sama seperti Oracle dan PostgreSQL. Di dalam dunia internet, MySQL dijadikan sebagai sebuah *database* yang paling banyak digunakan selain *database* yang bersifat *shareware* seperti *Microsoft Access*.

Penggunaan MySQL dapat dipadukan dengan menggunakan program aplikasi PHP. Dengan menggunakan PHP dan MySQL memungkinkan sebuah program untuk dapat menangani permintaan data (Nugroho, 2018).

MySQL memiliki *query* yang distandarkan oleh ANSI/ISO yaitu menggunakan bahasa SQL sebagai bahasa permintaannya, hal tersebut juga telah dimiliki oleh bentuk bentuk *database* server seperti Oracle, PostgreSQL, MSQL, SQL Server maupun bentuk-bentuk *database* yang berjalan yang berjalan pada mode grafis (sifatnya visual) seperti *Interbase* yang diproduksi oleh Borland. Selain itu, MySQL memiliki kemampuan untuk mendukung *Relational Database Manajement System* (RDBMS). Dengan kemampuan RDMS, MySQL mampu menangani data-data sebuah perusaaan dalam sekala besar hingga *Giga Byte* (Nugroho, 2018).

2.3.10 Laravel

Laravel adalah *framework* bahasa pemrograman *Hypertext Preprocessor* (PHP) yang ditujukan untuk pengembangan aplikasi berbasis web dengan menerapkan konsep *Model View Controller* (MVC). *Framework* ini dibuat oleh Taylor Otwell dan pertama kali dirilis pada tanggal 9 Juni 2011. Laravel berlisensi *open source* yang artinya bebas digunakan tanpa harus melakukan pembayaran. Alamat *website* resmi dari *framework* Laravel adalah https://laravel.com. Fitur-fitur modern Laravel yang sangat membantu developer dalam membuat aplikasi adalah *Bundles*, *Eloquent* ORM (*Object-Relational Mapping*), *Query Builder*, *Application Logic*, *Reverse Routing*, *Resource Controller*, *Class Auto Loading*, *View Composers*, *Blade*, IoC *Containers*, *Migration*, *Database Seeding*, *Unit Testing*, *Automatic Pagination*, *Form request*, dan *Middleware*.

2.3.11 Bootstrap

Bootstrap merupakan *framework* ataupun *tools* untuk membuat aplikasi web ataupun situs web *responsive* secara cepat, mudah, dan gratis. Bootstrap terdiri dari CSS dan HTML untuk menghasilkan *Grid*, *Layout*, *Typography*, *Table*, *Form*, *Navigation*, dan lain-lain. Di dalam Bootstrap juga sudah terdapat jQuery *plugins*

untuk menghasilkan komponen UI yang cantik seperti *Transitions*, *Modal*, *Dropdown*, *Scrollspy*, *Tooltip*, *Tab*, *Popover*, *Alert*, *Button*, *Carousel*, dan lain-lain. Dengan bantuan Bootstrap, kita bisa membuat *responsive website* dengan cepat dan mudah dan dapat berjalan sempurna pada *browser-browser* populer seperti Chrome, Firefox, Safari, Opera, dan Internet Explorer.

Bootstrap diciptakan oleh dua orang *programmer* di twitter, yaitu Mark Otto dan Jacob Thornton pada tahun 2011. Pada saat itu para *programmer* di Twitter menggunakan berbagai macam *tool* dan *library* yang mereka kenal dan suka untuk melaksanakan pekerjaan mereka, sehingga tidak ada standarisasi dan akibatnya sulit untuk dikelola. Mark Otto dan Jacob Thornton tergerak untuk menciptakan satu *tool* ataupun *framework* yang dapat digunakan bersama di lingkungan internal Twitter. Oleh karena faktor historis tersebut, walaupun nama resminya hanyalah Bootstrap, namun terkenal di kalangan *developer* sebagai Twitter Bootstrap. Sejak diluncurkan pada bulan Agustus 2011, Bootstrap telah berevolusi dari sebuah proyek yang hanya berbasis CSS menjadi sebuah *tool* ataupun *framework* yang lebih lengkap yang juga berisi Javascript *Plugin*, *Icon*, *Forms*, dan *Button*.

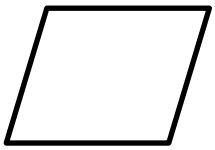
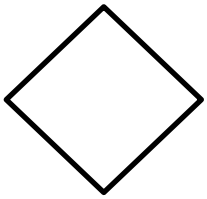
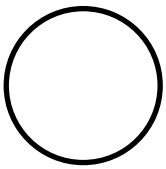
2.4 Metode dan Alat Bantu Pengembangan Sistem

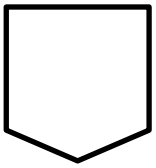
2.4.1 *Flowchart*

Flowchart adalah simbol-simbol yang menggambarkan sebuah tahapan proses suatu sistem ataupun urutan-urutan instruksi dari suatu program komputer. Oleh karena itu *flowchart* yang dihasilkan dapat bervariasi antara suatu program dengan yang lainnya (Suyanto, 2004).

Flowchart menurut Jogiyanto (2005) adalah bagan yang menunjukkan alir di dalam program atau prosedur sistem secara logika yang dapat didefinisikan sebagai bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. Setiap pengolahan dalam *flowchart* terbagi atas 3 bagian utama, yaitu *input*, proses pengolahan, dan *output*. Secara lengkap bagian-bagian *flowchart* ditunjukkan pada **Tabel 2.1** Simbol-Simbol *Flowchart*.

Tabel 2. 1 Simbol-Simbol *Flowchart*

Simbol	**Nama**	**Fungsi**
	*Terminator*	Permulaan / akhir program.
	Garis Alir (*Flow Line*)	Arah aliran program.
	*Preparation*	Proses inisialisasi / pemberian nilai awal.
	Proses	Proses perhitungan / proses pengolahan data.
	*Input* / *Output* Data	Proses *input* / *output* data, parameter, informasi.
	*Decision*	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya.
	*On Page Connector*	Penghubung bagian-bagian *flowchart* yang berada pada satu halaman.

Simbol	Nama	Fungsi
	*Off Page Connector*	Penghubung bagian-bagian *flowchart* yang berada pada halaman berbeda.

2.4.2 *System Development Life Cycle* (SDLC)

Pada awal pengembangan sebuah perangkat lunak, seorang *programmer* dapat saja langsung melakukan pengodean perangkat lunak tanpa menggunakan prosedur atau tahapan perangkat lunak. Melakukan pengembangan perangkat lunak tanpa menggunakan prosedur atau mengikuti tahapan perangkat lunak, besar kemungkinan akan ditemukannya kendala-kendala seiring dengan perkembangan skala sistem perangkat yang semakin besar. Pada tahun 1960-an, dibuatlah sebuah tahapan untuk mengembangkan sistem skala usaha besar secara fungsional yang diutamakan untuk golongan konglomerat di zamannya (Rosa & Shalahuddin, 2019).

SDLC atau *Software Development Life Cycle* atau sering disebut juga *System Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metedologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan *best practice* atau cara-cara yang sudah teruji baik). Seperti halnya proses metamorfosis pada kupu-kupu, untuk menjadi kupu-kupu yang indah maka dibutuhkan beberapa tahap untuk dilalui, sama halnya dengan membuat perangkat lunak, memiliki daur tahapan yang dilalui agar menghasilkan perangkat lunak yang berkualitas (Rosa & Shalahuddin, 2019).

Tahapan-tahapan yang ada pada SDLC secara global adalah sebagai berikut:

a. Inisiasi (*initiation*)

Tahap ini biasanya ditandai dengan pembuatan proposal proyek perangkat lunak.

b. Pengembangan konsep sistem (*system concept development*)

Mendefinisikan lingkup konsep termasuk dokumen lingkup sistem, analisis manfaat biaya, manajemen rencana, dan pembelajaran kemudahan sistem.

c. Perencanaan (*planning*)

Mengembangkan rencana manajemen proyek dan dokumen perencanaan lainnya. Menyediakan dasar untuk mendapatkan sumber data (*resources*) yang dibutuhkan untuk memperoleh solusi.

d. Analisis kebutuhan (*requirement analysis*)

Menganalisis kebutuhan pemakai sistem perangkat lunak (*user*) dan mengembangkan kebutuhan *user*. Membuat dokumen kebutuhan fungsional.

e. Desain (*design*)

Mentransofrmasikan kebutuhan detail menjadi kebutuhan yang sudah lengkap, dokumen desain sistem fokus pada bagaimana dapat memenuhi fungsi-fungsi yang dibutuhkan.

f. Pengembangan (*development*)

Mengonversi desain ke sistem informasi yang lengkap termasuk bagaimana memperoleh dan melakukan instalasi lingkungan sistem yang dibutuhkan, membuat basis data dan mempersiapkan prosedur kasus pengujian, mempersiapkan berkas *file* pengujian, pengodean, pengompilasian, memperbaiki, dan membersihkan program, peninjauan pengujian.

g. Integrasi dan pengujian (*integration and test*)

Mendemontrasikan sistem perangkat lunak bahwa telah memenuhi kebutuhan yang dispesifikasikan pada dokumen kebutuhan fungsional. Dengan diarahkan oleh staf penjamin kualitas (*quality assurance*) dan *user*. Menghasilkan laporan analisis pengujian.

h. Implementasi (*implementation*)

Termasuk pada persiapan implementasi, implementasi perangkat lunak pada lingkungan produksi (lingkungan pada *user*) dan menjalankan resolusi dari permasalahan yang terindentifikasi dari fase integrasi dan pengujian.

i. Operasi dan pemeliharaan (*operations and maintenance*)

Mendeskripsikan pekerjaan untuk mengoprasikan dan memelihara sistem informasi pada lingkungan produksi (lingkungan pada *user*), termasuk implementasi akhir dan masuk pada proses peninjauan.

j. Disposisi (*disposition*)

Mendeskripsikan aktivitas akhir dari pengembangan sistem dan membangun data yang sebenarnya sesuai dengan aktivitas *user*.

2.4.2.1 *Waterfall*

Model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial liniear (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*) (Rosa & Shalahuddin, 2019).

Adapun tahapan dalam metode *waterfall* yaitu:

1. Analisis

Analisis merupakan suatu kegiatan yang dimulai dari proses awal di dalam mempelajari sesuatu serta mengevaluasi suatu bentuk permasalahan yang ada.

2. Desain

Desain merupakan kegiatan dalam penggambaran, perencanaan, dan perancangan atau pengaturan dari beberapa elemen yang terpisah di dalam sistem menjadi kesatuan dan berfungsi dengan baik.

3. Pengodean

Pengodean merupakan upaya dalam mengimplementasikan desain menjadi perangkat lunak.

4. Pengujian

Pengujian merupakan upaya dalam menelusuri lebih lanjut terhadap perangkat lunak yang telah dibuat untuk mendapatkan informasi menyenai kualitas perangkat lunak yang sedang diuji.

5. Pemeliharaan

Pemeliharaan merupakan kegiatan yang dilakukan dalam perawatan dan pengubahan atau pengembangan dari perangkat lunak yang telah dibuat dan tidak terdeteksi saat pengujian.

2.4.3 *Unified Modeling Language* (UML)

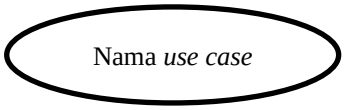
Mengikuti perkembangan teknologi perangkat lunak, diperlukannya sebuah bahasa yang digunakan untuk pemodelan dari suatu perangkat lunak saat akan dibuat. Berdasarkan alasan ini, dibuatlah sebuah standarisasi dengan tujuan agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Pada perkembangan berorientasi objek, dibuat sebuah standarisasi bahasa pemodelan

dalam pembangunan perangkat lunak dengan menggunakan teknik pemrograman berorientasi objek yang dinamakan *Unified Modeling Language* (UML). Kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan mendokumentasi perangkat lunak menjadi alasan dibuatnya standarisasi pemodelan UML. UML adalah bahasa visual untuk komunikasi dan pemodelan tentang sebuah sistem dengan menggunakan diagram dan teks-teks pendukung (Rosa & Shalahuddin, 2019).

2.4.3.1 *Use Case Diagram*

Use case diagram atau diagram *use case* adalah sebuah pemodelan untuk menggambarkan kelakuan (*behavior*) dari sistem informasi yang akan dibuat. Dengan menggunakan *use case*, seorang perancang aplikasi dapat mendeskripsikan interaksi apa saja yang dapat dilakukan masing-masing aktor di dalam sistem. Secara kasar, *use case* digunakan untuk memperlihatkan aktivitas apa saja yang dapat dilakukan masing-masing aktor terhadap sistem (Rosa & Shalahuddin, 2019). Berikut ini simbol-simbol notasi pada use case diagram dapat dilihat pada **Tabel 2.2** Simbol Notasi *Use Case*:

Tabel 2. 2 Simbol Notasi *Use Case*

No	Gambar	Nama	Keterangan
1		*Use Case*	Fungsionalitas yang disediakan sistem sebagai unti-unit yang saling bertukar pesan antara unit dan aktor.
2		Aktor / *Actor*	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi.
3		Asosiasi / *Association*	Komunikasi antar aktor dan *use case* yang

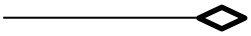
No	Gambar	Nama	Keterangan
			berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4		Ekstensi / *Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
5		Generalisasi / *Generalization*	Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6		Menggunakan / *Include* / *Uses*	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.

2.4.3.2 *Class Diagram*

Class Diagram atau diagram kelas adalah diagram yang menggambarkan struktur aplikasi dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun aplikasi. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Diagram kelas dibuat agar pembuat program membuat kelas-kelas sesuai sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron (Rosa & Shalahuddin, 2019). Berikut ini simbol-simbol notasi pada *class diagram* dapat dilihat pada **Tabel 2. 3** Simbol Notasi *Class Diagram*:

Tabel 2. 3 Simbol Notasi *Class Diagram*

No	Gambar	Nama	Keterangan
1	nama_kelas +atribut +operasi()	*Class*	Kelas pada struktur sistem.
2		*Depedency*	Relasi antar kelas dengan makna perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
3		*Association*	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4		*Directed Association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
5		*Generalization*	Relasi antar kelas dengan makna objek anak berbagi perilaku dan struktur data dari objek yang ada

No	Gambar	Nama	Keterangan
			di atasnya objek induk (*ancestor*).
6		*Aggregation*	Relasi antar kelas dengan makna semua-bagian.

2.4.3.3 *Sequence Diagram*

Sequence diagram atau diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *massage* yang dikirimkan dan diterima antar objek (Rosa & Shalahuddin, 2019). Berikut ini simbol-simbol notasi pada *sequence diagram* dapat dilihat pada **Tabel 2. 4** Simbol Notasi *Sequence Diagram*:

Tabel 2. 4 Simbol Notasi *Sequence Diagram*

No	Gambar	Nama	Keterangan
1		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
2		Garis Hidup / *Lifeline*	Menyatakan kehidupan suatu objek.
3	nama objek : nama kelas	Objek	menyatakan objek yang berinteraksi pesan

No	Gambar	Nama	Keterangan
4		Waktu Aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya.
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 *Return*	Menyatakan suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian

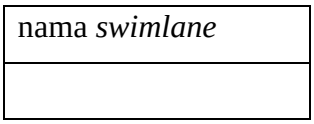
No	Gambar	Nama	Keterangan
9		Pesan Tipe *Destroy*	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaliknya jika ada *create* maka ada *destroy*.

2.4.3.4 *Activity Diagram*

Activity Diagram atau diagram aktivitas menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak (Rosa & Shalahuddin, 2019). Berikut ini simbol-simbol notasi pada *activity diagram* dapat dilihat pada **Tabel 2. 5** Simbol Notasi *Activity Diagram*:

Tabel 2. 5 Simbol Notasi *Activity Diagram*

No	Gambar	Nama	Keterangan
1		Status Awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2		Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3		Percabangan / *Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4		Penggabungan / *Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.

No	Gambar	Nama	Keterangan
5		Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
6		*Swimlane*	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.4.4 *Entity Relationship Diagram* (ERD)

Dalam pemodelan sebuah basis data, *Entity Relationship Diagram* (ERD) menjadi pilihan yang paling banyak digunakan untuk memodelkan sebuah basis data. ERD adalah sebuah pemodelan basis data yang dikembangkan berdasarkan teori himpunan dalam bidang matematika dan menggunakan pemodelan basis data relasional. ERD memiliki beberapa aliran notasi seperti notasi Chen (dikembangkan oleh Pater Chen), Barker (dikembangkan oleh Richard Barker, Ian Palmer, Harry Ellis), notasi Crow's Foot, dan beberapa notasi lain (Rosa & Shalahuddin, 2019). ERD memiliki derajat relasi atau kardinalitas rasio, yaitu meliputi:

a. *One to one* (1:1)

Setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, begitu pula sebaliknya.

b. *One to many* (1:M / *Many*)

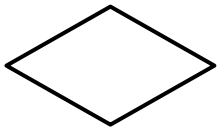
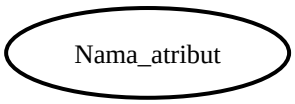
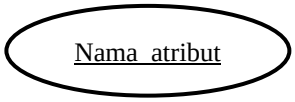
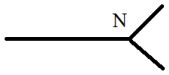
Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.

c. *Many* to *many* (M:M)

Setiap entitas A dapat berhubungan dengan banyak entitas himpunan entitas B dan demikian pula sebaliknya.

Berikut ini simbol-simbol notasi pada *entity relationship diagram* dapat dilihat pada **Tabel 2. 6** Simbol Notasi *Entity Relationship Diagram*:

Tabel 2. 6 Simbol Notasi *Entity Relationship Diagram*

No	**Simbol ERD**	**Nama**	**Keterangan**
1		Entitas	Memberikan identitas yang mewakili suatu objek dan dapat dibedakan dengan objek yang lain.
2		Relasi	Menunjukkan hubungan diantara sejumlah entitas yang berbeda.
3		Atribut	Properti yang dimiliki oleh suatu entitas, dimana dapat mendeskripsikan karakteristik dari entitas tersebut.
4		Atribut kunci primer	Properti kunci primer atau bersifat unik pada suatu entitas.
5		Alur	Garis yang memiliki fungsi untuk menghubungkan atribut dengan entitas dan entitas dengan relasi.
6		Asosiasi	Penghubung antara relasi dan entitas di mana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian.

2.4.5 Kamus Data (*Data Dictionary*)

Kamus data merupakan kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak. Pada Kamus data, masukan (*input*) dan keluaran (*output*) dapat dipahami karena memiliki standarisasi cara penulisan. Pada implementasi program, kamus data dapat menjadi parameter masukan atau keluaran dari rancangan fungsi atau prosedur (Rosa & Shalahuddin, 2019). Kamus data biasanya berisi:

- Nama - nama dari data.
- Digunakan pada – merupakan proses proses yang terkait data.
- Deskripsi – merupakan deskripsi data.
- Informasi tambahan – seperti tipe data, nilai data, batas nilai data, dan komponen yang membentuk data.

2.5 Pengujian Perangkat Lunak

Pengujian perangkat lunak adalah sebuah elemen sebuah topik yang memiliki cakupan luas dan sering dikaitkan dengan verifikasi (*verification*) dan validasi (*validation*) (V&V). Verifikasi mengacu pada sekumpulan aktivitas yang menjamin bahwa perangkat lunak mengimplementasi fungsi yang benar dan spesifik. Validasi mengacu pada sekumpulan aktivitas yang berbeda yang menjamin bahwa perangkat lunak yang dibangun dapat ditelusuri sesuai dengan kebutuhan pelanggan (*customer*) (Rosa & Shalahuddin, 2019).

- Verifikasi: “Apakah produk dibangun dengan benar?” (lebih ke arah apakah proses pengembangan produk sudah benar dan telah berhasil mengimplementasikan fungsi yang benar)
- Validasi: “Apakah sudah membangun produk yang benar?” (lebih ke arah hasil produk apakah sudah sesuai dengan yang diinginkan)

2.5.1 *Blackbox Testing*

Blackbox testing atau pengujian *blackbox* yaitu metode uji coba yang menguji 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 (Rosa & Shalahuddin, 2019).

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses *login* maka kasus uji yang dibuat adalah (Rosa & Shalahuddin, 2019):

- Jika user memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
- Jika user memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah.

2.5.2 *User Acceptance Testing* (UAT)

User Acceptance Testing merupakan pengujian yang dilakukan oleh *end-user* dimana *user* tersebut adalah staf/karyawan perusahaan yang langsung berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya (Perry, 2006).

Setelah dilakukan *system testing*, *acceptance testing* menyatakan bahwa sistem *software* memenuhi persyaratan. *Acceptance testing* merupakan pengujian yang dilakukan oleh pengguna yang menggunakan teknik pengujian *blackbox* untuk menguji sistem terhadap spesifikasinya. Pengguna akhir bertanggung jawab untuk memastikan semua fungsionalitas yang relevan telah diuji (Lewis, 2009).

Berikut adalah rumus menghitung skor pengujian *user acceptance testing* (UAT), yaitu:

Skor = Jumlah Jawaban / (Jumlah Pertanyaan x Jumlah Responden) x 100%

Keterangan:

Jumlah jawaban = Jumlah jawaban responden ya/ tidak

Jumlah pertanyaan = Jumlah pertanyaan yang diajukan kepada responden

Jumlah responden = Jumlah responden yang mengisi kuisioner

Penskalaan yang digunakan untuk mengukur tingkat keberhasilan aplikasi yang dibangun adalah teknik penskalaan *Likert's Summated Rating* (LSR). *Likert's Summated Rating* adalah skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang tentang suatu gejala atau fenomena pendidikan (Suwandi, 2019). Interpretasi skor user acceptance test dengan perhitungan penskalaan *Likert's Summated Rating* (LSR) dapat dilihat pada **Tabel 2. 7** di bawah.

Tabel 2. 7 Interpretasi Skor *User Acceptance Testing*

Interpretasi	**Persentase Skor**	**Nilai**
Sangat Buruk	0%-19,99%	1
Buruk	20%-39,99%	2
Cukup	40%-59,99%	3
Baik	60%-79,99%	4
Sangat Baik	80%-100%	5