BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian yang bertujuan untuk mengembangkan web berbasis SLiMS sudah banyak diterapkan di berbagai perpustakaan di Indonesia. Pada penelitian yang sudah dilakukan sebelumnya menunjukkan bahwa SLiMS memiliki fitur *plugin* yang memungkinkan para pengembang untuk menambahkan fitur pada SLiMS tanpa mengubah kode sumber yang sudah ada.

Pada penelitian yang berjudul "Model Pengembangan *Plug-In* SLiMS pada Komunitas SLiMS Kudus" yang dilakukan oleh Zaid Abdurrahman dan Thoriq Tri Prabowo menjelaskan tentang peran komunitas SLiMS Kudus pada pengembangan *plugin* SLiMS. Dalam tulisannya, Zaid dan Thoriq menggunakan model pengembangan *open-source* [5] untuk membantu mereka dalam mendeskripsikan pengembangan *plugin* SLiMS. Metode penelitian yang mereka gunakan adalah metode penelitian kualitatif. Sementara itu, teknik pengumpulan data yang mereka gunakan antara lain, teknik observasi terus terang, wawancara tidak terstruktur, dan dokumentasi. Untuk metode analisis data, mereka menggunakan model Miles and Huberman. Hasil dari penelitian ini berupa informasi tentang peranan komunitas SLiMS Kudus sebagai sumber inspirasi dan wadah pengembangan *plugin* SLiMS. Kesimpulan dari penelitian ini adalah komunitas SLiMS Kudus berperan sebagai sumber inspirasi dan wadah dalam penyebaran *plugin* SLiMS [2].

Pada penelitian kedua yang berjudul "Pengembangan Sistem Informasi Perpustakaan Berbasis SLiMS Akasia 8" yang dilakukan oleh Haidir Rahman dkk bertujuan untuk mengembangkan sistem informasi perpustakaan SMK Negeri 2 Banjarmasin. Sistem informasi yang dikembangkan menggunakan SLiMS Akasia 8. Sistem informasi perpustakaan yang dulunya menggunakan metode konvensional dikembangkan ke metode digital. Tujuan penelitian adalah yaitu

prosedur pengembangan sistem informasi perpustakaan berbasis SLiMS Akasia 8 dan sebagai pengembangan sistem informasi perpustakaan. Penelitian ini menggunakan jenis penelitian *research and development* (R&D) dengan menggunakan model pengembangan Alessi & Trollip yang sudah dimodifikasi. Hasil penelitian dan pengembangan sistem informasi perpustakaan ini dinyatakan layak setelah mendapat validasi dari ahli sistem dan ahli perpustakaa, maka direkomendasikan pengembangan ini agar bisa diterapkan ke lembaga pendidikan lainnya yang mempunyai sistem informasi perpustakaan untuk mengevaluasi lebih dalam lagi [3].

Pada penelitian ketiga yang berjudul "Senayan Library Management System (SLiMS) *Library Application Management in* Madrasah Tsanawiyah" bertujuan untuk menjelaskan pengelolaan aplikasi SLiMS di perpustakaan Madrasah Tsanawiyah Banyumas. Metode yang digunakan pada penelitian ini adalah metode penelitian kualitatif dengan pendekatan deskriptif. Hasil dari penelitian ini menunjukkan bahwa penggunaan SLiMS sebagai aplikasi untuk memasukkan data buku perpustakaan, mengelola, dan mengendalikannya memberikan sejumlah manfaat signifikan. Dengan mempercepat proses di perpustakaan, meningkatkan kinerja pustakawan, dan meningkatkan produktivitas kerja, aplikasi SLiMS membantu mencapai efektivitas kerja yang optimal. Pustakawan dapat secara efektif dan efisien melaksanakan berbagai tugas, mulai dari persiapan, pelayanan, hingga pengelolaan dan pengendalian perpustakaan.

Pada penelitian ketiga yang berjudul "Virtualisasi Katalog Senayan Library Management System (SLiMS) Berbasis 3D" bertujuan untuk menghasilkan aplikasi virtualisasi katalog SLiMS untuk penataan buku di rak dan pencarian buku dalam bentuk 3D serta mengetahui keakuratan dan performa. Metode dilakukan diawali membuat struktur *vertex* dari buku dan rak dengan menggunakan data polihedron dari objek kubus. Data polihedron dapat dibaca dengan aplikasi Blender 3D, kemudian penulis melakukan analisis untuk mendapatkan rumus pembentuk polihedron serta mendapatkan urutan indeks dari *vertex*. Data masukkan yang digunakan untuk membentuk polihedron buku adalah panjang, lebar dan tebal buku,

sedangkan untuk polihedron rak adalah panjang, lebar, tinggi dan tebal papan. Data masukkan disimpan di basis data agar dapat diakses dan dikonversi menjadi 3D dengan pemrograman web. Hasil penelitian ini merupakan aplikasi virtualisasi katalog SLiMS untuk penataan buku di rak dan pencarian buku dalam bentuk 3D. [6].

Pada penelitian keempat yang berjudul "*E-Library* Interaktif dengan SLiMS Bulian Menggunakan Metode RAD" bertujuan untuk merancang dan membangun perpustakaan digital berbasis web mengunakan SLiMS menjadi lebih interaktif dengan menambahkan sarana forum diskusi. Metode yang digunakan untuk pengumpulan data yaitu observasi ke perpustakaan sekolah, wawancara pihak pengelola perpustakaan, dan studi literatur untuk memperdalam pemahaman masalah. Adapun metode pengembangan perangkat lunak yang digunakan adalah *Rapid Application Development (RAD)* dengan memanfaatkan *Unified Modelling Language* (UML) untuk menggambarkan rancangan sistem yang dikembangkan. Metode pengujian yang digunakan adalah pengujian *black-box* dan *System Usability Scale* (SUS). Hasil dari penelitian ini berupa *e-library* yang dilengkapi dengan fitur forum sehingga dapat mendorong dan budaya literasi bagi siswa. [7].

Pada penelittian kelima yang berjudul "Implementasi SLiMS di Perpustakaan Perguruan Tinggi" bertujuan untuk mengetahui kendala, tantangan, dan manfaat dalam implementasi SLIMS di perpustakaan perguruan tinggi. Selain itu dalam penelitian ini juga dideskripsikan kendala dan tantangan serta manfaat yang diperoleh melalui implementasi SLiMS di perpustakaan perguruan tinggi. Penelitian ini menggunakan metode *Systematic Literature Review* (SLR) dengan pendekatan kualitatif. Adapun metode pengumpulan dan analisis data yang digunakan terdiri dari beberapa tahapan seperti tahap perencanaan, pelaksanaan, dan mensintesis terhadap hasil temuan pencarian. Temuan pada penelitian ini menunjukkan bahwa terdapat enam belas artikel jurnal terseleksi yang mengulas terkait implementasi SLiMS di perpustakaan perguruan tinggi, yang terdiri dari sepuluh artikel penelitian di perpustakaan universitas, tiga penelitian di perpustakaan sekolah tinggi, satu penelitian di perpustakaan politeknik dan dua

penelitian di perpustakaan institut. Setelah melakukan tahap tinjauan literatur terdapat dua artikel jurnal terseleksi yang mengulas sistem SLiMS sebagai kendala atau tantangan serta terdapat dua belas artikel jurnal terseleksi yang mendeskripsikan manfaat implementasi SLIMS di perpustakaan yang terkait dengan pelayanan perpustakaan. Adapun kendala yang dihadapi yaitu terkait dengan sistem SLIMS, sumber daya manusia dan fasilitas. Kemudian manfaat dari implementasi SLIMS di perpustakaan perguruan tinggi diantaranya yaitu terkait pelayanan perpustakaan, pengolahan bahan pustaka dan pengembangan koleksi perpustakaan. Hasil dari penelitian ini berupa kesimpulan bahwa penelitian terkait implementasi SLiMS di perpustakaan dilakukan di empat jenis perpustakaan perguruan tinggi seperti perpustakaan universitas, perpustakaan sekolah tinggi, perpustakaan politeknik dan perpustakaan institut. [8].

Tabel 2.1 Perbandingan Penelitian-Penelitian Terkait

No.	Peneliti	Judul	Metode	Hasil	Perbedaan	Persamaan
1.	Zaid Abdurrahman dan Thoriq Tri Prabowo	Model Pengembangan Plug-In SLiMS pada Komunitas SLiMS Kudus (2021)	Kualitatif	Informasi tentang peranan komunitas SLiMS Kudus sebagai sumber inspirasi dan wadah pengembangan *plugin* SLiMS.	- Metode penelitian yang digunakan - Hasil yang disimpulkan	- Membahas mengenai pengembangan *plugin* SLiMS.
2.	Haidir Rahman, Hamsi Mansur, dan Adrie Satrio	Pengembangan Sistem Informasi Perpustakaan Berbasis SLiMS Akasia 8 (2021)	Penelitian dan Pengembangan	Sistem informasi perpustakaan yang sudah divalidasi oleh ahli sistem dan ahli perpustakaan.	- Membahas pengembangan SLiMS secara umum. Tidak membahas pengembangan *plugin* SLiMS.	- Metode penelitian yang digunakan. - Membahas mengenai pengembangan sistem informasi perpustakaan berbasis SLiMS.
3.	Siswadi	Senayan Library Management System (SLiMS) Library Application Management in Madrasah Tsanawiyah	Kualitatif dengan pendekatan deskriptif	penggunaan SLiMS sebagai aplikasi untuk memasukkan data buku perpustakaan, mengelola, dan mengendalikannya memberikan sejumlah manfaat signifikan.	- Metode penelitian yang digunakan. - Tidak secara khusus membahas pengembangan *plugin* SLiMS.	- Membahas mengenai pengembangan sistem infromasi perpustakaan berbasis SLiMS.

No.	Peneliti	Judul	Metode	Hasil	Perbedaan	Persamaan
		(2023)				
4.	Arief Ichwani dan Muhammad Visal Bainuri	*E-Library* Interaktif dengan SLiMS Bulian Menggunakan Metode RAD (2022)	*Rapid Application Development* (RAD)	Menghasilkan *e-library* yang dilengkapi dengan fitur forum sehingga dapat mendorong dan budaya literasi bagi siswa.	- Metode penelitian yang digunakan. - Tidak secara khusus membahas pengembangan *plugin* SLiMS.	- Menambahkan fitur baru yang ada hubungannya dengan forum diskusi.

No.	Peneliti	Judul	Metode	Hasil	Perbedaan		Persamaan
5.	Iskandar dan Luki Wijayanti	Implementasi SLiMS di Perpustakaan Perguruan Tinggi (2022)	Oktober 2022	*Systematic Literature Review* (SLR) dengan pendekatan kualitatif	Berupa kesimpulan bahwa penelitian terkait implementasi SLiMS di perpustakaan dilakukan di empat jenis perpustakaan perguruan tinggi seperti perpustakaan universitas, perpustakaan sekolah tinggi, perpustakaan politeknik dan perpustakaan institut.	- Metode penelitian yang digunakan. - Tidak secara khusus membahas pengembangan *plugin* SLiMS.	- Membahas tentang implementasi SLiMS.

2.2 Landasan Teori

2.2.1 SLiMS

Senayan Library Management System (SLiMS) merupakan sistem atau aplikasi otomasi perpustakaan yang dicetuskan pertama kali pada tahun 2006 oleh Hendro Wicaksono dan Arie Nugrah [2]. SLiMS pertama kali digunakan di Perpustakaan Departemen Pendidikan Nasional [9]. Fungsi utama SLiMS yaitu sebagai pengelola sumber daya perpustakaan seperti buku, jurnal, dokumen digital, dan material perpustakaan lainnya. Selain itu, SLiMS juga dapat mengelola administrasi perpustakaan seperti sirkulasi koleksi, manajemen koleksi, keanggotaan, penghitungan stok, dan lain-lain [10].

SLiMS bersifat *open source software* berbasis web [11]. SLiMS mampu berjalan sempurna di dalam sistem jaringan lokal (intranet) maupun internet. Oleh karena itu, pemustaka dapat menelusuri katalog perpustakaan dari mana saja dan kapan saja melalui web perpustakaan yang disediakan [12].

2.2.2 PHP

Hypertext Preprocessor (PHP) merupakan salah satu bahasa pemograman yang berjalan dalam sebuah *web server* dan berfungsi sebagai pengolah data pada sebuah *server*. Data yang dikirim oleh klien akan diolah dan disimpan pada basis data *web server* dan dapat ditampilkan kembali apabila diakses. Untuk menjalankan kode-kode program PHP, berkas harus diunggah ke dalam *server*. Unggah adalah proses mentransfer data atau file dari komputer klien ke dalam web server.

Untuk membuat web yang dinamis dan mudah diperbaharui setiap saat dari *browser*, dibutuhkan sebuah program yang mampu mengolah data dari komputer klien atau dari komputer *server* itu sendiri sehingga mudah dan nyaman disajikan di *browser*. Salah satu program yang dapat dijalankan di *server* dan cukup andal adalah PHP.

PHP bekerja di dalam sebuah dokumen *Hypertext Markup Language* (HTML) untuk dapat menghasilkan isi dari sebuah halaman web sesuai permintaan. PHP memungkinkan pengguna agar dapat merubah situs menjadi sebuah aplikasi

berbasis web, tidak lagi hanya sekedar sekumpulan halaman statis yang jarang diperbaharui.

Pada awalnya, PHP dirancang untuk diintegrasikan dengan *web server* Apache. Namun belakangan ini, PHP juga dapat bekerja dengan *web server* seperti *Personal Web Server*, *Internet Information Server* dan Xitami. Yang membedakan PHP dengan bahasa pemrograman lain adalah adanya tag penentu, yaitu diawali dengan "<?" atau "<?php" dan diakhiri dengan "?>". Sehingga pengembang bebas menempatkan skrip PHP dimanapun dalam dokumen HTML yang telah dibuat [13].

Selain itu, PHP merupakan bahasa *scripting server-side* yang digunakan untuk mengembangkan situs web statis atau situs web dinamis atau aplikasi web. *Script* sendiri merupakan sekumpulan instruksi pemrograman yang ditafsirkan pada saat *runtime*. Bahasa *scripting* adalah bahasa yang menafsirkan skrip saat *runtime* dan biasanya tertanam ke dalam lingkungan perangkat lunak lain [14].

Karena PHP merupakan bahasa pemrograman bertipe *server-side*, PHP akan diproses oleh *server* yang hasil olahannya akan dikirim kembali ke *browser*. Oleh karena itu, salah satu alat yang harus tersedia sebelum menggunakan PHP adalah *server*. Agar dapat menjalankan PHP di komputer sendiri, yang harus dilakukan pertama kali adalah melakukan instalasi *server* [15].

2.2.3 MySQL

My Structured Query Language (MySQL) merupakan salah satu basis data *open-source* yang banyak digunakan pengembang perangkat lunak. MySQL bekerja menggunakan SQL (*Structure Query Language*). MySQL merupakan standar penggunaan database di dunia untuk pengolahan data [16].

MySQL adalah sebuah *Database Management System* (DBMS) menggunakan perintah SQL yang banyak digunakan saat ini dalam pembuatan aplikasi berbasis web. MySQL dibagi menjadi dua lisensi, pertama adalah *free software* dimana perangkat lunak dapat diakses oleh siapa saja. Dan kedua adalah *shareware* di mana perangkat lunak berpemilik memiliki batasan dalam penggunaannya [14].

2.2.4 XAMPP

XAMPP merupakan *server* yang paling banyak digunakan. Fiturnya lengkap namum tetap mudah untuk digunakan oleh pemrogram PHP pemula karena yang perlu dilakukan hanyalah menjalankan salah satu modul bernama Apache yang dapat memproses PHP [15].

2.2.5 Bootstrap

Bootstrap merupaka sebuah *framework* yang dikembangkan oleh Mark Otto dan Jacob Thornton dari Twitter. *Framework* ini diluncurkan sebagai *open source software* pada Agustus tahun 2011 di GitHub. Bootstrap memiliki fitur-fitur komponen antarmuka yang bagus seperti Typography, Forms, Buttons, Tables, Navigations, Dropdowns, Alert, Modals, Tabs, Accordion, Carousel, dan lain sebagainya. Bootstrap memudahkan proses pembuatan tata letak web yang responsif dengan mudah. Salah satu kelebihan yang dimiliki Bootstrap adalah *framework* ini berisi kumpulan *tool* yang gratis untuk membuat tata letak web yang fleksibel dan responsif. *Framework* ini juga memiliki komponen antarmuka bagus [17].

2.2.6 Metode Pengembangan Agile

Metode Pengembangan *Agile* adalah sekelompok metodologi pengembangan perangkat lunak yang didasarkan pada prinsip-prinsip yang sama atau pengembangan sistem jangka pendek yang memerlukan adaptasi cepat dari pengembang terhadap perubahan dalam bentuk apapun.

Pendekatan untuk pengembangan perangkat lunak yang cepat karena persyaratan berubah dalam waktu yang relatif singkat merupakan pengertian *Agile*. Pendekatan ini juga populer saat ini karena memberikan fleksibilitas terhadap pengembang ketika melakukan proses pengembangan. Konsep utama pengembangan *Agile* adalah pengembangan aplikasi, kolaborasi, dan komunikasi antar tim. Tim akan fokus pada pengerjaan aplikasi dengan meminimalkan dokumentasi. Bekerja sebagai tim yang terdiri dari dua atau lebih pemrogram yang mengerjakan fitur dan komunikasi intensif antara pemrogram dan pelanggan.

Ada beberapa macam metode *Agile*, di antaranya adalah *Extreme Programming*, *Adaptive Software Development*, *Dynamic Systems Development Method* (DSDM) dan *Scrum*. Terdapat beberapa prinsip yang berlaku untuk mengimplementasikan pengembangan perangkat lunak, yakni kepuasan pelanggan adalah prioritas utama, menerima perubahan persyaratan walaupun di akhir pengembangan, memberikan hasil/perangkat lunak dalam beberapa minggu hingga bulan, selama proses pengembangan, lingkungan tim yang dapat dipercaya dan memotivasi satu anggota yang lain, mengedepankan komunikasi antar tim dan mengedepankan hasil fungsi dari perangkat lunak tersebut [18].

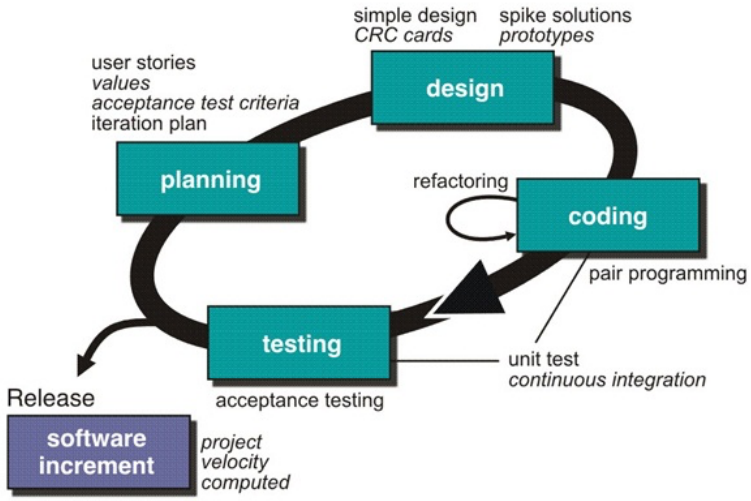
2.2.7 Extreme Programming (XP)

Metode *Extreme Programming* pertama kali digunakan pada 6 Maret 1996. *Extreme Programming* merupakan salah satu metode pengembangan *Agile* yang populer. *Extreme Programming* sudah dibuktikan sukses di banyak perusahaan di berbagai industri [19].

Extreme Programming bisa sukses diterapkan karena mendahulukan kepuasan pelanggan. Alih-alih mengembangkan perangkat lunak dengan banyak kriteria untuk masa yang akan datang. Metode ini mengembangkan perangkat lunak sesuai dengan kebutuhan yang prioritas saja [19].

Extreme Programming meningkatkan proyek perangkat lunak dalam lima cara; *communication*, *simplicity*, *feedback*, *respect*, and *courage*. Pengembang yang menerapkan metode ini melakukan komuikasi dengan pelanggan dan sesama pengembang secara berkesinambungan. Mereka menjaga desain mereka agar tetap sederhana dan bersih. Mereka mendapatkan umpan balik dengan menguji perangkat lunak mereka yang dimulai dari hari pertama. Mereka memberikan sistem ke pengguna sedini mungkin dan menerapkan perubahan seperti yang disarankan oleh pengguna. Setiap kesuksesan kecil memperdalam rasa hormat mereka terhadap kontribusi dari setiap anggota tim. Dengan landasan ini, pengembang dapat dengan berani menanggapi perubahan persyaratan dan teknologi [19].

Aspek yang paling unggul dari *Extreme Programming* adalah alurnya yang sederhana. Gambar 2.1 merupakan alur pada *Extreme Programming*:



Gambar 2.1 Alur *Extreme Programming* (XP) [20]

Tahap ***planning*** dimulai dengan membuat *user stories*, *acceptance test criteria*, dan merencanakan iterasi. Pada tahap ***design***, *Extreme Programming* akan menggunakan *spike solution* di mana pembuatan desain dibuat langsung ke tujuannya. Selanjutnya pada tahap ***coding***, *Extreme Programming* akan memulainya dengan membangun *unit test* yang kemudian dilanjut dengan membuat *production code* yang sesuai dengan *unit test* tersebut. Pada tahap ***testing***, dilakukan pengujian *user acceptance test* (UAT). Pengujian ini dilakukan oleh *customer* yang berfokus pada fitur dan fungsi sistem secara keseluruhan [20].

Extreme Programming sangat mirip dengan permainan *puzzle*. Ada banyak potongan kecil. Secara individual potongan-potongan itu tidak ada artinya, tetapi ketika digabungkan bersama, potongan-potongan itu menjadi gambar yang jelas. Aturan *Extreme Programming* mungkin tampak canggung dan bahkan mungkin naif pada awalnya, tetapi didasarkan pada nilai dan prinsip yang sehat [19].

Aturan *Extreme Programming* menetapkan harapan di antara anggota tim tetapi bukan tujuan akhir itu sendiri. Aturan ini menentukan lingkungan yang

mempromosikan kolaborasi dan pemberdayaan tim. Setelah tercapai, kerja sama tim yang produktif akan berlanjut meskipun aturan diubah agar sesuai dengan kebutuhan khusus perusahaan [19].

2.2.8 GitHub

Github adalah perangkat lunak *hosting* untuk *open source software* yang menggunakan sistem kontrol versi Git. Git adalah *tool* untuk melakukan revisi kode, sedangkan Github adalah web *hosting*-nya. Mudahnya Github adalah web *hosting* untuk proyek-proyek perangkat lunak seperti Linux, Atom, dan lain-lain. Dalam bahasa gaul, Github adalah jejaring sosial untuk software developer [21].

2.2.9 *Unit Test*

Unit test atau pengujian unit merupakan bagian dari *automated test* di mana pengujian yang umpan baliknya cepat, konsisten, dan tidak ambigu. Cepat yang dimaksud adalah sebuah *unit test* harus selesai dijalankan dalam milidetik. Konsisten yang dimaksud adalah *unit test* harus memberikan hasil yang sama pada kode yang sama. Urutan eksekusi *unit test* tidak berpengaruh, sehingga *unit test* dapat dijalankan secara paralel. Tidak ambigu yang dimaksud adalah *unit test* harus secara jelas memberikan informasi tentang *bug* yang terdeteksi [22].

2.2.10 PHPUnit

PHPUnit merupakan kerangka pengujian berorientasi programmer untuk PHP. PHPUnit menggunakan contoh arsitektur xUnit untuk kerangka pengujian unit. Dengan kata lain, kerangka pengujian ini digunakan untuk melakukan pengujian unit pada perangkat lunak yang menggunakan bahasa pemrograman PHP [23].

2.2.11 *Data Flow Diagram* (DFD)

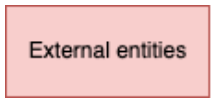
Data Flow Diagram (DFD) merupakan sebuah model data *Requirements Modeling Language* (RML) yang meyediakan representasi grafis dari aliran informasi melalui solusi, berfokus pada bagaimana data ditransformasikan saat dimanipulasi atau digunakan dalam proses. *Data Flow Diagram* menunjukkan

tampilan solusi yang menyatukan banyak alur proses yang berbeda, alur sistem, atau *use case*. *Data Flow Diagram* tidak menunjukkan keputusan (*decision*), dan tidak menunjukkan urutan proses. *Data Flow Diagram* hanya menunjukkan bagaimana data mengalir di antara proses dan diubah oleh proses [24].

Data Flow Diagram berasal dari seperangkat teknik analisis yang disebut analisis terstruktur, yang didefinisikan pada tahun 1970-an dan 1980-an (DeMarco 1979). Dalam analisis terstruktur, pengembang perangkat lunak membuat *Data Flow Diagram* dari diagram konteks dan kemudian menguraikan *Data Flow Diagram* untuk membuat modul fungsional dari solusi (Yourdon 1986). *Data Flow Diagram* digunakan untuk membantu mengumpulkan persyaratan daripada merancang arsitektur teknis. *Data Flow Diagram* adalah tampilan berorientasi data dari solusi yang memungkinkan pengembang untuk mengembangkan gambaran besar tentang bagaimana data bergerak melalui solusi.

Data Flow Diagram merupakan diagram visual yang menggunakan 4 tipe elemen, seperti yang tertera pada Tabel 2.2 berikut:

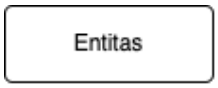
Tabel 2.2 Elemen *Data Flow Diagram*

Elemen	Arti
	*Data stores* merupakan tempat di mana data disimpan sementara atau permanen [24].
	*External entities* dapat berupa orang atau sistem lainnya yang memberi data atau mengambil data dari sistem [24].
	*Process* merupakan tanda bahwa data dimanipulasi [24].
	Merupakan aliran data. Aliran data harus melewati elemen *process*, karena *external entities* dan *data stores* tidak mengirimkan data antara satu sama lain secara langsung [24].

2.2.12 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram merupakan diagram struktural yang digunakan untuk merancang basis data. *Entity Relationship Diagram* akan mendeskripsikan data yang disimpan pada sebuah sistem maupun batasannya[25]. Tabel 2.3 merupakan penjelasan dari elemen-elemen *Entity Relationship Diagram*.

Tabel 2.3 Elemen *Entity Relationship Diagram*

Elemen	Nama Elemen	Keterangan
	Entitas	Entitas dapat berupa orang, tempat, objek, atau kejadian yang dapat dianggap penting bagi sebuah organisasi atau perusahaan. Setiap entitas memiliki beberapa atribut yang mendeskripsikan karakteristik dari objek. Atribut yang ada dalam entitas harus disimpan dan dicatat dalam basis data [25].
	Atribut	Setiap entitas memiliki karakteristik tertentu atau yang disebut dengan atribut. Atribut berfungsi untuk mendeskripsikan karakteristik yang ada pada entitas yang disimpan dalam basis data. Berdasarkan karakteristik sifatnya, atribut dapat dibedakan menjadi beberapa jenis yaitu *simple attribute* dan *composite attribute*, *single valued attribute*, *multi value attribute*, *derived attribute*, dan *key attribute*. *Primary key* adalah nama untuk atribut yang digunakan dalam mengenali suatu entitas. Atribut dalam entitas yang merupakan *primary key* adalah kode identifikasi yang bersifat unik ditunjukkan berdasarkan masing-

Elemen	Nama Elemen	Keterangan
		masing *record* pada sistem. *Primary key* bertujuan untuk memberitahu lokasi untuk tiap catatan pada suatu berkas tentang catatan-catatan yang sama [25].
	Relasi	Relasi adalah sebuah hubungan antara dua atau lebih entitas yang saling berkaitan. Relasi pada ERD dapat digambarkan dengan menggunakan simbol belah ketupat (*diamond*). Relasi memiliki beberapa jenis relasi yaitu *unary*, *binary*, dan *ternary* [25].

2.2.13 *Use Case Modeling*

Ide dasar di balik pemodelan *use case* cukup sederhana yaitu untuk mengetahui inti dari apa yang harus dilakukan sistem. Untuk membuat model *use case*, yang harus dilakukan pertama kali adalah fokus pada siapa (atau apa) yang akan menggunakannya, atau digunakan olehnya. Setelah itu, lihat apa yang harus dilakukan sistem untuk pengguna tersebut agar dapat melakukan sesuatu yang berguna.

Model *use case* mencakup komponen-komponen berikut:

Actors. mewakili orang-orang atau hal-hal yang berinteraksi dalam beberapa cara dengan sistem. *Actors* berada di luar sistem. *Actors* memiliki nama dan deskripsi singkat, dan mereka diasosiasikan dengan *use case* dimana mereka berinteraksi. [26].

Use cases. mewakili hal-hal yang dapat dilakukan sistem untuk para aktornya. *Use cases* bukanlah fungsi atau fitur, dan tidak dapat diuraikan. *Use cases* memiliki nama dan deskripsi singkat. *Use Cases* juga memiliki deskripsi detail yang pada intinya adalah cerita tentang bagaimana *actors* menggunakan sistem untuk melakukan sesuatu yang mereka anggap penting, dan apa yang sistem lakukan untuk memenuhi kebutuhan tersebut.

2.2.14 *Use Case Diagram*

Actors dan *use case* dapat digambarkan pada diagram *use case*. *Actors* diwakili oleh orang tongkat dan kasus penggunaan oleh elips. Panah (mewakili hubungan) menghubungkan aktor dan kasus penggunaan yang berinteraksi. Panah membantu menunjukkan penggagas interaksi. Tujuan dari diagram adalah untuk meringkas apa yang akan dilakukan sistem. Diagram tidak benar-benar mendeskripsikan sistem—mengira diagram *use case* sebagai model *use case* lengkap adalah kesalahan umum yang dilakukan banyak tim. Diagram memberikan ringkasan, tetapi sebagian besar deskripsi diadakan sebagai teks, dalam dokumen yang terkait dengan *use case*. Deskripsi *use case* ini memberikan cerita lengkap tentang apa yang terjadi dalam *use case*. Jadi untuk setiap *use case* dalam model *use case*, akan ada dokumen yang menjelaskan bagaimana *actors* dan sistem berkolaborasi untuk memenuhi tujuan yang diwakili oleh *use case*. Dalam karya tulis ini, saat peneliti mengacu pada *use case*, yang kami maksud adalah totalitas *use case*, termasuk representasi ikoniknya, hubungannya, dan, yang paling penting, deskripsi mendetailnya. Pada Tabel 2.4 dapat dilihat arti dari setiap elemen pada *use case*.

Tabel 2.4 Elemen *Use Case*

Elemen	Arti
	Sebuah *actor* mendefinisikan peran yang dapat dimainkan pengguna saat berinteraksi dengan sistem. Pengguna dapat berupa individu atau sistem lain [26].
	*Use case* menggambarkan bagaimana seorang *actor* menggunakan sistem untuk mencapai tujuan dan apa yang sistem lakukan untuk *actor* untuk mencapai tujuan tersebut. *Use case* bercerita tentang bagaimana sistem dan *actor*-nya berkolaborasi untuk memberikan sesuatu yang bernilai setidaknya untuk salah satu *actor* [26].

Elemen	Arti
	Actor dan *use case* yang berinteraksi satu sama lain dihubungkan oleh asosiasi komunikasi [26].
	Tanda panah bersifat opsional, tetapi jika digunakan, panah menunjukkan bahwa terdapat suatu elemen yang memulai interaksi. Elemen yang memulai interaksi berada di ujung garis yang tumpul [26].
<<include>>	Relasi *include* memberikan kemampuan untuk mengekstrak bagian umum dari dua atau lebih deskripsi *use case* dan menempatkannya dalam *use case* terpisah yang dapat dirujuk [26].

2.2.15 *Activity Diagram*

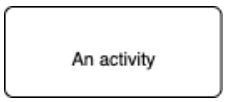
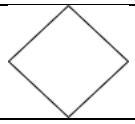
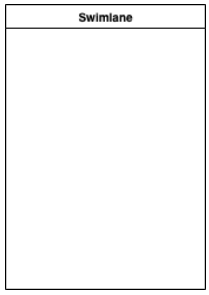
Activity Diagram digunakan untuk menspesifikasikan bagaimana sebuah sistem dapat mencapai tujuan. *Activity diagram* menampilkan proses yang terjadi di sebuah sistem secara umum. Sebagai contoh, *activity diagram* dapat digunakan untuk menggambarkan proses atau tahap-tahap dalam membuat akun blog [27].

Activity diagram pada dasarnya adalah sebuah *flowchart*, yang menampilkan alur kontrol dari suatu aktivitas ke aktivitas lainnya. Tidak seperti *flowchart* tradisional, *activity diagram* menampilkan konkurensi sebagai cabang kendali. Sebagian besar, *activity diagram* melibatkan pemodelan langkah-langkah sekuensial (dan mungkin saja bersamaan) dalam proses komputasi. Diagram ini juga dapat memodelkan aliran nilai di antara langkah-langkah [28].

Notasi-notasi dalam *activity diagram* dapat dilihat pada Tabel 2.5.

Tabel 2.5 Elemen *Activity Diagram*

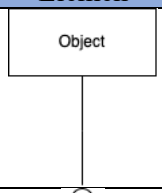
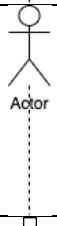
Elemen	Arti
	Merepresentasikan status awal atau titik mulai aktivitas [29].

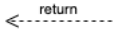
Elemen	Arti
	Merepresentasikan sebuah aktivitas pada sistem. Aktivitas biasanya diawali dengan kata kerja [30].
	Merepresentasikan sebuah pengambilan keputusan (*decision*) dalam alur logika pemrograman [30].
	Digunakan sebagai penghubung antar elemen [30].
	Merepresentasikan status akhir atau titik akhir aktivitas [30]
	Merupakan wadah dari setiap aktivitas atau proses yang terjadi [30].

2.2.16 *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *Use Case Diagram* [27]. Tabel 2.6 merupakan penjelasan dari elemen-elemen *Sequence Diagram*.

Tabel 2.6 Elemen *Sequence Diagram*

Elemen	Nama Elemen	Penjelasan
	*Object*	Sebuah objek yang berasal dari kelas. Atau dapat dinamai dengan kelasnya saja [27].
	*Actor*	*Actor* termasuk objek. Garis putus-putus menunjukkan garis hidup suatu objek [27].
	*Activation*	Menunjukkan masa hidup dari objek [27].

Elemen	Nama Elemen	Penjelasan
	Message	Interaksi antara satu objek dengan objek lainnya. Objek dapat mengirimkan pesan ke objek lain. Interaksi antar objek ditunjukkan pada bagian operasi pada diagram kelas [27].
	*Return*	Pesan kembalian dari komunikasi antar objek [27].

2.2.17 *Black-Box Testing*

Black-Box Testing memperhatikan perilaku eksternal, spesifikasi, dan hasil akhir yang diinginkan yang dihasilkan oleh suatu perangkat lunak dengan memberikan sekumpulan parameter *input*. Tujuan utama *black-box testing* adalah untuk memverifikasi perangkat lunak dengan menggunakannya tanpa memiliki pengetahuan tentang cara kerja internal sistem yang diuji. *Black-box testing* membantu penguji untuk mengidentifikasi apakah perangkat lunak memenuhi semua persyaratan yang dinyatakan dan tidak dinyatakan dan berperilaku sesuai perspektif pengguna akhir (*end-user*) [31]. Dibutuhkan paling sedikit 30 penguji untuk menjamin bahwa data yang dikumpulkan tidak bias [32].

2.2.18 *White-Box Testing*

White-Box Testing dilakukan pada kode aplikasi perangkat lunak. *White-box testing* melibatkan verifikasi fungsi, *loop*, pernyataan, strukturnya, aliran data, hasil keluaran yang diharapkan berdasarkan serangkaian nilai input tertentu, serta desain internalnya. Sebagian darinya dicakup selama proses peninjauan kode dan pengujian unit untuk memastikan cakupan kode sesuai dengan persyaratan yang ditentukan. Cakupan pernyataan, kondisi cakupan jalur, dan cakupan fungsi adalah semua komponen cakupan kode yang membantu peninjau meninjau setiap aspek kode [31]. Dengan bantuan *White-Box Testing*, peneliti dapat mengidentifikasi hal-hal berikut:

a) Bagian kode yang tidak dapat dijangkau, sebagian besar dibuat menggunakan pernyataan GOTO.

b) Variabel (lokal atau global) yang belum pernah digunakan atau menyimpan nilai yang tidak valid.
c) Kebocoran memori atau *memory leak* di mana alokasi memori dan dealokasi untuk variabel atau *pointer* telah diurus.
d) Apakah suatu fungsi mengembalikan nilai dalam jenis yang benar dan format yang diharapkan.
e) Apakah semua variabel, *pointer*, kelas, dan objek yang diperlukan diinisialisasi seperti yang diharapkan.
f) Apakah kode dapat dibaca dan mengikuti konvensi pengkodean organisasi.
g) Apakah kode yang baru ditambahkan berfungsi seperti yang diharapkan dengan bagian kode yang sudah ada.
h) Apakah aliran data berurutan dan akurat.
i) Efisiensi dan kinerjanya untuk mengoptimalkan kode.
j) Pemanfaatan sumber daya.
k) Apakah semua persyaratan konfigurasi telah dipenuhi dan menyertakan semua dependensi untuk menjalankan komponen atau seluruh aplikasi.

2.2.19 *User Acceptance Testing* (UAT)

Mengonfirmasi apakah produk atau layanan perangkat lunak dapat diterima dan berfungsi sesuai harapan pengguna akhir. Sebagian besar organisasi memiliki pengujian penerimaan pengguna atau *User Acceptance Testing* (UAT) sebagai fase pengujian terpisah, yang umumnya dilakukan oleh sekelompok kecil pengguna akhir atau klien. Tujuannya adalah untuk memverifikasi bahwa produk perangkat lunak berfungsi dan memenuhi kebutuhan pelanggan, aman digunakan, dan tidak memiliki efek buruk pada pengguna akhir. Ini memberi tim pengembangan kesempatan untuk memasukkan fitur yang hilang atau permintaan peningkatan sebelum merilis produk ke khalayak yang lebih luas. Pada tahap ini, klien masih bisa menolak produk atau fiturnya. Saat pengujian dilakukan di dalam organisasi, meniru penyiapan lingkungan dunia nyata, ini disebut sebagai pengujian alfa. Saat uji penerimaan dilakukan oleh pengguna akhir di lingkungannya sendiri, ini disebut sebagai pengujian beta. Dalam jenis pengujian ini, tim pengembangan tidak terlibat dengan pengguna akhir yang sebenarnya. Ini adalah tes yang bagus untuk berbagi

versi beta produk dengan kelompok pengguna akhir aktual yang relatif kecil sehingga mereka dapat memverifikasi produk, fungsinya, dan fiturnya [31].

Namun, saat merilis versi beta, penting untuk mencantumkan persyaratan perangkat keras atau perangkat lunak. Bersamaan dengan itu, tim eksekutif pendukung yang berdedikasi harus tersedia untuk menjawab pertanyaan pelanggan. Selain itu, versi perangkat lunak ini dapat tersedia secara gratis untuk waktu yang terbatas (umumnya, dua minggu hingga satu bulan) untuk mendorong lebih banyak orang berpartisipasi dalam pengujian yang sebenarnya [31]. Dibutuhkan paling sedikit 30 penguji untuk menjamin bahwa data yang dikumpulkan tidak bias [32].

Teknik UAT merupakan pengujian terakhir sebelum sistem dipakai oleh pengguna yang melibatkan pengujian dengan data pengguna sistem. Perhitungan dilakukan menggunakan skala likert. Skala likert merupakan suatu skala penilaian yang menyajikan pilihan skala dengan nilai pada setiap skala untuk mengukur tingkat persetujuan terhadap sesuatu. Skala Likert digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok tentang kejadian atau gejala sosial dimana setiap pertanyaan memiliki bobot nilai. Adapun bobot jawaban yang diberikan adalah sangat baik (5), baik (4), cukup baik (3), kurang baik (2), tidak baik (1). Perhitungan dilakukan dengan rumus (2.1) [33]:

$$L = \frac{\big((SB \times 5) + (B \times 4) + (CB \times 3) + (KB \times 2) + (TB \times 1)\big)}{NB} \times 100$$

(2.1) [33]

Keterangan:

L: Nilai akhir.

SB: Jumlah nilai yang diperoleh pada kategori Sangat Baik (skala lima tingkatan).

B: Jumlah nilai yang diperoleh pada kategori Baik (skala lima tingkatan).

CB: Jumlah nilai yang diperoleh pada kategori Cukup Baik (skala lima tingkatan).

KB: Jumlah nilai yang diperoleh pada kategori Kurang Baik (skala lima tingkatan).

TB: Jumlah nilai yang diperoleh pada kategori Tidak Baik (skala lima tingkatan).

NB: Total jumlah responden atau elemen dalam kategori nilai (jumlah nilai yang diberikan oleh semua responden) [33].

2.2.20 Teknik Analisis Deskriptif

Analisis deskriptif berfungsi untuk menghasilkan nilai presentase dari responden pengujian [34]. Salah satunya digunakan dalam pengujian fungsionalitas sebuah sistem dengan formalitas perhitungannya sebagai berikut yang sesuai dengan rumus 2.2 beserta penentuan interpretasi dari tabel presentasi kelayakan yang tertera pada Tabel 2.7 [35]:

$$Presentase\ kelayakan\ = \frac{Skor\ yang\ diobservasi}{Skor\ yang\ diharapkan} \times 100\% \quad \textbf{(2.2)}$$

Tabel 2.7 Tabel Presentase Kelayakan [36]

Presentase	Kelayakan
81% - 100%	Sangat Layak
61% - 80%	Layak
41% - 60%	Cukup
21% - 40%	Tidak Layak
≤20%	Sangat Tidak Layak