

II. TINJAUAN PUSTAKA

2.1 Aplikasi Monitoring

Prasetyo dkk. (2020) mengatakan bahwa monitoring merupakan kegiatan mengikuti suatu program yang pelaksanaannya dilakukan secara teratur dengan cara mendengar, mengamati, serta mencatat perkembangan dari program tersebut. Pernyataan ini selaras dalam seri monograf 3, *UNESCO Regional Office for Education in Asia and the Pacific*, dijelaskan bahwa monitoring adalah upaya yang dilakukan secara rutin untuk mengidentifikasi pelaksanaan dari berbagai komponen program yang telah direncanakan, dijadwalkan, dan kemajuan dalam mencapai tujuan program.

Pengertian lain juga dikemukakan oleh Munazilin dkk. (2017), yang mengatakan bahwa monitoring terfokus pada kegiatan yang sedang berjalan dengan tujuan untuk mengetahui apakah kegiatan yang sedang berlangsung sesuai dengan perencanaan dan prosedur yang telah disepakati. Berdasarkan uraian pendapat tentang monitoring di atas, dapat disimpulkan bahwa aplikasi monitoring berarti suatu *software* yang digunakan untuk menganalisis dan mengumpulkan sebuah informasi yang telah ditentukan.

2.2 Skripsi

Skripsi atau karya tulis ilmiah seorang mahasiswa yang disusun untuk memenuhi persyaratan memperoleh gelar sesuai dengan keilmuannya (Zulkarnaini, 2020). Menurut Kamus Besar Bahasa Indonesia (KBBI), skripsi merupakan karangan ilmiah yang wajib ditulis oleh mahasiswa sebagai bagian dari persyaratan akhir pendidikan akademisnya (Departemen Pendidikan Nasional, 2021).

Pemikiran mahasiswa yang dituangkan ke dalam bentuk karya ilmiah merupakan suatu proses pembelajaran untuk melatih mahasiswa. Mengingat tujuan penulisan skripsi dilakukan agar mahasiswa mampu menyusun dan menulis suatu karya ilmiah sesuai dengan bidang ilmu yang ditempuh, mampu melakukan penelitian mulai dari merumuskan masalah, mengumpulkan data, mengolah data, menarik suatu kesimpulan, mengaplikasikan ilmu yang diperoleh ke dalam kehidupan dan yang tidak kalah penting, skripsi menjadi tolak ukur mahasiswa sejauh mana tingkat pemahaman terhadap ilmu yang dimilikinya. Standar yang dapat digunakan untuk mengukur keilmiahan suatu karya tulis menurut Mansyur (2018) antara lain:

1. *Empiris*, adanya bukti-bukti yang dapat dihadirkan melalui kegiatan observasi, eksperimen, dan sebagainya.
2. Objektif, fakta atau data yang ditemukan sesuatu hal yang sebenarnya.
3. Sistematis, dalam menyusun dan menganalisis data menggunakan sistem yang baku. Demikian pula dengan hasil deskripsinya disajikan secara sistematis.

Proses penyusunan skripsi memerlukan waktu tidaklah sebentar. Namun, sesuai dari ketekunan mahasiswa. Mahasiswa yang tekun dan bersungguh sungguh dapat menyelesaikan skripsinya dalam satu semester begitu pun sebaliknya. Tahapan skripsi yang harus ditempuh pada Pendidikan Teknologi Informasi, Universitas Lampung antara lain:

1. Pengajuan judul skripsi beserta dosen pembimbing dan pembahas merupakan tahap penting, tahap tersebut diberikan keputusan diterima, revisi, atau ditolak judul skripsi yang diajukan mahasiswa. Setelah ditentukan oleh ketua program studi, maka mahasiswa dapat melanjutkan skripsinya.
2. Seminar usul penelitian atau seminar proposal merupakan tahap setelah mahasiswa menyelesaikan proposal penelitian dan mendapatkan persetujuan dari dosen pembimbing untuk mempresentasikan rencana penelitian yang akan dilakukan. Tujuan seminar proposal adalah untuk memperoleh masukan dari mahasiswa, dosen pembimbing, dan dosen pembahas untuk menyempurnakan rencana penelitian sebelum melaksanakan penelitian.

3. Seminar hasil merupakan tahap memaparkan hasil penelitian setelah menyelesaikan kegiatan penelitian sesuai dengan perencanaan. Tujuan seminar hasil adalah untuk memperoleh masukan dari mahasiswa, dosen pembimbing, dan dosen pembahas untuk menyempurnakan pembahasan hasil penelitian.
4. Sidang komprehensif merupakan sidang akhir yang dilakukan mahasiswa setelah melakukan tahapan seminar proposal dan seminar hasil. Sidang komprehensif merupakan kegiatan ujian yang bersifat menyeluruh, untuk mengukur kemampuan dan penguasaan mahasiswa berdasarkan masing-masing bidang keilmuan yang sudah ditempuh.

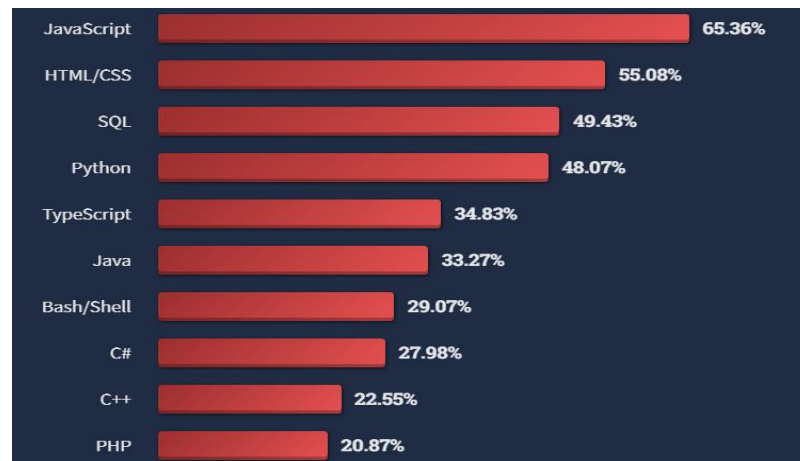
Berdasarkan pengertian dan uraian di atas, dapat disimpulkan bahwa skripsi merupakan suatu karya ilmiah yang wajib dibuat oleh seorang mahasiswa yang menjadi syarat untuk menuntaskan studi S-1 untuk menerima gelar sarjana. Penerapan di Universitas Lampung, terdapat beberapa tahapan untuk menyelesaikan skripsinya, dimulai dari pengajuan judul skripsi, seminar proposal, seminar hasil, hingga sidang komprehensif.

2.3 Pemrograman *Frontend*

Frontend merupakan antarmuka *interface* untuk memperbaiki tampilan *website* menjadi semakin menarik dan memudahkan pemahaman *user* dalam mengoperasikan *website* (Prawastiyo & Hermawan, 2020). Pernyataan tersebut selaras dengan pendapat sari *et al.* (2022) yang mengatakan bahwa *frontend* mengacu pada desain *layout* suatu aplikasi atau *website*, *frontend* bertujuan agar desain terlihat menarik dan tidak membosankan. Pengertian lain juga diungkapkan oleh Widhyaestoeti dkk. (2021) yang mengatakan bahwa antarmuka (*user interface*) yang disebut *frontend*, merupakan bagian sistem yang menyediakan suatu tampilan kepada pengguna untuk mengembangkan komponen visual pada sistem yang bertanggung jawab terhadap tampilan antarmuka.

Javascript adalah bagian dari tiga teknologi penting yang harus dikuasai oleh *programmer web*, yaitu HTML untuk isi konten, CSS untuk tampilan, dan *Javascript* untuk interaksi (Kurniawan & Rosa, 2020). Pernyataan lain juga dikemukakan oleh

Sari & Hidayat (2022) yang mengungkapkan bahwa dengan menggunakan *javascript* untuk mengakses dan berinteraksi dengan API menjadi semakin mudah. *Javascript* adalah bahasa yang berkembang, yang didefinisikan oleh serangkaian standar yang dikenal sebagai *script* ECMA oleh Ecma Internasional. *Javascript* menjadi satu-satunya bahasa yang dipahami oleh *browser* (Matuszek, 2023).



Gambar 1. Hasil Survei Stack Overflow Javascript 2022

Kelebihan javascript menurut Rohman (2020) adalah:

1. Tidak membutuhkan kompiler karena *web browser* dapat menginterpretasikannya dengan HTML.
2. Lebih mudah dipelajari dibandingkan dengan bahasa pemrograman lainnya.
3. *Error* semakin mudah ditemukan dan ditangani.
4. Dengan *javascript*, dapat dialih tugaskan ke elemen halaman *web* atau *event* tertentu.
5. Digunakan pada berbagai *browser*, *platform*, dan lainnya.
6. Dengan *javascript* dapat memvalidasi input dan mengurangi pengecekan data secara manual.
7. *Website* menjadi semakin interaktif.
8. Ringan dan cepat apabila dibandingkan dengan bahasa lainnya.

Kekurangan *javascript* menurut Rohman (2020) adalah:

1. Memiliki resiko tinggi dalam eksploitasi.
2. Dimanfaatkan dalam mengaktifkan sebuah kode yang memiliki kategori berbahaya pada komputer *user*.

3. Tidak selalu didukung oleh berbagai *browser*.
4. *Code snippet* lumayan banyak.
5. Dapat di-*render* pada perangkat yang berbeda.

Berdasarkan uraian di atas, disimpulkan bahwa *javascript* merupakan bahasa pemrograman yang tidak asing bagi *frontend developer* untuk menghasilkan *website* yang lebih *powerfull*. *Javascript* menjadi salah satu pilihan tepat dalam membangun *web server*, terlebih untuk seorang *frontend web developer* tidak perlu menggunakan bahasa yang berbeda dalam membangun *backend*.

2.4 Pemrograman *Backend*

Pemrograman *backend* menurut Nababan dkk. (2022) mengatakan bahwa *backend* itu bekerja di balik layar dengan mengolah *database* dan server. Pengertian lain juga diungkapkan oleh Mubariz dkk. (2020) bahwa *backend* berjalan di sisi server (*server-side*) pada sebuah aplikasi berbasis *web* dan *mobile* yang memiliki tugas untuk berinteraksi secara langsung dengan basis data dalam melakukan manipulasi data, sehingga *backend* tidak melakukan interaksi secara langsung kepada pengguna. Berdasarkan penelitian Syaftiaa dkk. (2021), diketahui bahwa *backend* sangat dibutuhkan dalam sebuah pengembangan suatu sistem dan manajemen sebuah data yang terdapat pada sistem.

Berdasarkan beberapa pendapat tentang pengertian *backend*, dapat disimpulkan bahwa *backend* bertanggung jawab untuk menyediakan kebutuhan di balik layar (tidak berinteraksi langsung dengan pengguna), seperti bagaimana data disimpan, diolah, serta ditransaksikan secara aman untuk mendukung *frontend* dapat bekerja sesuai dengan fungsinya.

2.4.1 *Representational State Transfer (REST)*

REST adalah gaya arsitektur untuk melakukan pengiriman informasi melalui HTTP. Umumnya cara kerjanya seperti *website*. *Client* mengirim *request* ke server kemudian server memberikan *response* kepada *client*. REST dikembangkan oleh *co-founder* yaitu Roy Fielding dari *Apache HTTP Server Project*. Pada arsitektur REST itu sendiri, server menyediakan sebuah *resources*

(sumber data) untuk *client* mengakses dan menampilkan *resource* (sumber data) tersebut. *Resource* dapat direpresentasikan dalam format JSON (Kurniawan dkk., 2020). Tujuan REST yaitu untuk menjadikan sistem memiliki performa yang baik, cepat dan mudah dalam hal pertukaran dan komunikasi data (Iswardhana & Widiono, 2021).

REST memanfaatkan empat metode pada protokol HTTP, yaitu:

1. *GET*, yang memiliki fungsi untuk membaca sebuah data/*resource* dari REST server.
2. *POST*, yang memiliki fungsi untuk membuat data/*resource* baru pada REST.
3. *PUT*, yang memiliki fungsi untuk memperbaharui data/*resource* di REST server.
4. *DELETE*, yang memiliki fungsi untuk menghapus data/*resource* dari REST server.

Pada REST API, berikut nilai-nilai status *code* yang sering digunakan:

1. 200 (*OK*) - Permintaan *client* berhasil dijalankan oleh server.
2. 201 (*Created*) - Server berhasil membuat/menambahkan *resource* yang diminta *client*.
3. 400 (*Bad Request*) - Permintaan *client* gagal dijalankan karena proses validasi input dari *client* gagal.
4. 401 (*Unauthorized*) - Permintaan *client* gagal dijalankan. Biasanya ini disebabkan karena pengguna belum melakukan proses autentikasi.
5. 403 (*Forbidden*) - Permintaan *client* gagal dijalankan karena tidak memiliki hak akses ke *resource* yang diminta.
6. 404 (*Not Found*) - Permintaan *client* gagal dijalankan karena *resource* yang diminta tidak ditemukan.
7. 500 (*Internal Server Error*) - Permintaan *client* gagal dijalankan karena server mengalami *error* (membangkitkan *Exception*).

Ketika permintaan *client* gagal dijalankan, kita harus mengembalikan status *code* yang sesuai dengan kesalahan yang terjadi. Penggunaan *response code*

yang tepat dapat meminimalisasi kebingungan *client/user* dalam memanfaatkan API.

2.4.2 *Application Programming Interface (API)*

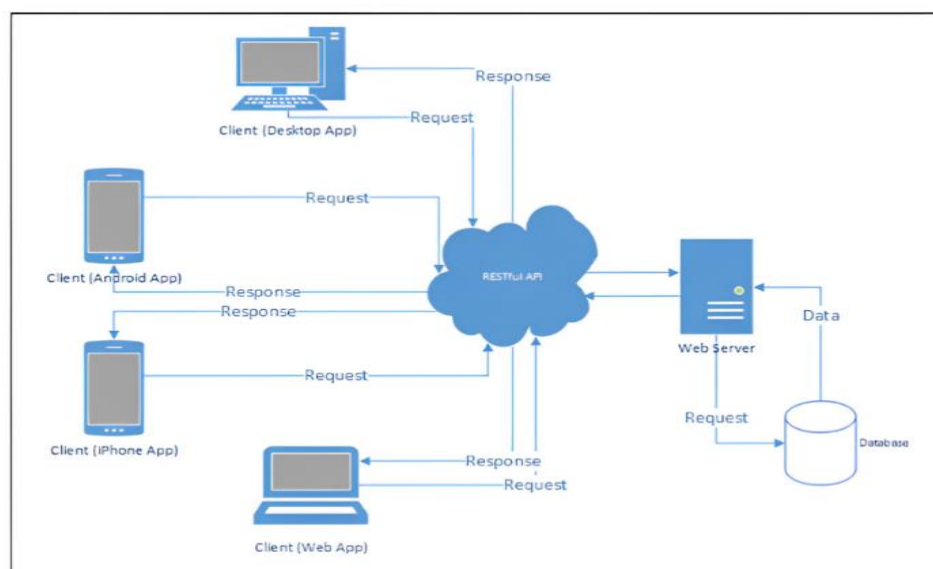
Muri dkk. (2019) mengungkapkan bahwa *Application Programming Interface* (API) adalah konsep fungsi antarmuka pemrograman aplikasi, yang menjadi salah satu cara agar suatu aplikasi dapat diakses dan dimanfaatkan oleh pihak lain tanpa mengubah struktur kode utama maupun *database* sistem, serta dapat memudahkan antar sistem untuk melakukan komunikasi meskipun pada *platform* yang berbeda. Sementara menurut Zein (2018), API dapat mengintegrasikan dari berbagai aplikasi secara bersamaan untuk bertukar data. Misalnya integrasi *login* menggunakan *google*, mengirim pesan verifikasi dari aplikasi melalui *gmail*, dan lain-lain.

Pernyataan tersebut sesuai dengan penelitian yang dilakukan oleh Hasanuddin dkk. (2022) yang mengatakan bahwa dilihat fungsi API, yaitu dapat mengakses aplikasi atau layanan dari sebuah program. API memungkinkan *developer* untuk dapat menggunakan fungsi dari aplikasi lain yang sudah ada sehingga tidak perlu *developer* membuat ulang dari awal. API memiliki tujuan untuk saling berbagi data dengan aplikasi yang berbeda, API juga memiliki tujuan yang dapat mempercepat proses pengembangan sebuah aplikasi dengan menyediakan *function* terpisah sehingga *developer* tidak perlu merancang fitur yang serupa.

Berdasarkan uraian di atas, dapat disimpulkan bahwa API adalah mekanisme untuk dua komponen perangkat lunak untuk saling berkomunikasi menggunakan HTTP protokol, yang menjelaskan cara kedua aplikasi saling berkomunikasi dengan menggunakan *request* dan *response*. Arsitektur API dikenal dengan sebutan klien dan server, yang mengirimkan *request* disebut sebagai *client* dan yang mengirimkan *response* disebut sebagai server.

2.4.3 Implementasi REST API

Terdapat beberapa *platform* aplikasi di sisi *client*, baik *desktop*, *mobile* maupun *web browser*. Komunikasi *client* ke server dilayani dengan menggunakan protokol HTTP dan format data JSON, baik *client* maupun server (Iswardhana & Widiono, 2021).



Gambar 2. Implementasi REST API

2.4.4 Visual Studio Code (VSC)

Visual Studio Code yang merupakan *text editor* yang kompatibel dengan bahasa pemrograman seperti *Javascript*, *Node Js*, *Typescript* dan bahasa pemrograman lainnya. Fitur yang disediakan text editor ini seperti *Git Integration*, *Intellisense*, *Debugging*, dan terdapat fitur ekstensi lainnya yang dapat menambah kemampuan *text editor*. Pembaruan *Visual Studio Code* dilakukan secara berkala, perbedaan ini yang membedakan *Visual Studio Code* dengan *text editor* lainnya (Tunggal Carlie Sucipta, 2020).

Pernyataan ini selaras dengan ungkapan Joni (2019) yang mengatakan bahwa *Visual Studio Code* termasuk dukungan untuk melakukan *debugging*, *git*, penyorotan sintaks, *snippet*, penyelesaian kode, dan *refactoring code* yang membuat *user* untuk mengganti sebuah tema, preferensi, pintasan keyboard dan menginstal ekstensi yang dibutuhkan. Pengertian lain juga dikemukakan

oleh Indriansy dkk. (2021) yang mengatakan *Visual Studio Code* adalah *text editor* yang ringan dan juga handal yang dibuat oleh *Microsoft* dan memiliki versi di berbagai sistem operasi seperti *Linux*, *Mac*, dan *Windows*.

Berdasarkan informasi diatas, *Visual Studio Code* merupakan salah satu *text editor* yang dapat dikatakan ringan dan kuat, yang mampu berjalan pada berbagai sistem operasi. *Text editor* ini dibuat oleh *microsoft* dan memiliki banyak fitur yang sangat cocok untuk membuat program, dikarenakan mendukung bahasa pemrograman *javascript*.

2.4.5 Node JS

Node Js adalah *platform* pengembangan *source* terbuka untuk mengeksekusi sisi server kode pada *javascript*. Pada tahun 2009, Ryan Dahl berhasil menciptakan *node js*, teknologi yang diharapkan oleh banyak *web developer*. Tak disangka saat ini teknologi yang diciptakannya menuai popularitas tinggi, berguna untuk mengembangkan aplikasi yang memerlukan koneksi secara terus menerus dari *browser* ke server. *Node Js* dijalankan menggunakan *single thread* dengan satu proses pada satu waktu.

Node Js dijalankan secara tidak sinkron. *Node Js* memproses *request* dengan cara *asynchronous* yaitu setiap *request* yang masuk akan berada di dalam tumpukan *event* yang konstan dan akan mengirimkan *request* satu demi satu tanpa menunggu *response*. Cara *asynchronous* adalah pergeseran dari model utama atau biasanya disebut *synchronous* yang menjalankan proses yang lebih besar, lebih kompleks, dan menjalankan beberapa perintah secara bersamaan, dengan masing-masing perintah akan menunggu *response* yang sesuai sebelum melanjutkan *request* lainnya (Shah & Soomro, 2017). Penelitian yang dilakukan oleh Saundariya et al. (2021) *Node Js* memiliki aliran asinkron, sehingga menghasilkan kinerja tinggi dengan skalabilitas tinggi.

Bahasa pemrograman *javascript* dikenal berjalan pada sisi *client/browser*, berbeda dengan *node js* yang memiliki peran dalam melengkapi *javascript*, sehingga mampu berjalan pada sisi server seperti halnya *PHP*, *Ruby*, *Perl*, dan sebagainya (Kurniawan dkk., 2020).

Perilisan *Node Js* yaitu dibedakan ke dalam tiga fase, yaitu:

1. *Current*
2. *Long Term Support / LTS*
3. *Maintenance*.



Gambar 3. Hasil Survei *Stack Overflow Node Js* 2022

Gambar 3 menyajikan hasil survei yang dilakukan oleh *stack overflow* pada tahun 2022, bahwa popularitas *node js* meroket menjadi *framework* nomor satu yang banyak digunakan oleh *developer*.

Berdasarkan uraian di atas, dapat disimpulkan bahwa *node js* berhasil menjadi *javascript runtime* yang dapat mengeksekusi kode *javascript* di luar *browser* yang menjadikan *node js* menjadi gerbang bagi para *javascript developer* untuk mengembangkan sistem di luar dari *browser*.

2.4.6 *Express Js*

Express Js adalah sebuah *framework* di mana *framework* ini mudah dikembangkan dalam sebuah pengembangan aplikasi *web*, *service API*, *routing*, maupun *security*. Dikembangkannya *express js* berguna pada penggunaan *design pattern* yang dapat disesuaikan dengan arsitektur apapun, sehingga sangat *powerfull* dan fleksibel. Pembuatan arsitektur dengan *express js*, API

yang digunakan sangat ringan dan tidak memakan *resource* banyak, sehingga sangat memangkas *cost* yang digunakan untuk pengembangan *website* selanjutnya (Fajrin, 2017). Mengatur *middleware* pada permintaan HTTP sebuah *routing*, akan memiliki fungsi yang berbeda sesuai HTTP atau *URL* yang dipakai. *Express Js* bersifat dinamis dan *powerfull* karena terdapat modul yang tersedia di dalam NPM (*Node Package Manager*) yang dapat diintegrasikan (Munawar, 2018).

Berikut mekanisme pada *express js*:

1. Penulisan penanganan (*handler*) untuk melakukan permintaan dengan HTTP di jalur URL atau rute (*routes*) yang berbeda.
2. Terintegrasi dengan mesin *rendering* ‘*view*’ atau terintegrasi dengan bagian *frontend*, sehingga dapat menghasilkan *response* dengan memasukkan data ke dalam tampilan atau *user interfaces template*.
3. Mengatur *web* secara umum seperti port yang digunakan untuk menghubungkan server dengan lokasi dalam tampilan atau *user interfaces template* yang digunakan untuk merender *response*.
4. Menambahkan pemrosesan permintaan tambahan “*middleware*” pada setiap titik dalam rute (*route*) permintaan (*request*).

2.4.7 PostgreSQL

PostgreSQL menurut Obe dan Hsu (2016) dalam buku *PostgreSQL: Up and Running* adalah sistem basis data yang disebarluaskan secara bebas dalam perjanjian lisensi BSD. Piranti ini menjadi salah satu basis data selain *MySQL* dan *Oracle* yang banyak digunakan pada saat ini. Pernyataan ini selaras dengan ungkapan Rahardja (2022) yang mengatakan bahwa *software* ini menjadi *database* yang paling banyak digunakan pada saat ini, selain *MySQL* dan *Oracle*. Di dalam *postgresql* terdapat fitur yang dapat berguna untuk replikasi *database*. Berdasarkan penelitian oleh Praba & Safitri (2020) untuk pembuatan *database* skala besar, *postgresql* menjadi solusi yang lebih baik karena beberapa *query* lebih unggul dibandingkan *MySQL*.

Berdasarkan pengertian di atas, dapat disimpulkan bahwa *postgreSQL* adalah sistem manajemen *database* relasional (RDBMS) yang bersifat *open source*, yang memiliki banyak fitur canggih untuk membuat pengelolaan data lebih mudah. *postgreSQL* banyak digunakan pada *web app* dan *mobile*. Itulah alasan *postgreSQL* lebih cocok untuk aplikasi yang membutuhkan pengolahan data yang lebih kompleks.

Fungsi utama *postgreSQL* adalah tempat untuk menyimpan dan mengelola data melalui perintah atau *query SQL*. Dengan *query* tersebut, maka *postgreSQL* bisa digunakan untuk:

1. Membuat atau memanipulasi tabel dengan *Data Definition Language* (DDL) menggunakan *query* seperti *CREATE*, *DROP*, *ALTER*.
2. Memanipulasi isian data atau value dari tabel dengan *Data Manipulation Language* (DML) menggunakan *query* *INSERT*, *UPDATE*, *DELETE*.
3. Mengelola transaksi di database dengan *Data Control Language* (DCL) menggunakan *query* seperti *GRANT*, *REVOKE*, *COMMIT*.

2.4.8 Sequelize

Sequelize yaitu sebuah *ORM* yang dapat mempermudah pemetaan *database* menjadi *object*, sehingga mempercepat dan mempermudah proses interaksi dengan *database* (Suyasa, 2018). Fitur ini mendukung transaksi yang solid, *lazy loading* dan *eager* (Hasanuddin dkk., 2022). *Sequelize* ORM bekerja dengan *database* dan relasi-relasi di dalamnya, sehingga saat *deploy* tidak melakukan perubahan konteks saat menuliskan kode, karena melalui API sudah membuat interaksi dengan menggunakan bahasa *javascript* yang telah disediakan oleh *Sequelize*. Fitur terbaik *Sequelize* yaitu dapat menggunakan API dan *database* yang berbeda (Fahmi, 2021).

2.4.9 JavaScript Object Notation (JSON)

JSON adalah format yang ringan untuk memasukan sebuah data ke dalam variabel. JSON sangat mudah untuk komputer dalam melakukan parsingnya,

selain itu juga mudah untuk dipahami dan diimplementasikan oleh manusia (Brata, 2015). Pengertian lain oleh Dewi dkk. (2019) yang mengungkapkan bahwa JSON merupakan salah satu jenis format *text* yang tidak bergantung pada bahasa pemrograman lainnya, dikarenakan format JSON sendiri menggunakan sebuah gaya bahasa pemrograman yang umum digunakan oleh *programmer* termasuk *C*, *C++*, *Java*, *Javascript*, *Perl*, *Python* dll. Oleh sebab itu, sifat tersebut menjadikan format teks JSON menjadi bahasa pertukaran data yang ideal.

Pendapat tersebut selaras dengan Hasanuddin dkk. (2022) yang mengungkapkan bahwa JSON termasuk turunan dari *Javascript*, namun dapat digunakan pada bahasa pemrograman lainnya termasuk *Python*, *Ruby*, *PHP*, dan *Java*. Saat *standalone* ekstensi yang digunakan ke dalam JSON yaitu *.json*. Sementara saat didefinisikan ke dalam format lain (seperti dalam *.html*), dapat tampil dalam tanda petik yaitu *JSON string* atau dapat dimasukkan kedalam variabel. Format *text* ini mudah untuk dapat ditransfer antar server *web* dengan *client* atau *browser*.

Berdasarkan uraian di atas, dapat disimpulkan bahwa JSON merupakan format pertukaran data yang mengirim data dari server ke *client*, sehingga dapat ditampilkan pada halaman *web*, atau sebaliknya. Dalam penyimpanan data, JSON menggunakan metode *object* dan *array*.

2.4.10 Group Inclusive Tour (GIT)

GIT merupakan *version control system* yang digunakan dalam perancangan REST API. Fungsi pada GIT yaitu dapat digunakan untuk mengatur versi dari *source code program* dengan memberi tanda atau catatan pada baris kode program yang diganti. GIT memudahkan *programmer* ketika bekerja secara tim, untuk dapat mengetahui perubahan yang terjadi pada *source code* yang dilakukan. GIT mempermudah tim untuk kolaborasi hasil kerja serta sebagai perantara layanan untuk proses *push* ke *cloud server* agar REST API dapat dikonsumsi oleh *Frontend*.

Berikut ini fitur-fitur pada GIT:

1. GIT *Init*

Fitur GIT init dapat membuat *repository* pada *file* yang berada di lokal, nantinya ada folder.git.

2. GIT Status

Fitur GIT status dapat mengetahui seluruh status yang berada pada *repository* lokal.

3. GIT Add

Fitur GIT add untuk menambahkan *file* baru pada *repository* yang dipilih.

4. GIT Commit

Fitur GIT commit digunakan menyimpan setiap terjadinya perubahan yang dilakukan, akan tetapi tidak ada perubahan pada *remote repository*.

5. GIT Push

Fitur GIT *push* dapat mengirimkan perubahan yang terjadi setelah melakukan *commit* pada *file* ke *remote repository*.

6. GIT Branch

Fitur GIT *branch* dapat melihat seluruh *branch* yang ada pada *repository*.

7. GIT Merge

Fitur GIT *merge* berfungsi untuk menggabungkan *branch* aktif dan *branch* yang dipilih.

8. GIT Clone

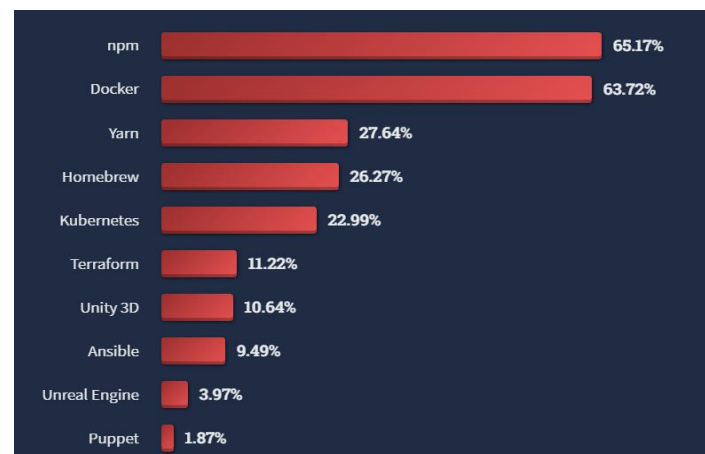
Fitur GIT *clone* untuk dapat membuat salinan *repository* pada lokal.

2.4.11 Node Package Manager (NPM)

Nasution (2021) mengungkapkan bahwa NPM atau *node package manager* digunakan *node js* untuk membantu *javascript developer* dengan mudah dalam berbagi modul kode. Adanya NPM proses dalam pengembangan perangkat lunak dapat menjadi lebih mudah dan juga cepat, dikarenakan *developer* tidak perlu membuat kode mulai dari nol, melainkan *developer* dapat menginstal *package* yang dibutuhkan dan dapat memodifikasi kode pada *package* tersebut

jika diperlukan. Cara mengimplementasikan NPM sendiri cukup mudah yaitu ketika ingin menginstal *package* cukup dengan melakukan *npm install <package-name>* misal ketika ingin menginstall *framework express js* dapat dilakukan dengan *npm install express js* atau *npm i express js*, maka akan secara otomatis menginstall *express js*.

Penelitian yang dilakukan oleh Alcantara *et al.* (2021), yang mengungkapkan bahwa NPM merender klien antarmuka pengguna grafis (GUI) setelah diluncurkan oleh pengguna dari dalam konfigurasi terintegrasi untuk menerbitkan NPM yang dibuat dalam IDE pada lingkungan pengembangan (IDE). Berdasarkan uraian di atas, dapat disimpulkan bahwa dalam pengembangan aplikasi, kita tidak lepas dari *package*. *Package manager* dapat membantu kita dalam mengembangkan aplikasi *web*, dapat dengan mudah memasang dan menghapus *package* yang dibuat oleh orang lain pada proyek.



Gambar 4. Hasil Survei *Stack Overflow* NPM 2022

Gambar 4 menyajikan hasil survei yang dilakukan oleh *stack overflow* pada tahun 2022, bahwa *npm* menjadi nomor satu yang banyak digunakan oleh *developer*.

2.4.12 *JSON Web Token (JWT)*

JWT yaitu *access token* dengan format *JSON* yang digunakan sebagai autentikasi token pada *client-server* (Satria dkk., 2018). Penggunaan JWT

memungkinkan *request* yang aman dengan API, yang dapat mengurangi risiko akan terjadinya hal yang tidak diinginkan ketika API aktif (Chandra & Tan, 2020). Berdasarkan penelitian Kurniawan dkk. (2020), JWT itu berbentuk string panjang yang sangat random, berfungsi untuk melakukan proses autentikasi dan pertukaran informasi. Umumnya, untuk melakukan *login* tidak seperti pada aplikasi *website* biasanya, di mana menggunakan *session* untuk dapat mengingat siapa yang sedang melakukan *login*, tetapi ketika di dalam API kita menggunakan konsep JWT atau dibacanya "jot", untuk *website* resminya yaitu dapat mengunjungi jwt.io.

JWT mempunyai tiga struktur yaitu:

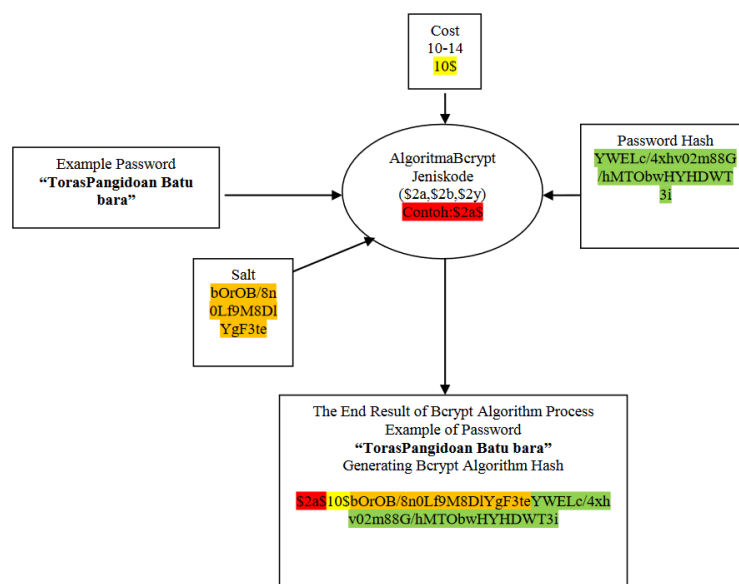
1. *Header* yang berisi informasi tentang *signature* JWT untuk diproses validasi JWT.
2. *Payload* untuk memuat data aplikasi yang dimasukkan ke dalam JWT.
3. *Signature* untuk memuat nilai *hash* untuk memverifikasi *payload*.

2.4.13 Bcrypt

Bcrypt menerapkan *salting* untuk dapat melindungi dari *rainbow table attacks*, yaitu suatu tabel yang telah dikomputasi untuk membalikkan fungsi *hash* kriptografi, biasanya digunakan untuk memecahkan *hash* kata sandi (Chandra & Tan, 2020). Pendapat tersebut selaras dengan Rompis & Aji (2018) yang mengatakan bahwa proses *login* melibatkan akses basis data, sekaligus eksekusi algoritma *bcrypt js* untuk membandingkan *hash password*. *Bcrypt js* adalah sebuah algoritma kriptografi yang relatif lebih rumit dibandingkan dengan verifikasi JWT.

Penelitian oleh Giffary & Ramadhani (2022) *Password* yang disarankan menggunakan campuran berbagai jenis karakter, seperti huruf kecil, huruf kapital, angka, dan simbol. Penggunaan *password* yang tidak bermacam-macam karakternya, akan dengan mudah dipecahkan oleh *hacker* oleh karena itu diperlukannya ilmu kriptografi yang ada pada *bcrypt* untuk menjaga keamanan pesan. Dalam penelitian Batubara *et al.* (2021) yang berjudul

“Analysis Performance Bcrypt Algorithm to Improve Password Security from Brute Force” mengungkapkan bahwa algoritma *bcrypt* adalah fungsi *hashing* yang dibuat dari algoritma *blowfish* oleh dua peneliti keamanan komputer, Niels Provos dan David Mazieres. Fungsi *hashing* ini memiliki beberapa keunggulan yang menggunakan *salt random* (salt adalah urutan penambahannya ke kata sandi untuk membuatnya lebih sulit untuk di-brute force).



Gambar 5. Proses Pengamanan Password Menggunakan Algoritma Bcrypt

Berdasarkan Gambar 5, dapat disimpulkan bahwa *bcrypt* memiliki fungsi sangat penting dalam proses *login*, yaitu ketika *user* melakukan pendaftaran akun. *Password* yang *user* masukan pada sistem tersebut secara otomatis dirubah berbentuk string panjang yang random oleh *bcrypt* yang nantinya masuk dalam *database* sehingga akan mengamankan *password user* dan hanya *user* yang dapat mengetahui *password* tersebut. Selain itu, untuk menjaga keamanan data *user* dari pelaku kejahatan.

2.5 Unified Modeling Language (UML)




Unified Modelling Language (UML) merupakan bahasa pemodelan dalam pengembangan sistem yang terintegrasi objek dalam pengembangan sistem

perangkat lunak (Devianty & Nur Ibrahim, 2021). Aulia (2022), mengungkapkan bahwa tujuan utama UML yaitu untuk membantu tim pengembang dalam berkomunikasi, mengeksplorasi desain, dan memvalidasi desain arsitektur perangkat lunak. Dalam buku yang berjudul *UML Diagramming a Case Study Approach* menjelaskan bahwa UML adalah *diagram* untuk mendesain, membantu dalam pemahaman yang lebih baik tentang perangkat lunak atau sistem yang akan dikembangkan. Diagram UML seperti *Use Case Diagram*, *Activity Diagram*, *Class Diagram*, dan *Sequence Diagram* (Sundaramoorthy, 2022).

1. Use Case Diagram

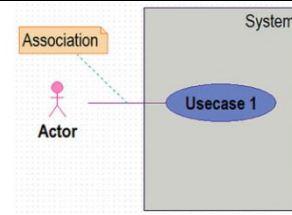
Use Case Diagram dapat mendeskripsikan interaksi antara *user* dengan sistem (Sundaramoorthy, 2022).

Tabel 1. Komponen Use Case Diagram

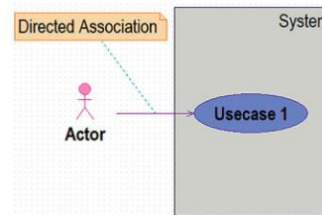
No	Komponen	Notasi UML	Tujuan
1.	<i>System Boundary</i>		Mewakili ruang lingkup sistem.
2.	<i>Actors</i>		Pengguna (orang atau organisasi) yang berinteraksi dengan sistem.
3.	<i>Use Case</i>		Fungsionalitas yang disediakan <i>system</i> untuk saling bertukar pesan antar unit atau <i>actor</i> .

Tabel 2. Relasi Use Case Diagram

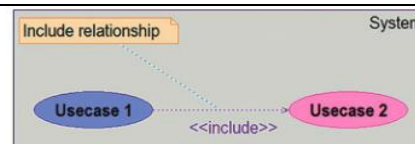
No	Relasi Use Case	Notasi UML dan Fungsinya
----	-----------------	--------------------------

1. *Association*


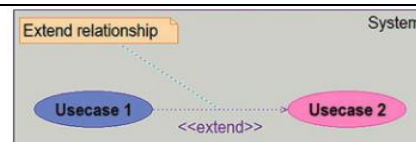
Menghubungkan antara aktor dan *use case*.

2. *Directed Association*


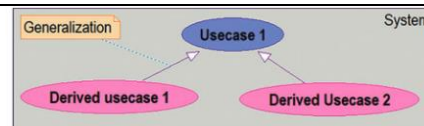
Hubungan satu arah di mana aktor mengambil tanggung jawab dalam mempengaruhi *use case*.

3. *Include*


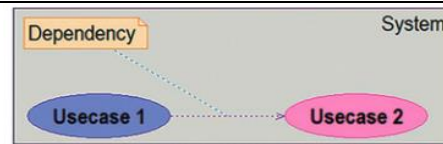
Hubungan satu *use case* menyertakan *fungsi*ionalitas dari satu *use case* lainnya.

4. *Extend*


Memperluas hubungan antara dua *use case*.

5. *Generalization*


Hubungan satu *use case* menyertakan *fungsi*ionalitas dari satu *use case* lainnya.

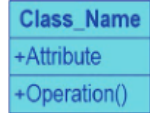

6. *Dependency*

Hubungan satu *use case* tergantung pada keberadaan *use case* lain.

2. *Class Diagram*

Class diagram menggambarkan struktur sistem dalam hal kelas dan objek. *Class diagram* ini sangat cocok diimplementasikan pada suatu *project* yang menggunakan konsep *object-oriented* karena *class diagram* mudah untuk digunakan (Sundaramoorthy, 2022).

Tabel 3. Komponen *Class Diagram*

No	Komponen	Notasi UML dan Tujuan
1.	<i>Class</i>	<ul style="list-style-type: none"> Kompartemen atas nama kelas Kompartemen tengah struktur kelas (atribut) Kompartemen bawah perilaku kelas (operasi) 
2.	<i>Class Relationships</i>	
	<i>Association</i>	Digunakan untuk menghubungkan kedua <i>class</i> . 
	<i>Directed Association</i>	Dua kelas terhubung dengan komunikasi satu arah, mereka terhubung dengan asosiasi terarah.



Dependency Satu kelas bergantung pada kelas lain, maka mereka terhubung menggunakan ketergantungan.



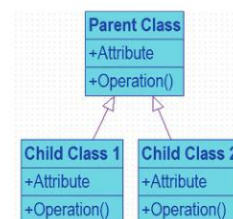
Aggregation Hubungan di mana kelas anak bisa terlepas dari kelas induknya.



Composition Hubungan di mana kelas anak tidak dapat berdiri sendiri dari kelas induk.

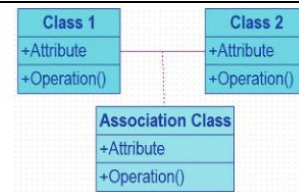


Generalization Hubungan antar kelas induk dan kelas anak.



3. *Association Class*





Tidak hanya menghubungkan satu set pengklasifikasi tetapi juga satu set hubungan itu sendiri.

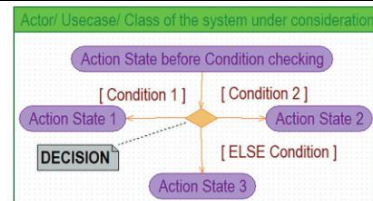


3. Activity Diagram

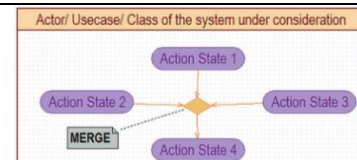
Activity diagram berfokus dalam pembuatan *activity diagram* hanya dapat dipakai untuk menggambarkan alur kerja atau aktivitas sistem (Sundaramoorthy, 2022).

Tabel 4. Komponen Activity Diagram

No	Komponen	Notasi UML dan Tujuan
1.	<i>Initial state</i>	 <p>Keadaan awal dari sistem.</p>
2.	<i>Final state</i>	 <p>keadaan penghentian sistem.</p>
3.	<i>Swimlanes</i>	 <p>Memisahkan organisasi yang bertanggung jawab terhadap aktivitas yang terjadi.</p>
4.	<i>Action state</i>	 <p>Suatu operasi atau proses.</p>

5. *Decision*

Percabangan jika ada pilihan aktivitas lebih dari satu.

6. *Merge*

Menggabungkan arus input.

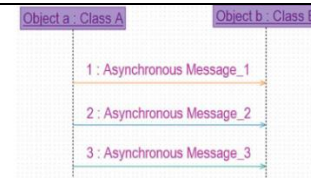
4. *Sequence Diagram*

Sequence diagram adalah sebuah diagram yang digunakan untuk menjelaskan dan menampilkan interaksi antar objek-objek dalam sebuah sistem secara terperinci (Sundaramoorthy, 2022).

Tabel 5. Komponen *Sequence Diagram*

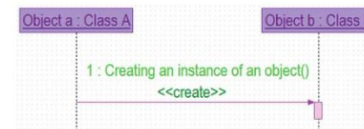
No	Komponen	Notasi UML dan Tujuan
1.	<i>Synchronous Message</i>	<p>Pesan yang dikirim dari pengirim berdasarkan semantik tunggu.</p>
2.	<i>Return Message</i>	<p>Pesan balasan untuk pesan <i>synchronous</i>.</p>

3. *Asynchronous Message*



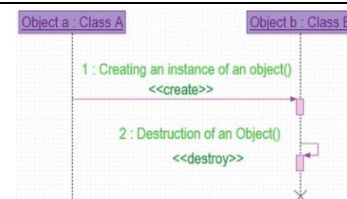
Pesan dikirim oleh pengirim ke penerima yang tidak perlu menunggu semua proses selesai.

4. *Create Message*



Membuat pesan

5. *Destroy Message*



Menghapus pesan.

2.6 Postman

Postman merupakan sebuah aplikasi yang digunakan untuk melakukan uji coba API yang telah dibuat. Menggunakan *tools postman* dapat menampilkan hasil dari HTTP *request* dengan cepat dan kompleks (Wintana dkk., 2022). *Postman* biasanya digunakan untuk menguji berbagai *web service* termasuk API. Pengujian yang dilakukan dengan mengakses *resources* untuk mengetahui sebuah keamanan API yang telah dibangun (Tedyyana dkk., 2021). Sementara menurut Kore (2022) *postman* adalah salah satu alat otomatisasi dan dokumentasi API terbesar yang tersedia saat ini. *Postman* menjadi solusi pengujian API lengkap yang digunakan oleh 5 juta pengembang dan lebih dari 100.000 perusahaan di seluruh dunia sehingga *postman* menjadi pengujian API yang canggih, kuat, dan serbaguna.

Terdapat empat REST *request*:

1. *Method*

Klien berkomunikasi dengan server, menunjukkan tindakan dari metode permintaan HTTP. Dalam jenis metode termasuk *GET*, *PUT*, *POST*, dan *DELETE*. Dapat melakukan operasi CRUD pada data yaitu, C untuk *Create*, R untuk *Read*, U untuk *Update*, dan D untuk *Delete*. *Create-POST Read- GET Update-PUT Delete- DELETE*.

2. Endpoint

Jika menjalankan *query* di server pengembangan, titik akhir URL adalah server pengembang.

3. Path Parameter / Query Parameter

Parameter adalah informasi tambahan yang dikirimkan ke server melalui *request parameter*. Parameter ini terkandung dalam URL.

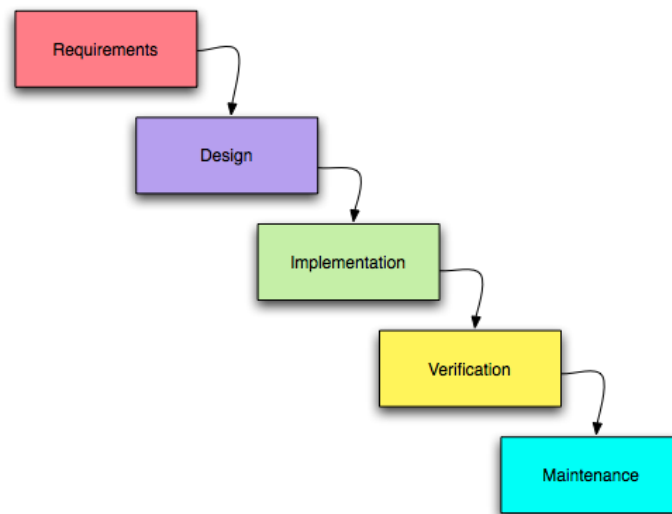
4. Headers

Permintaan atau *respons* HTTP. *Header* ditampilkan di tab *header postman*.

Berdasarkan beberapa pendapat tentang *postman*, maka dapat disimpulkan bahwa *postman* merupakan salah satu *tools* yang cocok untuk pengujian API secara cepat dan kompleks. *Tools postman* sangat *powerfull* dan mudah dipahami oleh pemula sekalipun. *Tools* ini tersedia secara gratis dan dapat berjalan pada sistem operasi *Windows*, *Linux*, dan *macOS*. Pengujian API ini bertujuan untuk mengetahui apakah hasil *response* sesuai dengan permintaan HTTP *request*.

2.7 Metode Pengembangan Waterfall

Metode Pengembangan pada penelitian ini menggunakan metode pengembangan sistem, yaitu *Waterfall*. Dalam buku *Software Engineering* yang berjudul “*A Practitioner's Approach*” yang ditulis oleh Roger S. Pressman, *waterfall* adalah salah satu pendekatan tradisional dalam pengembangan perangkat lunak yang terstruktur. Metode *waterfall* dijelaskan dengan langkah-langkah *Requirements*, *Design*, *Implementation*, *Verification*, dan *Maintenance* (Pressman, 2012).



Gambar 6. Metode *Waterfall*

Tahap-tahap dalam metode *waterfall* adalah sebagai berikut:

1. *Requirements* (Kebutuhan)

Tahap ini fokus utamanya adalah pengumpulan, analisis, dan dokumentasi kebutuhan pengguna dan sistem yang akan dikembangkan. Tujuannya untuk memahami secara menyeluruh kebutuhan yang mencakup deskripsi lengkap yang harus dicapai oleh sistem.

2. *Design* (Desain)

Pada tahap ini, Setelah kebutuhan dikumpulkan dilakukan tahap desain keseluruhan sistem dan komponen-komponennya berdasarkan kebutuhan yang telah ditetapkan. Desain arsitektur, desain antarmuka, dan desain lainnya dilakukan dalam tahap ini. Tujuan dari tahap ini adalah untuk merancang solusi dalam memenuhi kebutuhan pengguna dengan memperhatikan aspek teknis.

3. *Implementation* (Implementasi)

Tahap ini melibatkan penulisan kode dan pengembangan perangkat lunak berdasarkan desain yang telah dibuat sebelumnya. *Developer* akan menerjemahkan desain menjadi kode yang dapat dieksekusi.

4. *Verification* (Verifikasi)

Tahap ini melibatkan pengujian perangkat lunak untuk memastikan bahwa sistem berfungsi sesuai dengan kebutuhan yang telah ditetapkan. Pengujian dilakukan pada berbagai level, mulai dari pengujian unit untuk menguji komponen individu hingga pengujian integrasi dan pengujian sistem secara keseluruhan. Tujuan dari tahap ini adalah untuk memastikan kualitas dan keandalan sistem.

5. *Maintenance* (Pemeliharaan)

Setelah implementasi dan verifikasi, tahap pemeliharaan dimulai. Pada tahap ini, sistem diperbaiki, ditingkatkan, dan dikelola untuk memenuhi perubahan kebutuhan pengguna atau memperbaiki *bug* yang ditemukan selama penggunaan sistem. Tujuan dari tahap ini adalah untuk menjaga kinerja sistem dan memenuhi kebutuhan yang muncul seiring waktu. Tahap-tahap ini dijalankan secara berurutan, di mana setiap tahap bergantung pada penyelesaian tahap sebelumnya.

2.8 Penelitian Terkait

Terdapat penelitian terkait yang perlu dipertanggungjawabkan dalam hal akademis, berikut ini penelitian terkait sudah pernah dilakukan oleh peneliti sebelumnya dan peneliti jadikan referensi dalam penelitian yang peneliti lakukan disajikan oleh Tabel 6.

