

BAB II LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam hal ini yang menjadi permasalahan pada obyek penelitian yaitu PT Rezeki Mekanik Indonesia. Bagaimana melakukan perancangan yang tepat untuk masalah tersebut. Untuk mendukung penelitian ini berikut adalah beberapa penelitian/pustala/jurnal yang relevan dengan topik judul yang dibawakan.

Tabel 2.1 Tinjauan pustaka

No	Judul Penelitian	Penulis	Tahun	Teknologi	Hasil Penelitian
1	Penerapan <i>Rest Api</i> Dalam Perancangan Aplikasi Reservasi Perawatan dan Penitipan Hewan Berbasis Android	(Dewantoro and Waluyo, 2023)	2023	<i>Rest API</i>	<i>REST API</i> pada sistem reservasi hewan menggunakan metode HTTP <i>GET</i> , <i>POST</i> , <i>PUT</i> , dan <i>DELETE</i> , memfasilitasi integrasi antara aplikasi android dan website.
2	Implementasi Representatif <i>State Transition Application Programming Interface (Rest Api)</i> Pada Aplikasi Tip.In Berbasis Android	(Rivando, 2023)	2023	<i>Rest API</i>	Dari hasil penelitian, disimpulkan bahwa aplikasi bertujuan memberikan informasi berguna dan rekomendasi tempat percetakan kepada pengguna layanan jasa pemesanan percetakan.
3	Perancangan Sistem Informasi Penyewaan <i>Thermoking</i> Pada PT. Modern Prima	(Irmayanti, 2023)	2023	Flask, dan Python	Sistem yang dibangun dengan <i>Framework</i> Flask mampu memudahkan proses penyewaan dan dapat menghasilkan laporan transaksi secara <i>real-time</i> .

No	Judul Penelitian	Penulis	Tahun	Teknologi	Hasil Penelitian
	Transportasi Menggunakan Python Dengan <i>Framework</i> Flask				
4	<i>Application Of Random Forest Algorithm On Watch Price Prediction System Using Framework Flask</i>	(Dalimun the and Hakim, 2023)	2023	Flask	Penelitian "Penerapan Algoritma <i>Random Forest</i> pada Sistem Prediksi Harga Jam Tangan dengan <i>Framework</i> Flask" menemukan bahwa jam tangan Casio mendominasi penjualan (36%), sementara Citizen hanya 1%. Jenis jam tangan Quartz sangat populer (1357 produk), sedangkan Pilot hanya 3 produk
5	Implementasi Sistem <i>Reminder</i> Jadwal pada <i>eLearning Moodle</i> Berbasis <i>API</i> Menggunakan <i>Framework</i> Flutter	(Putra and Kurniawan, 2023)	2023	Flask dan Flutter	Penelitian ini berhasil mengimplementasikan sistem <i>reminder</i> jadwal pada platform <i>eLearning Moodle</i> dengan memanfaatkan <i>API</i> , menggunakan <i>framework</i> Flutter. Hasilnya menunjukkan bahwa aplikasi <i>reminder</i> ini dapat berjalan efisien dengan <i>running time</i> di bawah 2 detik dan penggunaan sumber daya <i>minimal</i> .
6	Implementasi <i>Rest Api</i> Pada Aplikasi E-Perpus (Studi Kasus: SMA Gama Yogyakarta)	(Bukhori and Artika, 2023)	2023	<i>REST API</i> CRUD	Hasil penelitian ini menyarankan penggunaan sistem perpustakaan berbasis <i>smartphone</i> dan <i>Rest Api</i> CRUD untuk memudahkan pemantauan dan pengelolaan buku oleh petugas dan pengguna.
7	Klasifikasi Jenis Anggur Berdasarkan Bentuk Daun Menggunakan Convolutional Neural Network Dan	(Pratiwi, 2023)	2023	Flask	Penelitian ini menyimpulkan bahwa penggunaan Convolutional Neural Network (CNN) dan K-Nearest Neighbor (K-NN) untuk mengklasifikasikan jenis daun anggur melalui aplikasi web berbasis Flask

No	Judul Penelitian	Penulis	Tahun	Teknologi	Hasil Penelitian
	K-Nearest Neighbor				menunjukkan bahwa model CNN memiliki tingkat akurasi yang sangat tinggi (99%), sedangkan model KNN memiliki tingkat akurasi yang lebih rendah (53%).
8	Integrasi Sistem Informasi Pemantauan Kualitas Lingkungan Air Dan Udara Menggunakan <i>Rest Api</i> Dan <i>Web Service</i>	(Salim and Wahjono, 2021)	2021	<i>REST API</i> dan JSON	Mengintegrasikan <i>database</i> pemantauan lingkungan air dan udara menggunakan <i>REST API</i> berbasis JSON, memudahkan pengelolaan data dan pemantauan <i>real-time</i> .
9	Implementasi Flask <i>Framework</i> pada Pembangunan Aplikasi <i>Purchasing Approval Request</i>	(Ningtyas and Setiyawati, 2021)	2021	Flask	Implementasi Flask memungkinkan pembangunan aplikasi <i>Purchasing Approval Request</i> yang optimal dan efisien sesuai kebutuhan dengan kemudahan penggunaan dekorator dan library serta peningkatan komunikasi data antara <i>backend</i> dan <i>frontend</i> .
10	Implementasi <i>Api Master Store</i> Menggunakan Flask, <i>Rest</i> Dan <i>Orm</i> Di PT. XYZ	(Putra and Susetyo, 2020)	2020	Flask dan ORM	Mempermudah perpindahan arsitektur sistem, memberikan keamanan dengan fitur alias kolom, serta menggunakan metode ORM dan <i>Framework</i> Flask.

Tinjauan pustaka yang terkait dengan penelitian "Implementasi Aplikasi *Backend Rest Api Platform E-Learning* Dokter Mekanik Academy Menggunakan Flask" memaparkan sejumlah aspek yang relevan dengan penelitian. Dari sepuluh tinjauan pustaka yang berkaitan, terdapat perbedaan signifikan yang dapat dijabarkan. Beberapa penelitian, seperti "Penerapan *Rest Api* Dalam Perancangan Aplikasi Reservasi Perawatan dan Penitipan Hewan Berbasis Android" serta "Implementasi Representatif *State Transition Application Programming Interface (Rest Api)* Pada Aplikasi Tip.In Berbasis Android," lebih menekankan pada

integrasi *REST API* pada aplikasi berbasis Android dengan fokus pada reservasi hewan dan rekomendasi tempat percetakan. Sebaliknya, penelitian "Perancangan Sistem Informasi Penyewaan *Thermoking* Pada PT. Moderen Prima Transportasi Menggunakan Python Dengan *Framework* Flask" dan "Implementasi Flask *Framework* pada Pembangunan Aplikasi *Purchasing Approval Request*" lebih mengarah pada pengembangan sistem informasi penyewaan dan aplikasi pengajuan pembelian dengan memanfaatkan Flask untuk perancangan yang optimal.

Ada juga penelitian yang memusatkan perhatian pada prediksi harga jam tangan dengan algoritma *Random Forest* dalam "*Application Of Random Forest Algorithm On Watch Price Prediction System Using Framework* Flask" serta pemanfaatan Flutter dalam pembangunan sistem *reminder* jadwal pada *eLearning Moodle* dalam penelitian "Implementasi Sistem *Reminder* Jadwal pada *eLearning Moodle* Berbasis *API* Menggunakan *Framework* Flutter." Dalam konteks klasifikasi, penelitian "Klasifikasi Jenis Anggur Berdasarkan Bentuk Daun Menggunakan Convolutional Neural Network Dan K-Nearest Neighbor" mengeksplorasi penggunaan CNN dan K-NN untuk mengklasifikasikan jenis daun anggur melalui aplikasi web berbasis Flask dengan fokus pada akurasi model.

Sementara itu, "Integrasi Sistem Informasi Pemantauan Kualitas Lingkungan Air Dan Udara Menggunakan *Rest Api* Dan *Web Service*" berfokus pada integrasi *database* pemantauan lingkungan menggunakan *REST API* berbasis JSON untuk pemantauan *real-time*. Selanjutnya, "Implementasi *Api Master* Store Menggunakan Flask, *Rest* Dan *Orm* Di PT. XYZ" lebih menyoroti implementasi *API* pada toko dengan fitur ORM dan keamanan. Dari kesepuluh tinjauan pustaka ini, perbedaan fokus pada penggunaan *REST API*, implementasi Flask, pengembangan aplikasi, prediksi, klasifikasi, dan integrasi sistem memberikan wawasan yang beragam dan relevan untuk mendukung penelitian penulis dalam membangun aplikasi *backend REST API* untuk platform *e-learning*.

2.2 Landasan Teori

2.2.1 *E-Learning*

E-Learning merupakan sebuah inovasi baru yang memiliki kontribusi sangat besar terhadap perubahan proses, dimana proses belajar tidak lagi hanya mendengarkan uraian materi dari tenaga pendidik secara langsung tetapi peserta didik juga melakukan aktivitas lain seperti mengamati, melakukan, mendemonstrasikan dan lain-lain. Materi bahan ajar divisualisasikan dalam berbagai bentuk yang lebih dinamis interaktif sehingga peserta didikan termotivasi untuk terlibat lebih jauh dalam proses tersebut. *E-Learning* merujuk pada penggunaan teknologi internet untuk mengirimkan serangkaian solusi yang dapat meningkatkan pengetahuan keterampilan. Ada pula yang menafsirkan *e-learning* sebagai bentuk pendidikan jarak jauh yang dilakukan melalui media internet (Aminoto, 2014).

Pembelajaran melalui *e-learning* membantuk penguasaan materi dengan indeks 84%. Kedua interaktifitas pembelajaran melalui *e-learning* meningkatkan motivasi/semangat belajar dengan indeks 71%. Interaktifitas pembelajaran melalui sistem *e-learning* memberikan rasa nyaman/kedekatan antara dosen dengan mahasiswa dan sesama mahasiswa dengan skor 66%. Keempat, interaktifitas pembelajaran melalui sistem *e-learning* memberikan rasa senang/puas dengan skor 70%. Jumlah rata-rata interaktifitas sistem *e-learning* ini adalah 73% dengan katagori cukup bagus. (Yusuf, 2023)

Kemudian dengan adanya inovasi sistem atau metode pembelajaran ini, peserta didik akan lebih terpacu dan bersemangat untuk belajar mandiri. Pengukuran kelayakan *e-learning* tersebut dapat dilihat dari penilaian ahli media yang mendapatkan nilai rerata skor 80,00% dengan kriteria layak untuk digunakan. Penilaian dari uji coba kelompok kecil mendapatkan skor rerata 77,30%. Berdasarkan hasil tersebut dapat disimpulkan bahwa *e-learning* berbasis *moodle* ini layak digunakan. (Setiawan, Mansur and Mastur, 2020)

2.2.2 Backend

Backend sistem merupakan bagian yang bertanggung jawab dalam mengelola aplikasi dan *database* agar dapat saling berkomunikasi dengan baik dan lancar guna mendukung antarmuka sistem bekerja sesuai dengan fungsinya (Prayoga dan Permadi, 2022). Sistem *backend* mengirim dan menerima *permintaan* dalam bentuk JSON, sehingga setiap parameter yang dikirimkan harus dalam bentuk JSON. Setiap prosedur pemanggilan fungsi pada *REST API* perlu mendefinisikan parameter *dataType* dengan bentuk JSON (Ramdhany *et al.*, 2021).

2.2.3 Python

Bahasa pemrograman Python, yang sangat populer saat ini, pertama kali ditemukan oleh Guido van Rossum di Stichting Mathematisch Centrum (CWI), Amsterdam pada tahun 1989. Python adalah bahasa pemrograman yang menggunakan *interpreter* untuk menjalankan kode programnya. *Interpreter* tersebut dapat menerjemahkan kode secara langsung, dan Python dapat dijalankan di berbagai platform seperti Windows, Linux, dan lain-lain. Python mengadopsi paradigma pemrograman dari beberapa bahasa lain, termasuk paradigma pemrograman prosedural seperti bahasa C, pemrograman berorientasi objek seperti Java, dan bahasa fungsional seperti Lisp. Kombinasi paradigma ini memudahkan para programmer dalam mengembangkan berbagai proyek menggunakan Python (Rizal, Kharisma dan Fahrurrozi, 2021).

2.2.4 Flask

Flask adalah sebuah *web framework* yang ditulis dengan Bahasa Python dan tergolong sebagai jenis *micro framework*. Flask berfungsi sebagai kerangka kerja aplikasi dari suatu web. Pengembang dapat membuat aplikasi yang terstruktur Flask dan bahasa python. Flask sendiri dapat mempermudah pengembangan website karena *framework* ini mudah untuk dipahami dan

fiturnya yang sederhana namun dapat melakukan pembangunan aplikasi basis web yang cukup rumit (Irmayanti, 2023).

Flask dirancang dan didesain untuk memulai dengan cepat dan mudah (berfungsi untuk meminimalisir waktu *load*), dengan kemampuan ini dapat meningkatkan kemampuan aplikasi yang kompleks. Flask termasuk pada jenis *micro framework* karena tidak memerlukan suatu alat atau pustaka tertentu dalam penggunaannya. Sebagian besar fungsi dan komponen umum seperti validasi form, *database*, dan sebagainya tidak terpasang secara *default* di Flask. Hal ini dikarenakan fungsi dan komponen-komponen tersebut sudah disediakan oleh pihak ketiga dan Flask dapat menggunakan ekstensi yang membuat fitur dan komponen-komponen tersebut seakan diimplementasikan oleh Flask sendiri (Santoso dan Saian, 2023).

Python dan *framework* Flask mampu memenuhi kriteria yang diharapkan oleh pengguna. Dengan adanya pengurangan biaya pemeliharaan aplikasi, PT. XYZ dapat mengalokasikan anggaran untuk pengembangan aplikasi lain atau kebutuhan bisnis lainnya yang lebih produktif. *framework* Flask dapat menjadi solusi untuk aplikasi berbasis web menggunakan Python, namun, penelitian yang dilakukan ini lebih fokus pada pengembangan aplikasi *Distribution Center System* serta mengatasi masalah biaya pemeliharaan aplikasi dan kemampuan aplikasi untuk beroperasi di berbagai platform. (Evan and Saian, 2023)

2.2.5 Web Service

Web service adalah kumpulan dari fungsi dan metode yang terdapat pada sebuah *server* dan dapat dipanggil oleh klien secara *remote*. *Web service* dirancang untuk mendukung interoperabilitas *machine-to-machine* dan berinteraksi melalui jaringan (Septiani *et al.*, 2020). *Web service* mempunyai alat penghubung yang diuraikan di dalam format *machine-processable* (secara spesifik WSDL). Sistem lain saling berhubungan dengan *Web service* di dalam

cara yang ditentukan oleh deskripsinya yang menggunakan pesan SOAP, secara khas disampaikan menggunakan HTTP dengan XML *serialization*, bersama dengan standar lain yang terkait dengan web (Suyanto, 2007).

Format data yang sering digunakan dalam web *service backend* adalah JSON (JavaScript Object Notation) (Saputra dan Witriyono, 2022). JSON dirancang untuk memudahkan pertukaran data pada situs dan merupakan perluasan dari fungsi-fungsi javascript. JSON tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer, seperti C, C++, C#, Java, JavaScript, Perl, Python dan lain-lain. Oleh karena memiliki sifat-sifat tersebut, JSON ideal digunakan untuk melakukan pertukaran data (Mangunwijaya et al., 2015).

2.2.6 HTTP dan HTTPS

Hypertext Transfer Protocol (HTTP) adalah sebuah protokol tingkat aplikasi untuk distribusi serta kolaborasi, sistem informasi *hypermedia*. HTTP telah digunakan oleh informasi *World Wide Web global* dimulai sejak tahun 1990. HTTP adalah pilihan yang paling populer untuk transportasi layanan. HTTP bersifat sederhana, stabil, dan banyak digunakan. Selain itu, sebagian *firewall* telah mengizinkan lalu lintas HTTP (Lumbantobing et al., 2021).

HTTP over TLS atau *Hypertext Transfer Protocol Secure* (HTTPS) adalah protokol yang paling sering digunakan oleh pengguna internet, HTTPS adalah protokol yang mengamankan lalu lintas HTTP untuk mencegah pihak ketiga menyadap atau mengubah konten yang dikembalikan sebagai HTTP *response* (Winawang, 2021).

HTTP dan HTTPS memiliki metode yang sama, berikut adalah beberapa metode untuk berkomunikasi :

- a. *GET*, meminta data dari *server* berdasarkan URL. Hanya mengambil data, tanpa mengubahnya.

- b. *POST*, mengirim data ke *server* untuk diproses, biasanya digunakan pada formulir web. Memungkinkan pengiriman data yang tidak terlihat di URL.
- c. *PUT*, mengganti atau memperbarui sumber daya di *server* dengan data dari klien.
- d. *DELETE*, menghapus sumber daya tertentu dari *server* sesuai permintaan.
- e. *PATCH*, memperbarui sebagian kecil dari sumber daya di *server*, berbeda dengan *PUT* yang mengganti seluruh sumber daya.

2.2.7 REST API

REST terdiri dari arsitektur web *service* dengan basis *http verbs* untuk mengolah data. Sedangkan *Application Programming Interface (API)* adalah aplikasi yang memungkinkan para *developer* dalam memberi persetujuan dan mengintegrasikan 2 (dua) buah aplikasi yang berbeda dalam satu waktu yang bertujuan untuk menjadi penghubung. *REST API* adalah salah satu desain arsitektur yang ada dalam *API* tersebut. Sedangkan cara kerja dari *REST API* yaitu *REST Client* yang melakukan akses pada data/*resource REST server* pada masing-masing *resource*. Data tersebut dapat dibedakan menggunakan *global id* atau URIs (*Universal Resource Identifiers*) (Hertantyo et al., 2021).

Pada penelitian yang sebelumnya yang berjudul “Implementasi RESTful Web *Service* pada Sistem Informasi Donor ASI Terintegrasi di Indonesia”. Penerapan *REST API* pada sistem informasi donor ASI dirancang untuk mempermudah agen dalam melakukan proses integrasi dan validasi data. Penelitian ini dapat dikembangkan lebih lanjut dengan lebih mengoptimalkan proses pertukaran data menggunakan *REST API* mengingat proses pertukaran data dilakukan oleh agen di seluruh Indonesia. (Badieah et al., 2022).

Penelitian lainnya juga yang berjudul (Salim and Wahjono, 2021) “Integrasi Sistem Informasi Pemantauan Kualitas Lingkungan Air Dan Udara Menggunakan *Rest Api* Dan Web *Service*” Berhasil mengintegrasikan *database*

pemantauan lingkungan air dan udara menggunakan *REST API* berbasis JSON, memudahkan pengelolaan data dan pemantauan *real-time*.

2.2.8 Database

Relational Database merupakan salah satu jenis atau tipe dari basis data yang digunakan untuk menyimpan dan menyediakan akses ke data lainnya yang terkait antara satu sama lain. Dalam penggunaannya *relational database* akan memberikan id unik pada setiap baris data yang tersimpan dalam tabelnya, dan setiap baris data atau *record* biasanya mempunyai nilainya sendiri yang dapat memudahkan pembentukan relasi antar data pada tabel lainnya. *Relational database* biasanya dikelola dengan menggunakan perangkat lunak untuk dapat melihat data apa saja yang terdapat dalam basis data tersebut (Engel dan Lie, 2022).

Studi "Perbandingan Performansi Antara MySQL dan PostgreSQL" (Praba and Safitri, 2020) menyoroti pentingnya sistem manajemen basis data (DBMS) dalam berbagai aspek, baik dalam skala besar maupun kecil. Basis data umumnya digunakan oleh organisasi untuk menyimpan informasi terkait. Penelitian ini berfokus pada perbandingan performansi antara dua sistem manajemen basis data relasional populer, yaitu MySQL dan PostgreSQL.

Sementara itu, dalam studi "Penerapan *Database Management* pada Apotek XYZ Menggunakan Aplikasi Microsoft Access" (Sibarani, Zikri and Darseni, 2023), perancangan sistem *Database Management* dilakukan dengan menggunakan pendekatan Microsoft Access. Tujuannya adalah untuk mengorganisir data sesuai dengan kriteria tertentu, memahami penggunaan basis data dengan baik, dan menggunakan Microsoft Access dalam pembuatan basis data. Hasil akhirnya adalah desain manajemen basis data yang dikhususkan untuk PT. XYZ. Studi ini memberikan wawasan tentang bagaimana perusahaan atau organisasi skala kecil hingga menengah dapat

memanfaatkan teknologi basis data yang lebih sederhana dan biaya yang lebih efisien.

2.2.9 Visual Studio Code

Visual Studio Code adalah *text editor* yang dikembangkan oleh Microsoft untuk Windows, Linux dan MacOS. Didalamnya sudah terdapat dukungan untuk *debugging*, kontrol Git atau terminal bawaan, *syntax highlighting*, *intelligent code completion*, *snippets*, dan *refactoring code*. Visual Studio Code juga dapat di *customizable*, sehingga pengguna dapat mengubah tema editor, *keyboard shortcuts* dan *preferences*. Visual Studio Code merupakan aplikasi *text editor* yang *open source* dirilis di bawah the permissive MIT License. Dalam survei pengembang Stack Overflow 2019, Visual Studio Code mendapat peringkat *the most popular developer environment tool*, dengan 50,7% dari 87.317 responden mengklaim menggunakannya (Pratama and Pertiwi, 2022).

2.2.10 Docker

Docker adalah sebuah aplikasi yang berbasiskan teknologi *open source* yang memungkinkan *developer* atau siapapun untuk membuat, menjalankan, melakukan percobaan dan meluncurkan aplikasi di dalam sebuah *container*. Docker membuat proses pemaketan aplikasi bersama komponennya secara cepat dalam sebuah *container* yang terisolasi, sehingga dapat dijalankan dalam infrastruktur lokal tanpa melakukan perubahan konfigurasi pada *container*. Docker juga sangat ringan dan cepat jika dibandingkan dengan mesin *virtual* yang berbasis *hypervisor* (Dwiyatno *et al.*, 2020).

Berikut adalah keunggulan yang diberikan oleh aplikasi docker.

- a. Portabilitas: Kontainer dapat dijalankan di berbagai platform.
- b. Isolasi: Memisahkan aplikasi dari lingkungan lain untuk kestabilan.

- c. Efisiensi: Menggunakan sumber daya lebih sedikit dibandingkan *virtual machine*.
- d. Skalabilitas: Memungkinkan penyebaran dan penskalaan aplikasi dengan mudah.
- e. Manajemen Mudah: Memfasilitasi manajemen aplikasi dan pembaruan dengan cepat.
- f. Cepat dan Ringan: Kontainer dapat mulai dan berhenti dengan cepat

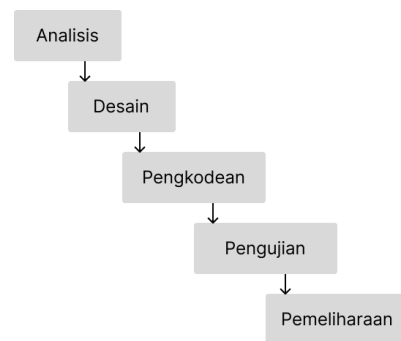
2.2.11 *Postman*

Postman adalah *platform* kolaborasi untuk pengembangan *API*. Dibuat oleh Abhinav Asthana, seorang programmer dan desainer yang berbasis di Bangalore, India, *Postman* memudahkan dalam menguji, mengembangkan, dan mendokumentasikan *API*. Fitur *Postman* yang sederhana membuat pengujian *API* dapat dilakukan dengan baik dan cepat. Cara kerja *Postman* dengan mengklasifikasi *request* berdasarkan *request method*, URL dan parameter parameter *request* (Sari dan Hidayat, 2022). *Postman* sering digunakan oleh *backend developer* untuk melakukan *testing REST API* dikarenakan beberapa hal berikut ini :

- a. Memudahkan pembuatan, manajemen, dan pengujian permintaan *API*.
- b. Mendukung permintaan HTTP seperti *GET*, *POST*, *PUT*, *DELETE*, dan lainnya.
- c. Memungkinkan penyesuaian nilai dinamis untuk pengujian dalam skenario yang berbeda.
- d. Kolaborasi tim dengan berbagi koleksi permintaan.
- e. Membuat dokumentasi rinci untuk memudahkan penggunaan oleh pengembang lain.

2.2.12 Metode Waterfall

Metode waterfall atau disebut juga dengan sekuensial linear merupakan metode pengembangan perangkat lunak yang bersifat sekuensial atau terurut dimulai dari analisis, desain, pengkodean, pengujian dan tahap pemeliharaan. Sifat dari metode waterfall yaitu sekuensial maksudnya adalah proses pada tahap kedua tidak bisa dilakukan sebelum proses tahap satu selesai, sehingga metode ini sangat mudah untuk dipahami (Bulman, 2017). Adapun tahapan Waterfall dapat dilihat pada gambar sebagai berikut.



Gambar 2.1 Metode waterfall

- a. Analisis, yaitu tahap di mana kebutuhan pengguna diidentifikasi untuk membuat perangkat lunak seperti apa yang akan dibutuhkan. Hasil dari analisis identifikasi ini nantinya akan didokumentasikan.
- b. Desain, yaitu tahap di mana melakukan proses desain arsitektur dari perangkat lunak yang akan dikembangkan. Tahapan ini diperlukan untuk tahapan selanjutnya yaitu pembuatan kode agar lebih mudah.
- c. Pengkodean, yaitu tahap di mana proses pengembangan perangkat lunak dilakukan. Pada proses ini pengkodean dilakukan berdasarkan dari tahapan sebelumnya yang sudah dilakukan yaitu analisis, dan desain.
- d. Pengujian, yaitu tahap di mana proses pengujian dari perangkat lunak yang sudah dikembangkan. Tahapan ini bertujuan untuk melihat apakah output program sudah sesuai yang diharapkan serta mengurangi terjadinya kesalahan atau *error*.


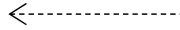





- e. Pemeliharaan, terkadang terjadi permasalahan ketika perangkat lunak telah digunakan oleh *end user* namun hal ini tidak terdeteksi ketiak dalam tahapan pengujian. Tahapan pemeliharaan ini bertujuan unuk memperbaiki jika terjadi permsalahan tersebut.

2.2.13 *Unified Modeling Language (UML)*

UML adalah merupakan sekumpulan alat yang biasanya digunakan untuk melakukan abstraksi terhadap sebuah sistem atau perangkat lunak berbasis objek. UML merupakan singkatan dari *Unified Modeling Language*. UML juga menjadi salah satu cara untuk mempermudah pengembangan aplikasi yang berkelanjutan. UML juga dapat menjadi alat bantu untuk transfer ilmu tentang sistem atau aplikasi yang akan dikembangkan dari satu *developer* ke *developer* lainnya (Noviantoro *et al.*, 2022). UML memiliki beberapa bagaian yaitu sebagai berikut.

- a. *Use Case Diagram*
 - 1) Menggambarkan hubungan antara aktor eksternal dan *use case* yang diberikan oleh sistem.
 - 2) Menyajikan fungsi sistem yang terlihat oleh aktor dari luar, bukan implementasi internal.

Tabel 2.2 Simbol *Use case* diagram

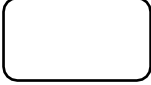




Gambar	Nama	Keterangan
	<i>Actor</i>	Simbol yang mewakili peran atau entitas yang berinteraksi dengan sistem. <i>Actor</i> adalah pengguna atau entitas eksternal yang terlibat dalam sebuah <i>use case</i> .
	<i>Association</i>	Hubungan antara objek-objek dalam diagram yang menggambarkan bagaimana mereka terkait satu sama lain.
	<i>System</i>	Menggambarkan paket atau komponen dalam sistem yang sedang dianalisis atau digambarkan dalam diagram.
	<i>Generalization</i>	Hubungan antara objek anak (<i>descendant</i>) dan objek induk (<i>ancestor</i>) yang menunjukkan bahwa objek anak mewarisi perilaku dan struktur data dari objek induk.
<<Include>>	<i>Include</i>	Menunjukkan bahwa <i>use case</i> sumber secara eksplisit mencakup <i>use case</i> target pada suatu titik tertentu.
<<Extend>>	<i>Extend</i>	Menunjukkan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
	<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	<i>Collaboration</i>	Representasi interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar daripada jumlah elemen individu, menciptakan sinergi.
	<i>Note</i>	Simbol yang digunakan untuk menambahkan keterangan atau catatan pada elemen-elemen dalam diagram.

b. *Class Diagram*

- 1) Memvisualisasikan struktur statis *class* di dalam sistem.

- 2) Menunjukkan hubungan antar *class* seperti asosiasi, dependensi, atau spesialisasi.
- c. *State* Diagram
- 1) Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu objek dari suatu *class*.
 - 2) Menyajikan keadaan yang menyebabkan perubahan *state* dan kejadian yang memicu perubahan tersebut.
- d. *Sequence* Diagram
- 1) Menggambarkan kolaborasi dinamis antara sejumlah objek.
 - 2) Menunjukkan rangkaian pesan yang dikirim antara objek dan interaksi antar objek pada titik tertentu dalam eksekusi sistem.
- e. *Collaboration* Diagram
- 1) Memvisualisasikan kolaborasi dinamis, serupa dengan *sequence* diagrams.
 - 2) Menyajikan objek dan hubungannya, lebih menekankan pada konteks daripada urutan waktu.
- f. *Activity* Diagram
- 1) Menggambarkan aliran aktivitas dalam suatu operasi atau proses.
 - 2) Digunakan untuk mendeskripsikan aktivitas yang terbentuk dalam suatu operasi atau dapat diterapkan pada *use case* atau interaksi.

Tabel 2.3 Simbol *activity* diagram

Gambar	Nama	Keterangn
	<i>Activity</i>	Representasi dari tindakan atau langkah yang harus dilakukan dalam sistem.
	<i>Action</i>	Mencerminkan eksekusi suatu tindakan dalam sistem.
	<i>Initial Node</i>	Titik awal atau awal dari proses dalam diagram.
	<i>Actifity Final Node</i>	Titik akhir atau akhir dari suatu aktivitas dalam diagram.
	<i>Fork Node</i>	Titik dalam aktivitas di mana aliran tunggal bercabang menjadi beberapa aliran yang berjalan secara bersamaan.

g. *Component Diagram*

- 1) Memvisualisasikan struktur fisik kode dari komponen sistem.
- 2) Menunjukkan komponen seperti *source code*, komponen biner, atau *executable* component beserta logika yang diimplementasikan.

h. *Deployment Diagram*

- 1) Menggambarkan arsitektur fisik perangkat keras dan perangkat lunak sistem.
- 2) Menunjukkan hubungan dan jenis ketergantungan antar komputer dan perangkat lunak pada nodes.

i. *Communication Diagram*

- 1) Memvisualisasikan interaksi dinamis antar objek.
- 2) Menunjukkan cara objek berkomunikasi dan berinteraksi satu sama lain.