BAB II

TINJAUAN PUSTAKA

2.1 Teori Dasar

2.1.1 *Software Development Life Cycle*

Software Development Life Cycle, yang diterjemahkan dalam Bahasa Indonesia sebagai siklus hidup pengembangan perangkat lunak. SDLC adalah pendekatan sistematis untuk merancang, mengembangkan, menguji, dan memelihara perangkat lunak. Pendekatan ini digunakan oleh pengembang perangkat lunak untuk memastikan bahwa proyek pengembangan perangkat lunak berjalan dengan efisien, mematuhi standar kualitas, dan memenuhi kebutuhan pengguna menurut (Kurniawan Farhurrahman et al., 2023). Berikut adalah beberapa tahapan utama dalam SDLC :

1. Perencanaan *(Planning) t*ahap perencanaan melibatkan pengumpulan persyaratan, analisis bisnis, dan perencanaan proyek. Tim pengembang dan pemangku kepentingan merumuskan tujuan proyek, mengidentifikasi sumber daya yang diperlukan, dan membuat rencana kerja yang jelas.
2. Analisis *(Analysis), p*ada tahap analisis, persyaratan bisnis dan pengguna dipelajari dengan cermat. Pengembang bekerja sama dengan klien untuk memahami kebutuhan perangkat lunak yang akan dikembangkan. Dokumen analisis kebutuhan (*requirement specification*) disusun sebagai panduan untuk tahap pengembangan berikutnya.

3. Desain *(Design),* tahap desain melibatkan merencanakan struktur perangkat lunak, antarmuka pengguna, *database*, dan elemen-elemen teknis lainnya. Desain ini mencakup pemilihan algoritma, struktur data, dan arsitektur sistem, hasilnya adalah desain rinci yang siap untuk implementasi.
4. Implementasi *(Implementation).* Di tahap implementasi, pengembang mulai menulis kode berdasarkan desain yang telah dibuat sebelumnya. Pemrograman dilakukan dengan mematuhi pedoman dan standar tertentu, proses pengkodean ini menghasilkan perangkat lunak yang dapat diuji.
5. Pengujian *(Testing),* pengujian adalah tahap di mana perangkat lunak diuji secara menyeluruh untuk memastikan bahwa itu berfungsi sebagaimana mestinya. Pengujian mencakup pengujian fungsional, pengujian kinerja, pengujian keamanan, dan pengujian kesalahan. Setiap *bug* atau masalah yang ditemukan selama pengujian diperbaiki sebelum perangkat lunak dilepaskan kepada pengguna akhir.
6. Pengimplementasian *(Deployment) s*etelah perangkat lunak diuji dan disetujui, itu diterapkan atau diimplementasikan di lingkungan produksi. Proses ini melibatkan pemasangan perangkat lunak di sistem pengguna akhir dan memastikan bahwa itu berjalan dengan benar.
7. Pemeliharaan *(Maintenance), s*etelah perangkat lunak diimplementasikan, pemeliharaan diperlukan untuk memastikan bahwa itu tetap berfungsi sebagaimana mestinya. Pemeliharaan melibatkan perbaikan *bug*, peningkatan kinerja, dan penambahan fitur baru sesuai kebutuhan pengguna.

Tahapan-tahapan dalam SDLC dapat bervariasi tergantung pada metodologi pengembangan yang digunakan, seperti model air terjun *(waterfall),* model *spiral*,

atau model pengembangan iteratif dan inkremental. Pemilihan metode tergantung pada kebutuhan proyek dan preferensi pengembang. Pada penelitian ini penulis menggunakan metode *agile* (Khawas & Shah, 2018).

2.1.2 Metode *Agile*

Agile adalah pendekatan pengembangan perangkat lunak yang menekankan kolaborasi, responsivitas terhadap perubahan, pengiriman perangkat lunak yang dapat digunakan, dan iterasi yang cepat. Metode *Agile* mengutamakan individu dan interaksi, perangkat lunak yang berfungsi, kolaborasi pelanggan, merespon perubahan, dan keberlanjutan.

Berikut adalah beberapa metodologi yang umum digunakan dalam pendekatan *Agile* (Kurniawan Farhurrahman et al., 2023).

1. *Scrum*

Deskripsi Singkat *Scrum* adalah kerangka kerja pengembangan *iteratif* dan *inkremental* yang menggunakan konsep *sprint* (*iterasi*) untuk menghasilkan perangkat lunak yang dapat digunakan dalam waktu singkat. Peran utama *Scrum Master* (memastikan tim mengikuti proses), *Product Owner* (mewakili kepentingan pelanggan), dan tim pengembangan (Fadiyah Ghina et al., 203 C.E.). Artefak utama *Product Backlog* (daftar kebutuhan pelanggan), *Sprint Backlog* (daftar tugas untuk *sprint* saat ini) dan *Increment* (hasil kerja yang dapat digunakan pada akhir *sprint*).

2. *Kanban*

Deskripsi Singkat *Kanban* adalah metode manajemen visual yang menggunakan papan *Kanban* untuk memvisualisasikan pekerjaan, mengidentifikasi *bottlenecks*, dan memastikan alur kerja yang lancar. Prinsip utama mengatur pekerjaan dalam kolom berbeda, mengontrol jumlah pekerjaan yang sedang

dikerjakan pada suatu waktu (*WIP limit)* dan memperbaiki alur kerja berdasarkan umpan balik visual.

3. *Extreme Programming* (XP)

Deskripsi singkat XP adalah pendekatan pengembangan perangkat lunak yang menekankan pengujian yang terus menerus, penggabungan perubahan kecil secara teratur, dan kolaborasi yang erat antara pengembang dan pelanggan. Praktik utama pengujian otomatis (*Unit Testing*), Integrasi *Continue* (*Continuous Integration)*, pengodean bersama (*Pair Programming*), desain bersama (*Collective Code Ownership*) dan perencanaan game (*Planning Game*).

4. *Lean Software Development*

Lean Software Development mengadopsi prinsip-prinsip dari *Lean Manufacturing* untuk mengoptimalkan proses pengembangan perangkat lunak, mengurangi pemborosan, dan meningkatkan nilai yang diberikan kepada pelanggan. Prinsip utama identifikasi dan eliminasi pemborosan *(waste)*, pahami nilai dari sudut pandang pelanggan, optimalkan proses alur kerja, dan dorong pembelajaran dan perbaikan terus menerus (*Kaizen*).

5. *Crystal*

Deskripsi singkat *Crystal* adalah keluarga metodologi yang memiliki beberapa varian, masing-masing cocok untuk situasi proyek yang berbeda. Mereka menekankan komunikasi yang baik, kolaborasi, dan pemahaman mendalam tentang bisnis dan teknologi. Fokus utama komunikasi yang efektif, kolaborasi, dan adaptasi metode sesuai dengan kebutuhan proyek.

6. *Dynamic Systems Development Method* (DSDM)

DSDM adalah metodologi yang menekankan pengembangan sistem secara cepat dengan tetap mempertahankan kualitas dan kepatuhan terhadap kebutuhan bisnis. Prinsip utama fokus pada kebutuhan bisnis, iterasi dan inkremental, dan kolaborasi antara pengembang dan pemangku kepentingan. Setiap metodologi *Agile* memiliki pendekatan uniknya sendiri, tetapi semuanya berbagi nilai-nilai dasar yang sama terkait fleksibilitas, kolaborasi tim, dan fokus pada kepuasan pelanggan. Tim pengembangan seringkali memilih atau mengkombinasikan elemen dari metodologi *Agile* yang berbeda sesuai dengan kebutuhan proyek dan karakteristik tim.

2.1.3 Aplikasi

Aplikasi atau *software* aplikasi adalah program komputer yang dirancang untuk melakukan tugas-tugas spesifik pada perangkat elektronik, seperti komputer, *smartphone,* tablet, atau perangkat cerdas lainnya. Aplikasi ini telah menjadi bagian integral dari kehidupan sehari-hari, memfasilitasi berbagai aktivitas dan memenuhi berbagai kebutuhan pengguna. Berikut adalah beberapa jenis aplikasi dan cara penggunaan aplikasi di berbagai kontek. Adapun pembagian jenis-jenis aplikasi sebagai berikut (Afitri Nurva & Budayawan Khairi, 2019).

1. Aplikasi perangkat lunak kantor (*Office Software*) *Microsoft Office*, *Google Workspace* digunakan untuk pekerjaan perkantoran, termasuk pengolah kata, *spreadsheet*, dan presentasi. *LibreOffice, OpenOffice,* alternatif gratis untuk pengolah kata dan aplikasi perkantoran lainnya.
2. Aplikasi produktivitas (*Evernote*) aplikasi catatan digital untuk mencatat ide, pengingat, dan gambar. *Trello, Asana* alat manajemen proyek dan tugas untuk

individu dan tim. *Todoist, Wunderlist* aplikasi manajemen tugas untuk mengatur daftar pekerjaan harian atau proyek.

3. Aplikasi komunikasi *WhatsApp*, *Messenger*, *Telegram*, aplikasi pesan teks dan panggilan suara/video. *Skype, Zoom* aplikasi konferensi video untuk pertemuan bisnis atau pribadi. *Slack Platform* komunikasi tim yang memungkinkan kolaborasi dalam konteks proyek.
4. Aplikasi media sosial *Facebook*, *Instagram*, *Twitter* platform media sosial untuk berbagi foto, video, dan status. *LinkedIn* jaringan sosial bisnis untuk membangun hubungan profesional.
5. Aplikasi navigasi dan peta *Google Maps, Waze* aplikasi navigasi dan peta dengan arahan suara dan pembaruan lalu lintas *real-time*. *Maps.me* aplikasi peta *offline* untuk navigasi tanpa koneksi internet.
6. Aplikasi hiburan *Netflix, Hulu, Disney+* layanan *streaming* video untuk menonton film, serial dan dokumenter. *Spotify*, *Apple Music* aplikasi *streaming* musik untuk mendengarkan lagu-lagu favorit. *Games* (seperti *Angry Birds*, *Candy Crush*) aplikasi permainan untuk hiburan dan rekreasi.
7. Aplikasi perbelanjaan *(E-commerce) Amazon, eBay Platform e-commerce* untuk membeli berbagai produk. *Alibaba, Taobao Platform e-commerce* terkemuka di Tiongkok.

2.1.4 *Cloud Computing*

Awan komputasi adalah model pengelolaan dan penyediaan sumber daya komputasi, seperti *server*, penyimpanan, *database*, jaringan, perangkat lunak, analisis, dan kecerdasan buatan, melalui internet (awan). Dalam model ini,

pengguna dapat mengakses dan menggunakan sumber daya ini tanpa perlu memiliki atau memelihara infrastruktur komputasi fisik sendiri (Senthil et al., 2019).

Keunggulan utama dari teknologi awan komputasi adalah skalabilitas, fleksibilitas, dan efisiensi biaya. Dengan menggunakan layanan awan, perusahaan dapat mengalokasikan sumber daya komputasi sesuai dengan kebutuhan mereka, mengurangi biaya investasi awal dalam perangkat keras dan perangkat lunak, dan mengakses teknologi terbaru tanpa memerlukan pembaruan infrastruktur internal. Awan komputasi memiliki tiga model pelayanan utama (Saraswat et al., 2022).

1. *Infrastructure as a Service* (IaaS), menyediakan akses ke sumber daya infrastruktur, seperti *server* virtual dan penyimpanan awan. Pengguna dapat mengelola dan mengontrol sistem operasi, tetapi bebas dari tugas-tugas pemeliharaan fisik.
2. *Platform as a Service* (PaaS) menyediakan lingkungan pengembangan yang lengkap, termasuk alat pengembangan, penyimpanan data, dan layanan penyelesaian masalah. Pengembang dapat fokus pada pengembangan aplikasi tanpa harus khawatir tentang kompleksitas infrastruktur di bawahnya.
3. *Software as a Service* (SaaS) memberikan aplikasi perangkat lunak yang diakses melalui internet. Pengguna dapat mengakses aplikasi ini dari perangkat apa pun yang terhubung ke internet, tanpa perlu menginstal atau memperbarui perangkat lunak secara lokal.

Selain itu, awan komputasi memungkinkan konsep seperti *Big* data, kecerdasan buatan, dan *Internet of Things* (IoT) untuk berkembang dengan cepat. Meskipun ada kekhawatiran tentang keamanan dan privasi data, penyedia layanan awan

terkemuka terus meningkatkan praktik keamanan mereka untuk melindungi data pengguna. Secara keseluruhan, awan komputasi telah mengubah cara perusahaan menyusun, menyimpan, dan mengakses data serta mengembangkan aplikasi, membawa efisiensi dan inovasi yang signifikan ke dunia teknologi informasi modern.

2.1.5 Pengujian Aplikasi

Pengujian perangkat lunak melibatkan beberapa pendekatan, di antaranya *White Box, Black Box,* dan *Grey Box Testing*. *White Box Testing* melibatkan pemeriksaan internal kode sumber perangkat lunak. Pengujian ini memeriksa struktur dan logika kode untuk menemukan *bug* dan memastikan setiap bagian aplikasi berfungsi sesuai yang diharapkan. Berikut penjelasan tentang ketiga metode pengujian aplikasi tersebut menurut (Ilfa et al., 2023).

1. *Black Box Testing*, di sisi lain memeriksa fungsionalitas aplikasi tanpa memperhatikan struktur internalnya. Pengujian ini melibatkan penggunaan *input* dan mengamati *output* untuk memeriksa apakah aplikasi berperilaku sebagaimana mestinya dan memenuhi kebutuhan pengguna (Ibrahim, 2020).
2. *White Box Testing,* metode pengujian perangkat lunak di mana pengujian dilakukan dengan memahami struktur internal dan logika aplikasi. Pengujian ini melibatkan pemeriksaan kode sumber, aliran kontrol, dan jalur data untuk memastikan bahwa semua kondisi dan skenario telah diuji secara menyeluruh. Tujuannya adalah mengidentifikasi kesalahan logika, kekurangan dalam kode, dan memastikan bahwa aplikasi berfungsi sesuai dengan spesifikasi desainnya. Dalam *white box testing*, pengujian dilakukan

dari sudut pandang pengembang untuk memastikan keandalan, keamanan, dan kinerja aplikasi sebelum dilepas ke pengguna akhir.

3. *Grey Box Testing* menggabungkan elemen dari kedua pendekatan sebelumnya. Pengujian ini memberikan pengetahuan sebagian tentang internal aplikasi, memungkinkan pengujian yang lebih terfokus pada titik-titik penting tanpa harus memeriksa semua kode (Ibrahim, 2020).

Ketiga metode ini penting karena mereka memberikan pandangan yang berbeda terhadap kualitas perangkat lunak, memastikan bahwa aplikasi tidak hanya berfungsi dengan benar dari segi penggunaan, tetapi juga kokoh secara internal, dan dapat diandalkan dalam berbagai kondisi.

2.2 Teori Khusus

2.2.1 Android

Android merupakan sistem operasi yang berbasis linux yang dikembangkan oleh google untuk perangkat mobile seperti smartphone dan tablet. Tentunya dalam sistem android mencangkup sistem operasi, middleware dan aplikasi. Android sendiri menyediakan platform yang bersifat terbuka sehingga pengembang dapat mengembangkan aplikasi yang akan dibuatnya (Jingjing et al., 2020).

2.2.2 Android Studio

Android studio adalah lingkungan pengembangan terintegrasi *Integrated Development Environment* (IDE) yang disediakan oleh google untuk membangun aplikasi android. *Integrated Development Environment* (IDE) ini dirancang khusus untuk pengembangan aplikasi android, menyediakan berbagai alat dan fitur yang mempermudah proses pengembangan. Dengan android studio, pengembang dapat

membuat aplikasi android dengan antarmuka pengguna yang menarik dan fungsionalitas yang kaya (Zalukhu & Pangaribuan, 2023).

Salah satu keunggulan android studio adalah *emulator* android bawaannya, yang memungkinkan pengembang menguji aplikasi mereka pada berbagai perangkat android *virtual* tanpa perlu perangkat fisik. IDE ini juga menyediakan editor kode cerdas dengan penyelesaian otomatis, memungkinkan pengembang menulis kode dengan cepat dan efisien. Selain itu, android studio dilengkapi dengan alat pengujian yang kuat, memungkinkan pengembang melakukan pengujian unit, pengujian integrasi, dan pengujian uji coba pada aplikasi mereka (Senthil et al., 2019).

Android Studio juga terintegrasi dengan layanan google seperti *firebase*, memungkinkan pengembang mengakses berbagai alat pengembangan dan analisis yang disediakan oleh google. Dengan dukungan penuh untuk bahasa pemrograman *kotlin*, android studio memberikan fleksibilitas kepada pengembang untuk memilih bahasa pemrograman yang paling sesuai dengan proyek mereka. Dengan rangkaian fitur yang lengkap dan dukungan terus-menerus dari komunitas pengembang, android studio menjadi pilihan utama bagi para pengembang yang ingin membuat aplikasi android yang inovatif dan berkualitas.

2.2.3 *Java*

Java adalah bahasa pemrograman yang sangat populer dan serbaguna, termasuk untuk pengembangan aplikasi *mobile*. Dalam konteks pengembangan *mobile, java* banyak digunakan untuk membuat aplikasi android. Berikut adalah 150 kata yang menjelaskan *java* dalam konteks pengembangan aplikasi *mobile*.

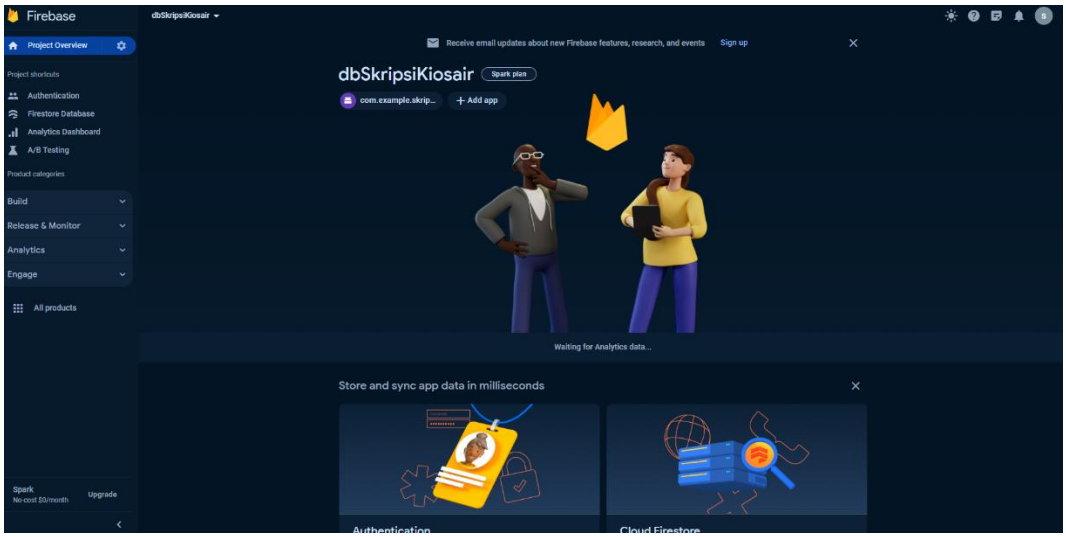
Java adalah bahasa pemrograman objek yang kuat dan umum digunakan untuk mengembangkan aplikasi *mobile*, terutama aplikasi android. Keunggulan *java* terletak pada portabilitasnya yang tinggi, artinya kode yang ditulis dalam *java* dapat dijalankan di berbagai *platform* dengan sedikit atau tanpa perubahan. *Java mobile development* memungkinkan pengembang untuk membuat aplikasi android dengan antarmuka pengguna yang menarik, fungsionalitas yang kompleks, dan kinerja yang optimal.

Menurut (Kurniawan Farhurrahman et al., 2023) pengembang *java* dapat memanfaatkan fitur-fitur seperti *garbage collection, multithreading*, dan kontrol alur program yang memudahkan dalam pengembangan aplikasi *mobile* yang kompleks. Selain itu, *java* menyediakan kerangka kerja pengembangan yang kaya dan berbagai pustaka yang mendukung berbagai fungsi, mulai dari pengelolaan tata letak hingga interaksi dengan *database*. Pengembang *mobile* yang menggunakan *java* dapat mengakses *Application Programming Interface* (API) Android yang luas, memungkinkan integrasi dengan perangkat keras dan perangkat lunak android dengan mudah. Dengan dukungan komunitas yang besar dan berbagai alat pengembangan yang tersedia, *java* tetap menjadi pilihan populer bagi pengembang yang ingin membuat aplikasi *mobile* yang andal dan inovatif.

2.2.4 *Firebase*

Firebase Cloud Computing adalah *platform* pengembangan aplikasi seluler dan web yang disediakan oleh google. Ini memungkinkan pengembang untuk membuat aplikasi berkualitas tinggi dengan cepat, dengan menyediakan berbagai alat dan layanan yang dapat diintegrasikan dengan mudah ke dalam proyek pengembangan. *Firebase* menyediakan solusi *cloud computing* lengkap, termasuk

penyimpanan data, pengautentikasian pengguna, pengelolaan basis data, serta analisis dan pelacakan kinerja aplikasi.



Gambar 2. 1 *Database Firebase*
Sumber : (Data Penelitian 2023)

Menurut (Khawas & Shah, 2018) salah satu fitur unggulan *Firebase* adalah *Firebase Realtime Database*, yang memungkinkan penyimpanan data dalam bentuk *Java Script Object Notation* (JSON) dan sinkronisasi data secara *real-time* antara pengguna aplikasi. Ini memungkinkan pengembang membangun aplikasi kolaboratif dan dinamis dengan mudah.

Firebase juga menawarkan *firebase authentication*, yang memungkinkan pengembang menerapkan sistem pengautentikasian pengguna dengan cepat dan aman menggunakan *email*, media sosial, atau penyedia identitas lainnya.

Selain itu, *firebase* menyediakan layanan *cloud messaging (Firebase Cloud Messaging*) untuk mengirimkan pemberitahuan *push* ke pengguna aplikasi, serta layanan analisis aplikasi *(Firebase Analytics)* untuk memahami perilaku pengguna dan meningkatkan pengalaman pengguna.

Dengan *Firebase Cloud Computing*, pengembang dapat menghemat waktu dan upaya dalam mengelola infrastruktur *backend*, sehingga dapat fokus pada pengembangan fitur dan fungsionalitas aplikasi yang inovatif. *Firebase* mendukung pertumbuhan aplikasi dengan skalabilitas yang mudah disesuaikan sesuai kebutuhan, menjadikannya pilihan yang populer dalam dunia pengembangan aplikasi seluler dan web.

Arsitektur firebase secara umum dapat dijabarkan sebagai berikut:

1. Firabase SDK *(Software Development Kit)*

   Firebase SDK adalah firebase yang memungkinkan pengembang untuk mengintegrasikan berbagai layanan firebase. SDK tersedia untuk berbagai platform termasuk android, IOS, web dan platform lainnya.

2. Firebase Console

   Firebase yang menyediakan antarmuka web yang memungkinkan pengembang untuk mengelola firebase mereka.

3. *Backend Services*

   Firebase yang digunakan oleh pengembang dalam mengolah data,otentikasi,hosting dan analitik.

2.2.5 Metode *Black Box*

Pengujian aplikasi, terutama melalui metode *blackbox testing* adalah langkah kritis dalam pengembangan perangkat lunak modern. Dalam *blackbox testing* aplikasi diuji tanpa mempertimbangkan struktur internal atau logika kode. Sebagai gantinya, fokus ditempatkan pada memverifikasi apakah aplikasi berperilaku sesuai dengan spesifikasi yang telah ditentukan.

Metode ini memungkinkan pengujian menyeluruh dari perspektif pengguna akhir, memastikan bahwa aplikasi berfungsi dengan baik, mudah digunakan, dan bebas dari *bug* atau kesalahan yang dapat mempengaruhi pengalaman pengguna. Selain itu, *blackbox testing* memungkinkan pengujian pada berbagai *platform*, sistem operasi, dan perangkat keras tanpa memerlukan pengetahuan mendalam tentang implementasi internal aplikasi.

Dalam pengujian *black box*, berbagai teknik seperti uji fungsional, uji kesalahan, dan uji keamanan dapat diterapkan. Tim pengujian menggunakan skenario pengguna nyata untuk memastikan bahwa aplikasi dapat menanggapi *input* dengan benar dan memberikan *output* yang diharapkan. Hasil dari pengujian *black box* memberikan gambaran menyeluruh tentang keandalan dan kualitas aplikasi, memungkinkan pengembang untuk memperbaiki masalah sebelum aplikasi dirilis ke pasar.

Pentingnya *black box testing* tidak bisa diabaikan dalam siklus pengembangan perangkat lunak. Dengan pendekatan ini, pengembang dapat memastikan bahwa aplikasi yang mereka kembangkan memenuhi standar kualitas tinggi dan memberikan pengalaman pengguna yang optimal.

2.2.6 Metode *Extreme Programming*

Extreme Programming (XP) adalah metodologi pengembangan perangkat lunak yang menekankan pada kolaborasi tim, komunikasi terbuka, dan respon cepat terhadap perubahan kebutuhan pelanggan. Dalam XP, pengembang bekerja dalam pasangan untuk menulis kode, melakukan pengujian otomatis, dan berkolaborasi secara terus-menerus. Prinsip-prinsipnya melibatkan pengembangan perangkat

lunak dalam iterasi pendek, pengujian terus-menerus, desain sederhana, dan fleksibilitas terhadap perubahan kebutuhan pelanggan (Carolina & Rusman, 2019).

Menurut (Supriyatna, 2018) salah satu aspek kunci dari XP adalah "*Pair Programming*," di mana dua pengembang bekerja bersama pada satu komputer. Selain itu, "*Continuous Integration*" memastikan bahwa perubahan kode diintegrasikan ke repositori umum secara terus-menerus, memungkinkan deteksi cepat terhadap konflik atau kesalahan. XP juga mendorong untuk merilis versi perangkat lunak yang kecil secara berkala, memberi pelanggan akses langsung ke fitur terbaru (Fadiyah Ghina et al., 203 C.E.).

Metodologi ini mengutamakan umpan balik pelanggan yang cepat dan mengadaptasi perangkat lunak sesuai dengan kebutuhan mereka. XP juga mendorong tim untuk menjaga desain perangkat lunak tetap sederhana dan mudah dimengerti oleh semua anggota tim, meminimalkan kompleksitas yang tidak perlu. Dengan pendekatan yang berfokus pada kerjasama tim, XP membantu mengurangi risiko dan meningkatkan kualitas perangkat lunak yang dihasilkan.

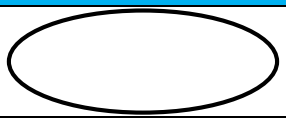
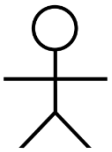
2.3 UML *(Unified Modeling Language)*

UML (*Unified Modeling Language*) merupakan sebuah bahasa standar yang digunakan untuk memodelkan, merancang, serta mendokumentasikan sistem perangkat lunak dengan menggunakan pengembangan dengan berorientasikan pada objek. Dengan adanya UML perancang dapat melihat gambaran dari sebuah sistem, struktur serta komunikasi antar komponen. Dengan adanya UML dapat memudahkan pemahaman dan pengembangan suatu sistem secara kompleks dikarenakan diagram yang dipakai di dalam UML adalah gambaran secara ringkas

sehingga dapat dipahami dan dipelajari (Mayangsari & Badrul, 2023). Bagian-bagian *Unified Modeling Language* adalah sebagai berikut :

1. *Use case Diagram* di dalam diagram *use case* ada dua unsur utama yaitu *actor* dan *use case*. Aktor adalah orang yang berinteraksi dengan sistem informasi. Sedangkan *use case* adalah sebuah fungsi yang dapat digunakan oleh aktor tersebut (Hendini, 2016).

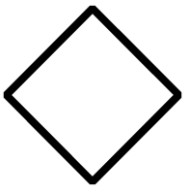
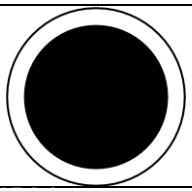
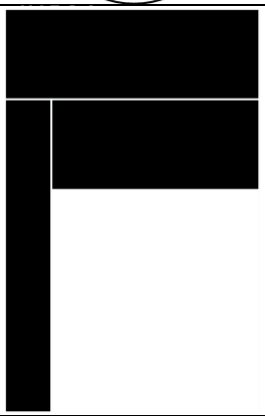
Tabel 2. 1 Simbol-simbol *Use case Diagram* Gambar

Gambar	Keterangan
	Berfungsi dalam memenuhi kebutuhan dari pengguna dan kegian yang dilakukan oleh sistem tersebut
	Aktor yang mampu melakukan berbagai perintah dalam sebuah sistem, seperti mengambil data, memperbarui data, menghapus data, atau mengirimkan data
	Asosiasi menggambarkan bagaimana aktor menggunakan sistem untuk mencapai tujuannya
	Asosiasi menggambarkan bagaimana aktor menggunakan sistem untuk mencapai tujuannya
	*Symbol* yang bertujuan untuk menghindari perulangan informasi sehingga *use case* dapat dengan mudah dibaca
	Memungkinkan untuk menggabungkan kelas dan menambahkan atribut dan metode yang baru.

Sumber : (Hendini, 2016)

2. Diagram *activity* (*Activity Diagram)* menggambarkan urutan kegiatan. Diagram ini menggunakan simbol-simbol seperti *elips*, kotak, dan panah. Persegi panjang melambangkan aktivitas, oval melambangkan keadaan, dan panah melambangkan arus (Suhendri, 2018).

Tabel 2. 2 Simbol *Activity Diagram*

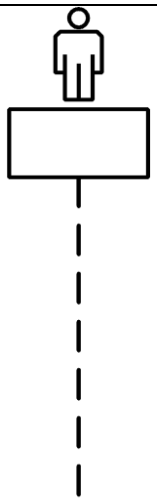
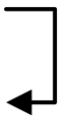
Simbol	Keterangan
	Status awal, Sebuah diagram aktivitas memiliki sebuah status awal
	Aktifitas, yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
	Percabangan dimana ada pilihan aktivitas yang lebih dari satu
	Status Akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
	*Swimlane*, memisahkan organisasi bisnis yang bertanggung jawab terhadap aktifitas yang terjadi

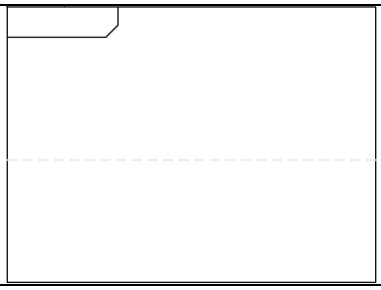
Sumber : (Hendini, 2016)

3. Diagram Urutan *(Sequence Diagram)*. Diagram ini mewakili fase-fase proses yang terjadi antar objek dalam suatu sistem, seperti interaksi yang dilakukan, respon yang diterima, dan informasi yang dikirim. Diagram ini juga membantu menjelaskan jalur komunikasi dalam sistem (Hendini, 2016).

Simbol-simbol yang digunakan dalam *Sequence Diagram* yaitu:

Tabel 2. 3 Simbol – simbol *Sequence Diagram*

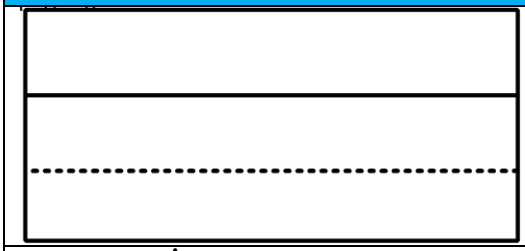
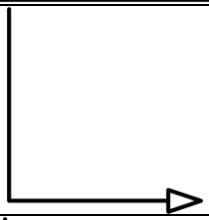
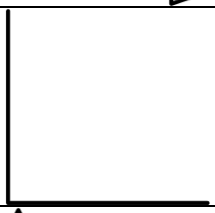
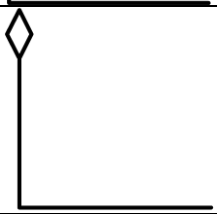
Simbol	Keterangan
	*Lifeline,* Berikutnya adalah *lifeline*. Komponen ini digambarkan dengan bentuk garis putus-putus. *Lifeline* ini biasanya memiliki kotak yang berisi objek yang memiliki fungsi untuk menggambarkan aktivitas dari objek.
	*Actor Activation,* Representasi dari entitas yang berpartisipasi dalam skenario atau proses. Objek diidentifikasi dengan nama kelas atau instansiasi objek tertentu
	*Message*, Representasi dari panggilan metode atau pertukaran pesan antara objek-objek. Ada dua jenis pesan utama: pesan pengirim (pesan yang dikirim oleh objek pengirim) dan pesan penerima (pesan yang diterima oleh objek penerima)
	*Self Message*, mencerminkan proses atau metode baru yang dijalankan dalam operasi jalur panggilan. Ini adalah spesifikasi Pesan, biasanya dalam diagram Urutan. Panggilan Pesan Mandiri menunjukkan pemanggilan yang disarankan; tingkat aktivasi baru ditambahkan dengan setiap Panggilan.

	*Alternative Combined Fragment*, digunakan dalam diagram urutan untuk menentukan bagian dari sekelompok garis hidup/aktor, yang mewakili aliran bersyarat. Ini juga memodelkan logika *if-then-else* dalam diagram urutan.
	Create Message, mendefinisikan komunikasi tertentu antara Jalur Kehidupan suatu Interaksi. Buat pesan adalah jenis pesan yang mewakili contoh garis hidup (target).

Sumber : (Hendini, 2016)

4. *Class Diagram*, membuat ilustrasi struktur sistem dengan *class diagram* yang berisikan definisi dari tiap-tiap kelas yang akan dibuat. Sebuah *class diagram* berisi sebuah deskripsi dari sebuah kelas yang memiliki hubungan dengan kelas lainnya, berdasarkan penelitian yang dilakukan oleh (Hendini, 2016) *class diagram* adalah entitas yang diwakili oleh suatu kotak yang menggambarkan sebuah objek. Atribut adalah deskripsi dari entitas yang ditampilkan sebagai teks yang berada dalam kotak kelas. Operasi adalah aksi yang dapat dilakukan oleh entitas yang diwakili oleh kotak kelas (Mayangsari & Badrul, 2023).

Tabel 2. 4 Simbol-simbol *Class Diagram* Gambar Keterangan Kelas

Gambar	Keterangan
	Representasi dari suatu entitas atau objek dalam sistem, yang mencakup atribut (variabel) dan metode (fungsi) yang dimilikinya.
	Pewarisan adalah Hubungan di mana suatu kelas dapat mewarisi atribut dan metode dari kelas lain.
	Asosiasi adalah Hubungan antara dua kelas yang menunjukkan keterkaitan atau ketergantungan di antara mereka
	*Agregasi* adalah Hubungan di mana suatu kelas merupakan bagian dari kelas lain, tetapi dapat eksis secara independen.

Sumber : (Bun, 2021)

2.4 Penelitian Terdahulu

Penelitian ini didasari dari hasil penelitian terdahulu, baik dari jenis penelitian maupun dari teori yang digunakan pada penelitian sebelumnya :

1. (Khawas & Shah, 2018) pada penelitiannya yang berjudul ***"Application of Firebase in Android App Development-A Study"*** mengungkapkan sistem manajemen basis data relasional (RDBMS) berguna untuk menangani data yang tidak terstruktur. Maka dari itu *firebase* sebagai solusi untuk menangani hal tersebut, menggunakan layanan *cloud computing* khususnya *firebase* sangat dapat diandalkan.

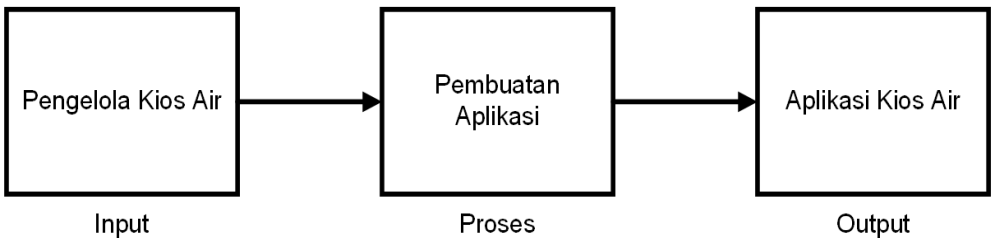
2. (Saraswat et al., 2022), pada penelitiannya yang berjudul ***"Anti-spoofing-enabled Contactless Attendance Monitoring System in the COVID-19 Pandemic"*** membahas tentang penandaan kehadiran fisik dengan *virtual* dimana permasalahan ini bermula munculnya pandemi *covid* 19 dengan berbagai keterbatasan untuk bertemu secara fisik. Dengan memanfaatkan aplikasi absensi berbasis kamera nirkontak dengan dilengkapi fungsi *anti-spoofing* dapat mendeteksi keaktifan hingga absensi palsu. Aplikasi ini dibangun menggunakan *database firebase* dengan tingkat akurasi 95,85%.
3. (Supriyatna, 2018), pada penelitiannya yang berjudul **"Metode *Extreme Programming* Pada Pembangunan Web Aplikasi Seleksi Peserta Pelatihan Kerja"** penelitian ini bertujuan menciptakan aplikasi web menggunakan metode *Extreme Programming* (XP) untuk mempermudah pendaftaran dan ujian seleksi peserta pelatihan kerja di balai latihan kerja. Saat ini, proses penerimaan peserta masih konvensional, memaksa calon peserta datang ke lokasi, menciptakan hambatan bagi masyarakat. Aplikasi ini dirancang untuk mengatasi kendala tersebut dengan memberikan informasi tentang periode penerimaan, memungkinkan pendaftaran *online*, dan pelaksanaan ujian seleksi. *Extreme Programming* (XP) dipilih karena fleksibilitasnya dalam menghadapi perubahan *requirement* yang cepat. Hasilnya adalah aplikasi web yang memfasilitasi penyebaran informasi dan proses seleksi peserta pelatihan kerja, meminimalkan hambatan dan memberikan kemudahan kepada calon peserta.
4. (Ilfa et al., 2023), pada penelitiannya yang berjudul" **Metode *Agile Scrum* Dalam Pembuatan Aplikasi Permohonan Informasi E-PPID Bawaslu"**

melakukan pengembagan aplikasi di Bawaslu Sleman, sebagai lembaga pengawas pemilihan umum, memiliki *website* resmi, tetapi tata letak informasinya kurang terstruktur. Dalam mengakomodasi pengguna aplikasi *mobile* yang semakin bertambah, Bawaslu Sleman memutuskan untuk mengembangkan aplikasi android. Aplikasi ini bertujuan menyajikan informasi dengan lebih teratur dan efisien kepada masyarakat. Pengembangan aplikasi menggunakan metode *Agile Scrum* dengan menggunakan bahasa pemrograman *Kotlin* untuk android dan *server database* SQL untuk menyimpan data. Proses pembuatan aplikasi ini didasarkan pada wawancara dengan pihak lembaga, memastikan akses informasi dan kebutuhan yang diperoleh *valid*. Dengan aplikasi ini, diharapkan masyarakat dapat dengan mudah mengakses informasi terstruktur tentang pemilihan umum melalui perangkat *mobile* mereka.

5. (Kurniawan Fathurrahman et al., 2023), pada penelitiannya yang berjudul **“ Rancang Bangun Aplikasi *Cloud Storage* Dengan *Angular* Dan *Firebase* Berbasis Android”** penelitian ini menunjukkan bahwa teknologi *Cloud Storage* sangat berguna sebagai medium penyimpanan data, aplikasi android dapat dibuat dengan *Angular* menggunakan *Cordova* untuk android, dan *Firebase* menyediakan berbagai fitur dan layanan, termasuk *Firebase storage,* dalam pengembangan aplikasi android.

2.5 Kerangka Pemikiran

Pada kerangka pemikiran penulis melakukan analisa pengelola kios air, kemudian pembuatan aplikasi, dan hasilnya aplikasi dan sistem pengelolaan kios air dengan android.



Gambar 2. 2 Kerangka Pemikiran
Sumber : (Data Penelitian 2023)

Penjelasan mengenai kerangka pemikiran:

1. Input : Pengelola kios air

   Studi kasus yang ditemukan diusaha UMKM Kios air ketika pengelola ingin melihat persentase keuntungan bulanan pengelola harus membuka buku catatan pembayaran dan menjumlahkan nya terlebih dahulu tentunya ini tidak efektif dikarenakan akan memakan waktu dan kesulitan lain yang dirasakan oleh pengelolal kios air adalah sulitnya mendata pelanggan kios air.

2. Proces : pembuatan aplikasi

   Dengan ditemukannya beberapa kendala atau permasalahan pada usaha UMKM kios air maka peneliti membuat sebuah aplikasi kios air sederhan untuk mengatasi permasalahan tersebut.

3. Output : Implementasi aplikasi kios air

   Dengan dibuatnya aplikasi ini peneliti tentunya berharap adanya perubahan dari pendataan dan pencatatan yang konvensional ke

digitalisasi, sehingga kemungkinan terjadinya kesalahan dalam pendataan dapat di minimalisir.