# BAB II
# KAJIAN LITERATUR

## 2.1 Aplikasi *Mobile*

### 2.1.1 Pengertian Aplikasi *Mobile*

Aplikasi *mobile* merupakan perangkat lunak yang dirancang untuk berjalan pada perangkat bergerak seperti *smartphone* dan *tablet*. Aplikasi ini dikembangkan untuk memenuhi kebutuhan pengguna yang bersifat dinamis, fleksibel, serta dapat diakses kapan saja dan di mana saja melalui perangkat genggam. Berbeda dengan aplikasi *desktop* yang dijalankan pada komputer personal, aplikasi *mobile* dirancang dengan mempertimbangkan keterbatasan sumber daya perangkat seperti ukuran layar, kapasitas memori, dan daya baterai.

Aplikasi *mobile* merupakan sistem perangkat lunak berbasis sistem operasi *mobile* seperti Android dan iOS yang berfungsi untuk mempermudah aktivitas pengguna dalam berbagai bidang, termasuk pendidikan, bisnis, dan layanan informasi [12]. Perkembangan teknologi *mobile* yang pesat mendorong meningkatnya kebutuhan akan aplikasi yang responsif, interaktif, dan mudah digunakan.

Selain itu, aplikasi *mobile* juga memanfaatkan teknologi jaringan internet untuk menyediakan fitur sinkronisasi data, penyimpanan berbasis *cloud*, serta integrasi dengan layanan pihak ketiga. Hal ini memungkinkan aplikasi memberikan pengalaman pengguna (*user experience*) yang lebih optimal dibandingkan sistem konvensional berbasis *web desktop* [13].

Dengan demikian, aplikasi *mobile* dapat didefinisikan sebagai perangkat lunak yang dirancang khusus untuk perangkat bergerak dengan mempertimbangkan efisiensi, mobilitas, dan interaktivitas sebagai karakteristik utamanya.

### 2.1.2 Karakteristik Aplikasi *Mobile*

Aplikasi *mobile* memiliki beberapa karakteristik utama yang membedakannya dari aplikasi berbasis *desktop* maupun *web*, yaitu:

1. Portabilitas

   Aplikasi dapat digunakan di berbagai lokasi selama perangkat aktif dan memiliki sumber daya yang cukup [14].

2. Antarmuka Responsif

Tampilan dirancang menyesuaikan ukuran layar perangkat, baik *smartphone* maupun *tablet* [15].

3. Interaktivitas Tinggi

   Mendukung interaksi berbasis sentuhan (*touchscreen*), *gesture*, notifikasi, dan integrasi sensor perangkat [14].

4. Konektivitas Jaringan

   Sebagian besar aplikasi *mobile* terhubung dengan internet untuk mengambil dan menyimpan data secara *real-time* [15].

5. Efisiensi Sumber Daya

   Pengembangan aplikasi *mobile* harus mempertimbangkan penggunaan memori, CPU, dan baterai secara optimal [14].

Penelitian oleh Azila et al. menyatakan bahwa keberhasilan aplikasi *mobile* sangat dipengaruhi oleh kemudahan penggunaan (*usability*), kecepatan akses, dan kestabilan sistem [16]. Oleh karena itu, dalam perancangan aplikasi resep masakan berbasis *mobile*, aspek antarmuka dan performa menjadi faktor penting yang harus diperhatikan.

### 2.1.3 Peran Aplikasi *Mobile* dalam Sistem Informasi Kuliner

Aplikasi mobile berperan sebagai media digital yang memudahkan pengguna dalam mengakses informasi kuliner, seperti resep, bahan, langkah memasak, dan kategori makanan secara cepat dan fleksibel melalui perangkat bergerak [17].

Dibandingkan media konvensional, aplikasi mobile mampu meningkatkan kemudahan akses informasi, mempercepat penyampaian informasi, serta memberikan pengalaman pengguna yang lebih interaktif [12][16].

Perkembangan aplikasi kuliner juga telah mendorong lahirnya berbagai penelitian mengenai digitalisasi resep masakan. Penelitian terdahulu menunjukkan bahwa aplikasi resep umumnya menyediakan fitur pencarian resep, pengelompokan kategori, dan penyimpanan resep untuk membantu pengguna memperoleh informasi memasak dengan lebih mudah [17]. Namun, penelitian lain menunjukkan bahwa sebagian besar aplikasi masih berfokus pada penyajian konten resep, sedangkan integrasi fitur yang mendukung aktivitas memasak secara menyeluruh, seperti pengelolaan resep, *meal planning*, dan daftar belanja, masih terbatas [6][18]. Selain itu, aspek *usability* dan *user engagement* juga masih menjadi tantangan dalam pengembangan aplikasi kuliner berbasis mobile [6]. Oleh karena itu, masih diperlukan pengembangan sistem informasi kuliner yang mampu mengintegrasikan berbagai

kebutuhan tersebut dalam satu aplikasi sehingga dapat memberikan pengalaman pengguna yang lebih efektif dan efisien.

## 2.2 Resep Masakan Digital

### 2.2.1 Konsep Resep Masakan Digital

Resep masakan digital merupakan transformasi resep konvensional (buku cetak atau catatan manual) ke dalam bentuk sistem informasi berbasis digital yang dapat diakses melalui perangkat elektronik seperti *smartphone*, *tablet*, maupun komputer. Digitalisasi ini tidak hanya mengubah format penyimpanan, tetapi juga menambahkan fitur interaktif seperti pencarian otomatis, filter kategori, penyimpanan favorit, hingga rekomendasi resep [17].

Menurut penelitian dalam bidang sistem informasi, digitalisasi konten informasi memungkinkan peningkatan efisiensi akses, kemudahan distribusi, serta personalisasi pengalaman pengguna [19]. Dalam konteks kuliner, resep digital memberikan kemudahan bagi pengguna untuk:

1. Mencari resep berdasarkan bahan tertentu
2. Mengakses langkah memasak secara sistematis
3. Menyimpan dan membagikan resep

Penelitian oleh Nurlaelatul (2023) dalam pengembangan aplikasi berbasis *mobile* menunjukkan bahwa digitalisasi informasi berbasis aplikasi mampu meningkatkan efektivitas penyampaian informasi serta mempermudah pengguna dalam memperoleh data yang dibutuhkan secara *real-time* [20]. Hal ini sejalan dengan kebutuhan aplikasi resep masakan yang dirancang untuk menyediakan informasi secara cepat dan terstruktur.

### 2.2.2 Komponen Utama Resep Masakan Digital

Dalam implementasinya, resep masakan digital umumnya memiliki beberapa komponen utama, yaitu [21]:

1. Judul Resep

   Menunjukkan nama masakan secara spesifik.
2. Daftar Bahan

   Berisi bahan-bahan yang diperlukan lengkap dengan takaran.
3. Langkah-Langkah Pembuatan

   Disusun secara sistematis agar mudah dipahami.
4. Kategori Masakan

   Misalnya makanan utama, makanan penutup, minuman, dan lain-lain.

5. Gambar atau Media Visual

   Mendukung pemahaman pengguna terhadap hasil akhir masakan.

6. Fitur Pencarian dan Filter

   Memungkinkan pengguna menemukan resep dengan cepat.

Menurut studi mengenai sistem informasi berbasis *mobile*, struktur data yang terorganisir dengan baik akan meningkatkan *usability* dan pengalaman pengguna (*user experience*) [14]. Oleh karena itu, dalam penelitian ini, struktur resep dirancang menggunakan model data terstruktur agar memudahkan pengolahan dan pencarian data.

### 2.2.3 Keunggulan Resep Masakan Digital Dibanding Konvensional

Perkembangan teknologi informasi telah mendorong transformasi dalam cara masyarakat memperoleh dan mengelola informasi, termasuk dalam bidang kuliner. Pada masa sebelumnya, resep masakan umumnya disajikan dalam bentuk media fisik seperti buku resep, majalah, atau catatan pribadi. Namun, dengan berkembangnya teknologi digital dan perangkat *mobile*, informasi resep kini dapat diakses melalui aplikasi maupun sistem informasi berbasis *web*. Transformasi ini memberikan berbagai keuntungan dari segi efisiensi pencarian informasi, fleksibilitas akses, serta kemudahan pengelolaan data.

Sistem informasi resep masakan berbasis digital memungkinkan pengguna memperoleh rekomendasi resep berdasarkan bahan yang dimiliki, preferensi makanan, atau kategori tertentu secara lebih cepat dan akurat. Integrasi teknologi seperti API dan basis data juga memungkinkan aplikasi menyediakan informasi resep secara dinamis dan interaktif sehingga meningkatkan pengalaman pengguna dalam mencari inspirasi masakan. Pengembangan sistem informasi resep dengan integrasi layanan digital terbukti mampu menyediakan proses pencarian resep yang lebih efisien dan praktis dibandingkan metode manual yang mengandalkan media cetak atau catatan pribadi [17].

Perbandingan antara resep konvensional dan resep digital dapat dianalisis melalui beberapa aspek utama seperti aksesibilitas, proses pencarian informasi, penyimpanan data, pembaruan informasi, serta interaksi pengguna dengan sistem. Perbandingan tersebut dapat dilihat pada Tabel 2.1 berikut.

Tabel 2.1 Perbandingan Resep Konvensional dan Resep Digital

| Aspek | Resep Konvensional | Resep Digital |
|---|---|---|
| Aksesibilitas | Terbatas pada media fisik seperti buku atau catatan | Dapat diakses kapan saja melalui perangkat *mobile* |

| Aspek | Resep Konvensional | Resep Digital |
|---|---|---|
| Pencarian | Dilakukan secara manual dengan membuka halaman atau indeks | Menggunakan fitur pencarian otomatis (*search engine*) |
| Penyimpanan | Disimpan dalam bentuk buku atau catatan | Disimpan dalam basis data digital |
| Pembaruan | Sulit diperbarui karena harus mencetak ulang atau menulis ulang | Mudah diperbarui melalui sistem |
| Interaksi | Bersifat statis dan tidak interaktif | Mendukung fitur interaktif seperti favorit, komentar, atau rekomendasi |

Berdasarkan aspek aksesibilitas, resep konvensional memiliki keterbatasan karena pengguna harus memiliki akses fisik terhadap media tersebut. Hal ini membuat pengguna hanya dapat mengakses informasi resep pada tempat dan waktu tertentu. Sebaliknya, resep digital yang diimplementasikan dalam aplikasi *mobile* memungkinkan pengguna mengakses informasi secara fleksibel melalui perangkat *smartphone* atau *tablet*. Aplikasi *mobile* mampu menyediakan informasi secara cepat dan mudah diakses karena sistem telah terintegrasi dengan *database* dan jaringan internet sehingga pengguna dapat memperoleh data secara *real-time* [22].

Dari aspek pencarian informasi, metode konvensional mengharuskan pengguna mencari resep secara manual dengan membaca halaman demi halaman atau menggunakan indeks pada buku resep. Proses ini cenderung memakan waktu dan kurang efisien, terutama ketika pengguna ingin menemukan resep berdasarkan bahan tertentu. Sebaliknya, sistem resep digital memanfaatkan teknologi pencarian berbasis kata kunci atau bahan makanan yang memungkinkan pengguna menemukan resep secara otomatis dan lebih cepat. Dalam penelitian mengenai sistem pencarian resep berbasis API, aplikasi mampu memberikan rekomendasi resep berdasarkan bahan yang dimasukkan oleh pengguna sehingga mempermudah proses pencarian menu masakan yang sesuai dengan kebutuhan pengguna [17].

Pada aspek penyimpanan data, resep konvensional biasanya disimpan dalam bentuk dokumen fisik seperti buku atau catatan. Metode penyimpanan ini memiliki beberapa kelemahan, antara lain risiko kerusakan, kehilangan, serta keterbatasan dalam pengelolaan data dalam jumlah besar. Sebaliknya, resep digital disimpan dalam sistem basis data yang memungkinkan pengelolaan data secara terstruktur dan sistematis. Dengan menggunakan

*database*, aplikasi dapat menyimpan ribuan resep sekaligus serta memungkinkan proses pencarian dan pengambilan data dilakukan dengan cepat dan efisien.

Selanjutnya pada aspek pembaruan informasi, sistem konvensional memiliki keterbatasan karena setiap perubahan atau penambahan resep memerlukan proses pencetakan ulang atau penulisan ulang. Proses tersebut memerlukan waktu dan biaya tambahan. Sebaliknya, pada sistem digital, pembaruan data dapat dilakukan secara langsung melalui sistem *backend* sehingga informasi yang tersedia pada aplikasi selalu diperbarui secara otomatis. Hal ini menjadikan sistem digital lebih fleksibel dan adaptif terhadap perubahan informasi atau kebutuhan pengguna.

Aspek terakhir adalah interaksi pengguna dengan sistem. Resep konvensional umumnya bersifat statis karena hanya menyajikan informasi dalam bentuk teks atau gambar tanpa adanya interaksi langsung dengan pengguna. Sebaliknya, aplikasi resep digital memungkinkan integrasi berbagai fitur interaktif seperti penyimpanan resep favorit, rekomendasi resep berdasarkan bahan, hingga personalisasi pengalaman pengguna. Selain itu, dalam pengembangan aplikasi *mobile* modern, aspek kegunaan (*usability*) menjadi faktor penting dalam memastikan aplikasi dapat digunakan secara efektif, efisien, dan memberikan tingkat kepuasan yang baik bagi pengguna. Oleh karena itu, pengujian *usability* sering digunakan untuk mengevaluasi kemudahan penggunaan suatu aplikasi serta mengidentifikasi permasalahan yang dialami pengguna dalam berinteraksi dengan sistem sehingga dapat dilakukan perbaikan pada desain antarmuka aplikasi [23].

Berdasarkan analisis tersebut, dapat disimpulkan bahwa resep masakan digital memiliki keunggulan yang signifikan dibandingkan metode konvensional, terutama dalam hal aksesibilitas informasi, efisiensi pencarian data, fleksibilitas pembaruan informasi, serta kemampuan interaksi dengan pengguna. Oleh karena itu, pemanfaatan teknologi *mobile* dan sistem informasi digital menjadi solusi yang lebih efektif dalam menyediakan informasi resep masakan yang mudah diakses, interaktif, dan sesuai dengan kebutuhan pengguna di era digital saat ini.

## 2.3 *Framework* Flutter

### 2.3.1 Pengertian Flutter

Flutter adalah *framework open-source* yang dikembangkan oleh Google untuk membangun aplikasi *mobile*, *web*, dan *desktop* menggunakan satu basis kode (*single codebase*). Flutter menggunakan bahasa pemrograman Dart, yang dikembangkan khusus untuk UI yang reaktif dan performa tinggi.

Menurut Gregor et al., Flutter merupakan salah satu teknologi *cross-platform development* yang semakin populer karena kemampuannya menghasilkan aplikasi dengan performa mendekati *native* serta mendukung pengembangan cepat melalui fitur seperti *hot reload* [24].

### 2.3.2 Struktur dan Arsitektur Flutter

Secara teknis, Flutter memiliki struktur arsitektur yang terdiri dari beberapa lapisan utama yang bekerja secara terintegrasi untuk mendukung proses pembangunan aplikasi. Setiap lapisan memiliki fungsi yang berbeda dalam mengelola antarmuka pengguna, pemrosesan grafis, maupun komunikasi dengan sistem operasi perangkat. Struktur arsitektur Flutter secara umum dapat dijelaskan sebagai berikut [25].

1. *Framework Layer*

   *Framework Layer* merupakan lapisan yang berisi kumpulan *widget* dan pustaka yang digunakan untuk membangun antarmuka pengguna (*User Interface*). Pada lapisan ini pengembang dapat menyusun berbagai komponen tampilan seperti teks, tombol, gambar, serta elemen interaktif lainnya. Flutter menggunakan pendekatan berbasis *widget*, dimana setiap elemen dalam tampilan aplikasi direpresentasikan sebagai *widget* yang dapat dikombinasikan serta dikustomisasi sesuai kebutuhan aplikasi.

2. *Engine Layer*

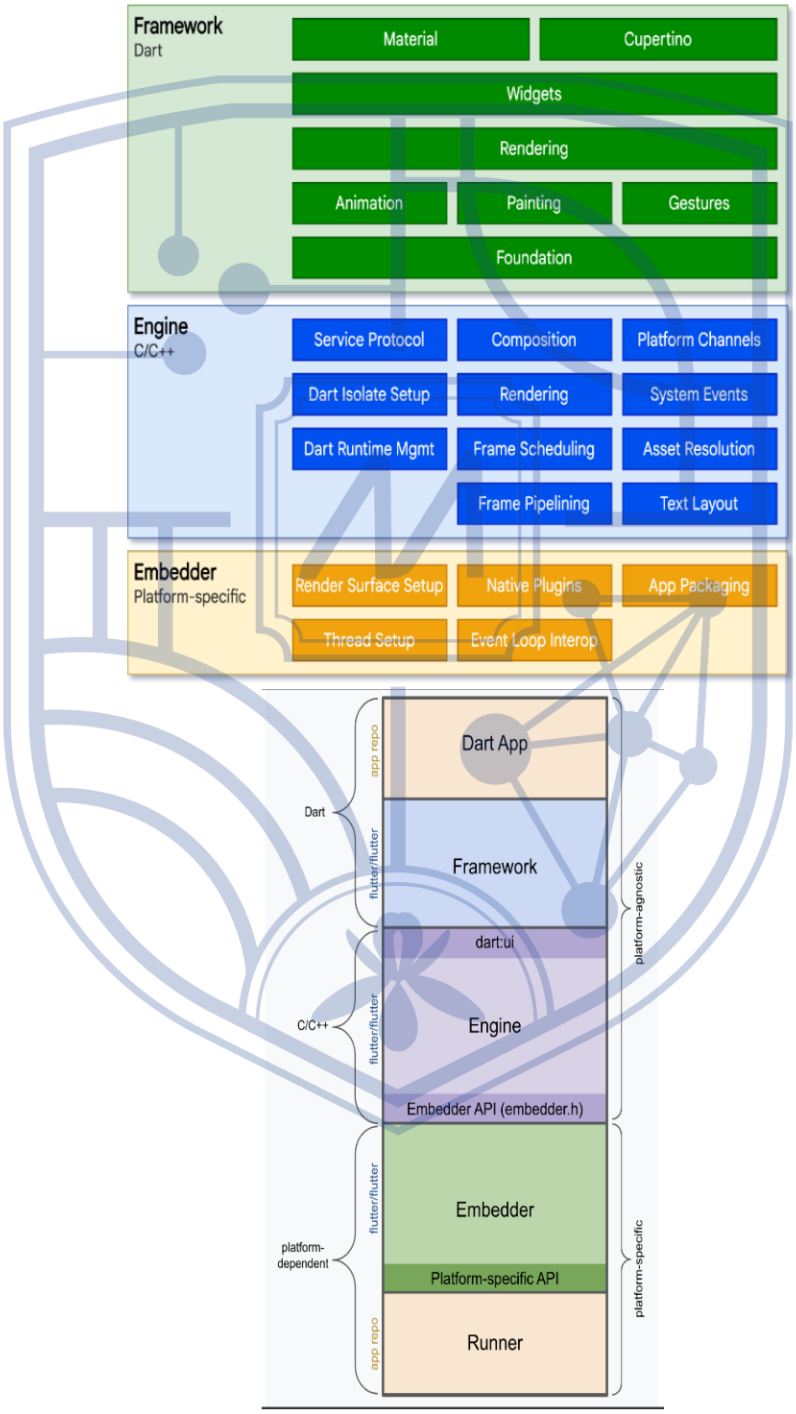
   *Engine Layer* merupakan lapisan inti yang bertanggung jawab terhadap proses *rendering* grafis, animasi, serta pengolahan *input* dari pengguna. Lapisan ini dikembangkan menggunakan bahasa pemrograman C++ dan memanfaatkan *Skia Graphics Engine* untuk menampilkan antarmuka secara langsung pada layar perangkat. Dengan mekanisme tersebut, Flutter mampu menghasilkan performa grafis yang cepat dan responsif saat menjalankan aplikasi *mobile*.

3. *Embedder Layer*

   *Embedder Layer* berfungsi sebagai penghubung antara Flutter dengan sistem operasi perangkat seperti Android maupun iOS. Lapisan ini memungkinkan aplikasi yang dibangun menggunakan Flutter dapat mengakses berbagai fitur perangkat keras maupun layanan sistem operasi seperti kamera, sensor, sistem navigasi, serta berbagai layanan perangkat lainnya.

Melalui arsitektur tersebut, Flutter mampu mengontrol hampir seluruh proses *rendering* antarmuka aplikasi secara mandiri tanpa bergantung pada komponen antarmuka bawaan sistem operasi. Pendekatan ini memberikan keuntungan dalam hal konsistensi

tampilan serta fleksibilitas desain pada berbagai platform perangkat. Selain itu, Flutter juga menyediakan fitur *hot reload*, yaitu kemampuan yang memungkinkan pengembang melihat perubahan kode secara langsung tanpa harus melakukan proses kompilasi ulang secara penuh. Fitur ini sangat membantu dalam proses pengembangan karena dapat mempercepat tahap pengujian serta perbaikan antarmuka aplikasi selama proses implementasi berlangsung [26].




Gambar 2.1 Arsitektur *Framework* Flutter

Beberapa keunggulan Flutter yang menjadikannya banyak digunakan dalam pengembangan aplikasi *mobile* modern dapat dilihat pada Tabel 2.2 berikut.

Tabel 2.2 Keunggulan Flutter

| No. | Keunggulan | Penjelasan |
|---|---|---|
| 1 | *Hot Reload* | Memungkinkan pengembang melihat perubahan kode secara langsung tanpa perlu melakukan kompilasi ulang sehingga mempercepat proses pengembangan aplikasi [24]. |
| 2 | Antarmuka Konsisten Lintas Platform | *Widget* Flutter di-*render* langsung oleh *framework* sehingga menghasilkan tampilan dan perilaku antarmuka yang konsisten pada platform Android maupun iOS [26]. |
| 3 | Performa Mendekati Aplikasi *Native* | Flutter menggunakan proses kompilasi langsung ke kode mesin sehingga aplikasi dapat berjalan dengan cepat dan responsif seperti aplikasi *native* [25]. |
| 4 | Satu Basis Kode untuk Banyak Platform | Pengembang hanya perlu menulis satu basis kode untuk menjalankan aplikasi pada berbagai platform sehingga dapat menghemat waktu dan biaya pengembangan [24]. |
| 5 | Ekosistem Paket yang Luas | Flutter menyediakan berbagai pustaka dan *plugin* yang mendukung pengembangan fitur tambahan seperti integrasi *database*, notifikasi, maupun akses perangkat keras [26]. |

Selain memiliki berbagai keunggulan, Flutter juga memiliki beberapa keterbatasan yang perlu diperhatikan dalam proses pengembangan aplikasi, antara lain sebagai berikut:

Tabel 2.3 Kekurangan Flutter

| No. | Kekurangan | Penjelasan |
|---|---|---|
| 1 | Ukuran Aplikasi Relatif Besar | Aplikasi yang dibangun menggunakan Flutter cenderung memiliki ukuran file (APK/IPA) yang lebih besar dibandingkan aplikasi native. Hal ini disebabkan oleh penyertaan engine Flutter serta library tambahan dalam paket aplikasi. Ukuran aplikasi yang besar dapat menjadi kendala terutama bagi pengguna dengan keterbatasan kapasitas penyimpanan perangkat[27]. |
| 2 | Keterbatasan Akses terhadap Fitur Native Tertentu | Meskipun Flutter menyediakan berbagai plugin, tidak semua fitur native perangkat dapat diakses secara langsung. Dalam beberapa kasus, pengembang perlu menulis kode tambahan menggunakan bahasa native seperti Kotlin (Android) atau Swift (iOS) untuk mengimplementasikan fitur tertentu. Hal ini dapat menambah kompleksitas dalam proses pengembangan [26]. |

| No. | Kekurangan | Penjelasan |
|---|---|---|
| 3 | Ekosistem yang Masih Berkembang | Dibandingkan dengan framework yang lebih lama seperti React Native atau platform native, ekosistem Flutter masih tergolong berkembang. Meskipun jumlah plugin terus meningkat, beberapa pustaka mungkin belum stabil atau belum mendukung kebutuhan spesifik tertentu [24]. |
| 4 | Kurva Pembelajaran (Learning Curve) | Flutter menggunakan bahasa pemrograman Dart yang relatif kurang populer dibandingkan Java, Kotlin, atau JavaScript. Hal ini dapat menjadi hambatan bagi pengembang yang belum familiar, sehingga membutuhkan waktu adaptasi dalam mempelajari sintaks dan konsep pemrograman Flutter [25]. |
| 5 | Ketergantungan pada Rendering Sendiri | Flutter menggunakan rendering engine sendiri (Skia), sehingga tidak memanfaatkan komponen UI bawaan sistem operasi. Meskipun memberikan konsistensi tampilan, pendekatan ini terkadang menyebabkan perbedaan perilaku dengan komponen native, terutama dalam hal integrasi dengan fitur spesifik platform [26]. |

Dalam konteks penelitian ini, penggunaan Flutter bertujuan untuk mendukung pengembangan aplikasi resep masakan berbasis *mobile* yang memiliki antarmuka responsif serta mudah digunakan oleh pengguna. Penelitian sebelumnya menunjukkan bahwa penggunaan Flutter dalam pengembangan aplikasi *mobile* mampu mengurangi waktu pengembangan serta meningkatkan efisiensi penulisan kode dibandingkan dengan beberapa *framework* lintas platform lainnya [25][28]

Selain itu, dari sisi performa Flutter juga menunjukkan kinerja yang kompetitif dibandingkan *framework* lintas platform lainnya. Hasil penelitian yang membandingkan performa Flutter dengan *framework* lain menunjukkan bahwa Flutter mampu menghasilkan *rendering* antarmuka yang responsif dan stabil pada berbagai perangkat. Meskipun ukuran aplikasi yang dihasilkan cenderung lebih besar dibandingkan aplikasi *native*, keunggulan dalam kemudahan pengembangan dan konsistensi antarmuka menjadikan Flutter sebagai pilihan yang tepat untuk implementasi aplikasi *mobile* modern [27].

## 2.4 Metodologi *Agile Scrum*

### 2.4.1 Pengertian *Scrum*

Scrum adalah sebuah *framework* dalam pengembangan perangkat lunak yang termasuk ke dalam pendekatan *Agile*, yang menekankan proses pengembangan secara iteratif dan inkremental. Dalam *Scrum*, pengembangan sistem dilakukan melalui siklus kerja pendek yang disebut *sprint*, sehingga tim dapat menghasilkan produk secara bertahap serta mampu beradaptasi dengan perubahan kebutuhan pengguna selama proses pengembangan

berlangsung. *Framework* ini memfokuskan pada pengelolaan proyek melalui pembagian peran yang jelas, penggunaan artefak sebagai dokumentasi kerja, serta serangkaian kegiatan atau *ceremony* yang memastikan proses pengembangan sistem berjalan secara kolaboratif, transparan, dan adaptif terhadap perubahan kebutuhan [29].

Dalam penerapannya, *Scrum* memiliki tiga komponen utama yang menjadi dasar proses pengembangan perangkat lunak, yaitu *roles* (peran), *artifacts* (artefak), dan *events* (kegiatan). Peran dalam *Scrum* terdiri dari *Product Owner*, *Scrum Master*, dan *Development Team*. *Product Owner* bertanggung jawab terhadap pengelolaan kebutuhan produk serta menentukan prioritas pekerjaan dalam *product backlog*. *Scrum Master* berperan sebagai fasilitator yang memastikan proses *Scrum* berjalan dengan baik serta membantu tim mengatasi hambatan yang muncul selama pengembangan. Sementara itu, *Development Team* merupakan tim yang bertugas mengembangkan dan menghasilkan fitur-fitur sistem sesuai dengan kebutuhan yang telah ditentukan [30].

Dengan karakteristik yang fleksibel dan adaptif, *Scrum* banyak digunakan dalam pengembangan perangkat lunak modern karena mampu meningkatkan kolaborasi tim, mempercepat proses pengembangan, serta memungkinkan penyesuaian kebutuhan sistem secara cepat sesuai dengan umpan balik pengguna. Oleh karena itu, penggunaan *Scrum* menjadi salah satu pendekatan yang efektif dalam pengembangan sistem berbasis teknologi informasi yang membutuhkan proses pengembangan yang dinamis dan berorientasi pada kebutuhan pengguna [30].

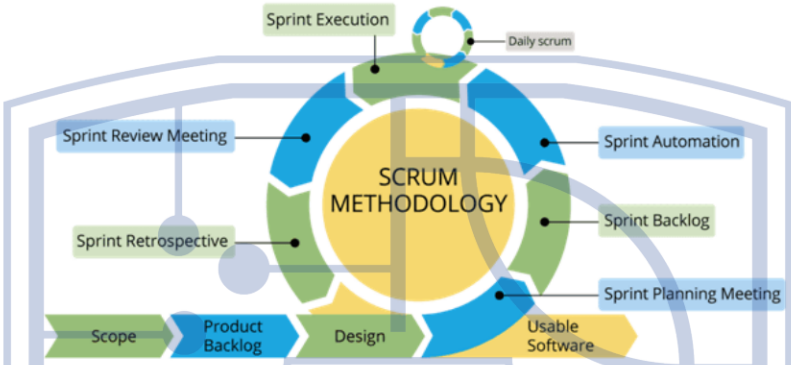
### 2.4.2 Komponen Utama *Scrum*

*Scrum* terdiri dari elemen-elemen berikut:

1. Peran (*Roles*)
   a. *Product Owner* – Bertanggung jawab menentukan prioritas fitur dan kebutuhan melalui *Product Backlog*.
   b. *Scrum Master* – Membantu tim agar dapat menerapkan prinsip *Scrum* dengan benar.
   c. *Development Team* – Kelompok pengembang yang bertugas mengimplementasikan *backlog* menjadi fitur yang berfungsi.
2. Artefak
   a. *Product Backlog* – Daftar kebutuhan atau fitur aplikasi secara keseluruhan.
   b. *Sprint Backlog* – Kebutuhan yang dipilih untuk diselesaikan pada satu periode *sprint*.
   c. *Increment* – Hasil kerja yang siap diuji atau disajikan.
3. *Event* (*Scrum Ceremonies*)

a. *Sprint Planning* – Perencanaan *backlog* untuk *sprint*.
b. *Daily Scrum* – Pertemuan singkat harian untuk melihat progres.
c. *Sprint Review* – Evaluasi hasil *sprint* bersama pemangku kepentingan.
d. *Sprint Retrospective* – Refleksi tim untuk memperbaiki proses *sprint* berikutnya.

Dengan struktur ini, *Scrum* mendukung siklus kerja yang terus menerus melalui *sprint-sprint* yang pendek dan fokus pada penyerahan nilai fungsional.




Gambar 2.2 Tahapan *Agile Scrum*

### 2.4.3 Alasan Pemilihan *Scrum* dalam Pengembangan Aplikasi *Mobile*

Dalam pengembangan aplikasi *mobile* seperti sistem resep masakan, *Scrum* dipilih karena beberapa alasan utama yang didukung oleh temuan penelitian terkini:

1. Fleksibilitas terhadap Perubahan Kebutuhan

   Studi kasus menunjukkan bahwa *Scrum* dapat menghadapi perubahan kebutuhan yang dinamis dalam proyek digital, termasuk pengembangan fitur baru berdasarkan *feedback* pengguna secara cepat selama *sprint* berlangsung [29].
2. Peningkatan Efisiensi Kolaborasi Tim

   Implementasi *Scrum* dalam pengembangan aplikasi *mobile* terbukti meningkatkan efisiensi tim melalui rutinitas *Daily Scrum*, yang mempercepat identifikasi hambatan dan pemecahan masalah antaranggota tim [31].
3. Iterasi Pengembangan dan Evaluasi Berkelanjutan

   Pendekatan *sprint* yang berulang memberikan kesempatan untuk evaluasi fitur secara berkala sehingga kualitas perangkat lunak dapat meningkat secara bertahap dan risiko kesalahan dapat dikurangi sebelum masuk ke fase berikutnya [29].

4. Adaptasi terhadap Kebutuhan Pasar yang Cepat Berubah

   *Scrum* membantu tim mempertahankan ritme pengembangan berkelanjutan pada produk yang mengalami perubahan kebutuhan konsumen yang cepat, seperti penambahan kategori resep baru, fitur pencarian lanjutan, atau sinkronisasi data *realtime* [29][31].

## 2.5 Firebase Firestore

### 2.5.1 Pengertian Firebase

Firebase Firestore merupakan layanan basis data NoSQL berbasis *cloud* yang disediakan oleh Google melalui platform Firebase untuk mendukung penyimpanan serta pengelolaan data pada aplikasi *mobile* dan *web*. Firestore dirancang sebagai basis data modern yang menggunakan pendekatan *document-oriented database*, dimana data disimpan dalam bentuk dokumen yang terorganisasi dalam suatu koleksi. Setiap dokumen berisi sekumpulan pasangan *key–value* yang dapat merepresentasikan berbagai tipe data, sehingga memungkinkan penyimpanan informasi aplikasi secara fleksibel tanpa memerlukan skema tabel yang kaku seperti pada basis data relasional. Pendekatan ini memberikan kemudahan bagi pengembang dalam menyesuaikan struktur data sesuai dengan kebutuhan sistem yang terus berkembang [32].

Dalam implementasinya, Firestore menggunakan struktur data yang bersifat hierarkis, yaitu terdiri dari koleksi (*collection*), dokumen (*document*), serta subkoleksi (*subcollection*). Koleksi berfungsi sebagai wadah yang menyimpan sekumpulan dokumen, sedangkan setiap dokumen dapat berisi data maupun subkoleksi tambahan. Struktur ini memungkinkan pengelolaan data yang lebih terorganisasi serta mempermudah proses pengambilan data secara spesifik sesuai kebutuhan aplikasi. Selain itu, Firestore menyediakan mekanisme *query* yang memungkinkan pengembang melakukan pencarian, penyaringan, serta pengurutan data berdasarkan atribut tertentu sehingga proses pengolahan data dalam aplikasi dapat dilakukan secara lebih efisien [32].

Firestore juga dirancang untuk mendukung pengembangan aplikasi yang membutuhkan sistem penyimpanan data yang fleksibel dan mudah diakses melalui berbagai platform. Melalui penggunaan *Application Programming Interface* (API) yang disediakan oleh Firebase, pengembang dapat mengintegrasikan Firestore dengan berbagai *framework* pengembangan aplikasi, termasuk *framework mobile* modern seperti Flutter maupun aplikasi berbasis *web*. Dengan pendekatan tersebut, Firestore tidak hanya berfungsi sebagai tempat penyimpanan data, tetapi juga sebagai komponen penting dalam arsitektur aplikasi modern yang mendukung pengelolaan data secara terpusat dan terstruktur [33].

### 2.5.2 Keunggulan Firebase Firestore

Berdasarkan beberapa studi ilmiah terbaru, Firestore memiliki sejumlah keunggulan penting yang mendukung pengembangan aplikasi *mobile* modern:

1. Sinkronisasi Data *Real-Time*

   Firestore secara otomatis menyinkronkan perubahan data antarperangkat pengguna secara *real-time* melalui koneksi yang transparan, sehingga setiap perubahan seperti penambahan resep atau *update list* favorit dapat langsung terlihat tanpa perlu *refresh* manual [32][33].

2. Dukungan *Offline*

   Firestore menyediakan dukungan akses data saat *offline*, memungkinkan aplikasi tetap menampilkan data yang sudah dimuat meskipun tidak terkoneksi internet. Saat koneksi kembali tersedia, data akan otomatis tersinkronisasi [32][33].

3. Infrastruktur Terkelola dengan Skalabilitas Tinggi

   Karena berjalan di infrastruktur Google *Cloud*, Firestore menawarkan skalabilitas yang tinggi dan dapat menangani pertumbuhan jumlah pengguna serta volume data aplikasi tanpa konfigurasi *server* sendiri [32][34].

4. Integrasi dengan Layanan *Cloud* Lain

   Firestore terintegrasi dengan layanan Firebase lain seperti *authentication* untuk manajemen pengguna dan *Cloud Functions* untuk logika *backend*, memudahkan pengembangan aplikasi tanpa *backend* terpisah [32][33].

## 2.6 Cloudinary

### 2.6.1 Pengertian dan Fungsi Cloudinary

Cloudinary merupakan layanan berbasis *cloud computing* yang digunakan untuk menyimpan, mengelola, mengoptimasi, serta mendistribusikan media digital seperti gambar dan video. Layanan ini menyediakan antarmuka berbasis *Application Programming Interface* (API) yang memungkinkan integrasi dengan berbagai platform pengembangan aplikasi, baik berbasis web maupun mobile[35].

Dalam pengembangan sistem informasi, Cloudinary memiliki beberapa fungsi utama, yaitu sebagai media penyimpanan file gambar, pengelola media, serta alat optimasi gambar secara otomatis. Selain itu, Cloudinary juga mendukung transformasi gambar seperti pengaturan ukuran dan resolusi, serta distribusi media melalui *Content Delivery Network* (CDN) sehingga dapat diakses dengan cepat dari berbagai lokasi[36].

### 2.6.2 Cara Kerja dan Keunggulan Cloudinary

Mekanisme kerja Cloudinary dalam suatu sistem dimulai dari proses pemilihan gambar oleh pengguna melalui aplikasi. Selanjutnya, aplikasi mengunggah gambar tersebut ke Cloudinary melalui API. Setelah proses upload selesai, Cloudinary akan menyimpan dan memproses gambar secara otomatis, kemudian menghasilkan URL unik yang digunakan sebagai alamat akses media[37].

Dengan mekanisme tersebut, sistem tidak perlu menyimpan file gambar secara langsung dalam database, melainkan hanya menyimpan referensi berupa URL. Pendekatan ini memberikan beberapa keunggulan, antara lain efisiensi penyimpanan, optimasi ukuran gambar secara otomatis, serta kecepatan akses yang lebih baik melalui CDN. Selain itu, penggunaan Cloudinary juga dapat mengurangi beban server utama dalam pengelolaan media [36][37].

### 2.6.3 Penerapan Cloudinary pada Sebuah Aplikasi

Cloudinary digunakan sebagai layanan penyimpanan media yang terintegrasi dengan Firebase. Firebase dimanfaatkan untuk autentikasi pengguna dan pengelolaan data menggunakan Cloud Firestore, sedangkan Cloudinary digunakan untuk menyimpan gambar resep. Proses integrasi dilakukan dengan cara mengunggah gambar ke Cloudinary melalui API, kemudian sistem menerima URL hasil upload yang selanjutnya disimpan dalam Firebase Firestore[35]. Dengan demikian, pengelolaan data teks dan media dilakukan secara terpisah namun tetap terintegrasi.

Penggunaan Cloudinary memberikan beberapa manfaat, antara lain meningkatkan efisiensi penyimpanan, mempercepat akses gambar, serta mempermudah pengelolaan media dalam sistem. Oleh karena itu, pemanfaatan Cloudinary mendukung peningkatan performa aplikasi yang dikembangkan.

## 2.7 Alat Bantu Pengembangan Sistem

### 2.7.1 *Use Case Diagram*

Use Case Diagram merupakan salah satu diagram dalam Unified Modeling Language (UML) yang digunakan untuk memodelkan kebutuhan fungsional suatu sistem berdasarkan interaksi antara aktor dan sistem. Diagram ini menggambarkan hubungan antara pengguna (aktor) dengan fungsi-fungsi sistem (use case) yang disediakan, sehingga memberikan gambaran mengenai bagaimana sistem akan digunakan dari sudut pandang eksternal.

Dalam rekayasa perangkat lunak, UML telah menjadi standar pemodelan yang digunakan untuk merepresentasikan struktur dan perilaku sistem secara visual. Penggunaan Use Case Diagram sangat penting pada tahap analisis kebutuhan karena mampu menyederhanakan kompleksitas sistem menjadi representasi yang mudah dipahami oleh pengembang maupun pemangku kepentingan. Studi literatur menunjukkan bahwa UML digunakan secara luas untuk mengidentifikasi kebutuhan sistem serta menentukan ruang lingkup aplikasi melalui model visual yang terstruktur [38].

Lebih lanjut, Use Case Diagram dianggap sebagai salah satu alat yang efektif dalam proses rekayasa kebutuhan (requirements engineering) karena mampu menjembatani komunikasi antara pengguna dan pengembang sistem. Penelitian terbaru menyatakan bahwa use case diagram merupakan alat yang sangat kuat untuk memodelkan kebutuhan pengguna serta membantu tim pengembang memahami interaksi dan fungsionalitas sistem secara menyeluruh [39]. Selain itu, model use case juga digunakan sebagai representasi awal dari blueprint sistem sebelum memasuki tahap desain dan implementasi [40].

### 2.7.2 Fungsi *Use Case Diagram* dalam Pengembangan Sistem

Use Case Diagram memiliki peranan yang sangat penting dalam siklus pengembangan perangkat lunak, khususnya pada tahap analisis dan perancangan sistem. Penggunaan diagram ini membantu pengembang dalam memahami kebutuhan sistem secara menyeluruh sebelum proses implementasi dilakukan.

Adapun fungsi utama Use Case Diagram dalam pengembangan sistem adalah sebagai berikut:

1. Mengidentifikasi kebutuhan fungsional system

   Use Case Diagram digunakan untuk mengidentifikasi fungsi-fungsi utama yang harus disediakan oleh sistem berdasarkan kebutuhan pengguna.

2. Menyajikan interaksi antara aktor dan system

   Diagram ini menggambarkan bagaimana pengguna berinteraksi dengan sistem melalui berbagai layanan yang tersedia.

3. Menentukan ruang lingkup sistem (system scope)

   Dengan adanya system boundary, pengembang dapat menentukan batasan sistem yang akan dikembangkan secara jelas.

4. Sebagai dasar perancangan sistem lanjutan

   Use Case Diagram menjadi acuan dalam pembuatan model lanjutan seperti class diagram, sequence diagram, dan activity diagram.

5. Mendukung komunikasi antar pemangku kepentingan

   Representasi visual yang sederhana memudahkan komunikasi antara pengembang, analis sistem, dan pengguna.

Penelitian terbaru menunjukkan bahwa model use case memiliki peran penting dalam meningkatkan kualitas desain sistem karena membantu dalam validasi kebutuhan serta mengurangi kesalahan interpretasi pada tahap awal pengembangan [41]. Selain itu, penggunaan UML dalam proses analisis dan desain sistem juga terbukti mampu meningkatkan kualitas dokumentasi perangkat lunak dan efisiensi pengembangan system, karena mampu memvisualisasikan kebutuhan dan struktur sistem secara jelas dan terstandarisasi [42].

Dengan demikian, Use Case Diagram tidak hanya berfungsi sebagai alat visualisasi, tetapi juga sebagai dasar konseptual dalam membangun sistem yang sesuai dengan kebutuhan pengguna.

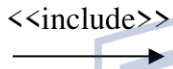
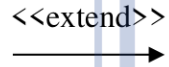
### 2.7.3 Simbol dan Notasi *Use Case Diagram*

Dalam Use Case Diagram, terdapat sejumlah simbol dan notasi yang digunakan untuk merepresentasikan elemen-elemen sistem secara standar. Notasi ini mengacu pada standar UML yang telah diakui secara luas dalam bidang rekayasa perangkat lunak, sehingga memastikan konsistensi dalam proses pemodelan sistem [38].

Berikut adalah tabel simbol dan notasi Use Case Diagram beserta penjelasannya:

Tabel 2.4 Simbol dan Notasi *Use Case Diagram*

| No | Simbol | Nama | Deskripsi |
|---|---|---|---|
| 1 |  | Aktor (Actor) | Merepresentasikan entitas eksternal yang berinteraksi dengan sistem. Aktor dapat berupa pengguna (user) atau sistem lain yang berada di luar sistem utama. |
| 2 |  | Use Case | Menggambarkan fungsi atau layanan yang disediakan oleh sistem kepada aktor. Setiap use case merepresentasikan satu proses atau aktivitas sistem. |

| No | Simbol | Nama | Deskripsi |
|---|---|---|---|
| 3 |  | System Boundary | Menunjukkan batasan sistem yang sedang dikembangkan. Seluruh use case berada di dalam batas sistem ini. |
| 4 |  | Association | Menunjukkan hubungan komunikasi antara aktor dan use case. Relasi ini menggambarkan interaksi langsung antara pengguna dan sistem. |
| 5 |   | Include | Menunjukkan bahwa suatu use case selalu melibatkan use case lain sebagai bagian dari prosesnya (bersifat wajib). |
| 6 |   | Extend | Menunjukkan bahwa suatu use case dapat memperluas fungsi use case lain dalam kondisi tertentu (bersifat opsional). |
| 7 |  | Generalization | Menunjukkan hubungan pewarisan antara aktor atau use case yang memiliki karakteristik serupa. |

Contoh skenario *use case*:

Tabel 2.5 Contoh Skenario *Use Case*

| Elemen | Deskripsi |
|---|---|
| Nama Use Case | Interaksi Pengguna dengan Sistem |
| Aktor | Pengguna (User) |
| Deskripsi | Menggambarkan proses umum interaksi pengguna dalam mengakses dan menggunakan fitur yang tersedia pada sistem |
| Pre-condition | Sistem telah berjalan dan pengguna telah membuka aplikasi |
| Post-condition | Sistem menampilkan hasil sesuai dengan permintaan pengguna |
| Alur Utama | 1. Pengguna membuka aplikasi<br>2. Pengguna memilih atau mengakses salah satu fitur yang tersedia<br>3. Pengguna memasukkan data atau perintah yang diperlukan<br>4. Sistem memproses input yang diberikan |

| | |
|---|---|
| | 5. Sistem menampilkan hasil atau informasi sesuai dengan permintaan pengguna |
| Alur Alternatif | 1. Jika input tidak valid, sistem menampilkan pesan kesalahan<br>2. Jika terjadi kegagalan sistem, maka sistem memberikan notifikasi kepada pengguna |
| Exception | Terjadi gangguan pada sistem atau koneksi sehingga proses tidak dapat dilanjutkan |

### 2.7.4 *Class Diagram*

*Class diagram* merupakan salah satu diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk memodelkan struktur statis dari suatu sistem berbasis objek. Diagram ini menggambarkan kelas-kelas yang terdapat dalam sistem, atribut yang dimiliki oleh masing-masing kelas, metode atau operasi yang dapat dilakukan, serta hubungan antarkelas yang membentuk suatu kesatuan sistem perangkat lunak.

Dalam pengembangan perangkat lunak modern, UML digunakan sebagai standar pemodelan karena mampu memberikan representasi visual yang sistematis dan mudah dipahami oleh pengembang maupun pemangku kepentingan lainnya. *Class diagram* secara khusus berperan dalam menjelaskan struktur internal sistem serta hubungan antarkomponen sebelum sistem diimplementasikan ke dalam bentuk kode program [38].

Selain itu, penggunaan *class diagram* dapat membantu dalam mengurangi kesalahan desain pada tahap awal pengembangan, karena seluruh struktur sistem telah dirancang dan divisualisasikan secara jelas [43]. Dengan demikian, *class diagram* menjadi salah satu alat bantu yang penting dalam pendekatan *Object-Oriented Programming* (OOP).

### 2.7.5 Fungsi *Class Diagram*

Dalam penelitian ini, penggunaan *class diagram* memiliki beberapa fungsi utama, yaitu sebagai berikut:

1. Memodelkan struktur sistem

   *Class diagram* digunakan untuk menggambarkan struktur sistem aplikasi resep masakan berbasis *mobile* yang terdiri dari beberapa entitas utama seperti *User*, Resep, Kategori, *Bookmark*, dan *Meal Plan*.
2. Sebagai acuan implementasi sistem

Diagram yang telah dirancang menjadi dasar dalam proses pengkodean menggunakan bahasa pemrograman Dart pada *framework* Flutter.

3. Mempermudah proses analisis dan perancangan

   Dengan adanya visualisasi sistem, pengembang dapat memahami hubungan antarkomponen secara lebih sistematis.

4. Sebagai dokumentasi sistem

   *Class diagram* dapat digunakan sebagai dokumentasi teknis yang berguna dalam proses pengembangan lanjutan maupun pemeliharaan sistem.

Penggunaan UML dalam proses perancangan terbukti dapat meningkatkan kualitas desain perangkat lunak serta mempermudah proses pengembangan berbasis objek [44].

### 2.7.6 Simbol dan Notasi pada *Class Diagram*

Dalam perancangan sistem berbasis objek, penggunaan simbol dan notasi pada *class diagram* memiliki peranan penting dalam menggambarkan struktur sistem secara jelas dan terstandarisasi. Notasi yang digunakan mengacu pada standar *Unified Modeling Language* (UML) yang telah diakui secara luas dalam rekayasa perangkat lunak [38].

1. Notasi Kelas (*Class*)

   Kelas merupakan representasi dari entitas utama dalam sistem yang memiliki atribut dan metode. Secara umum, kelas digambarkan dalam bentuk persegi panjang yang terbagi menjadi tiga bagian, yaitu:

   a. Nama Kelas (*Class Name*)

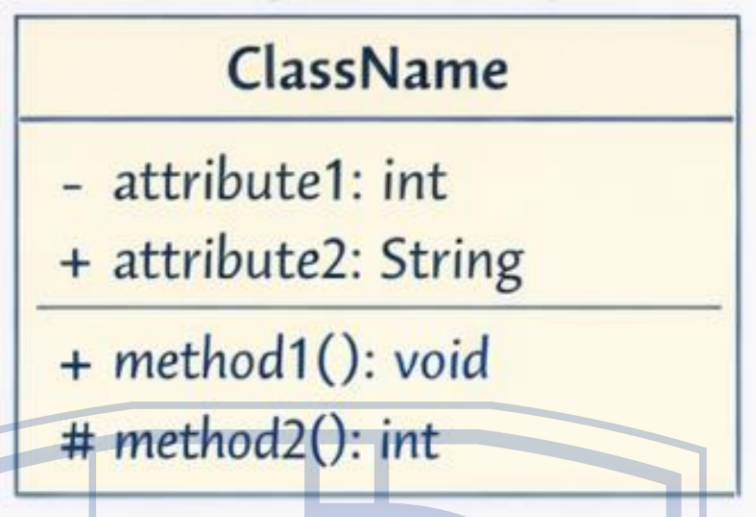
      Ditulis pada bagian atas dan biasanya menggunakan huruf kapital di awal kata.

   b. Atribut (*Attributes*)

      Berisi data atau karakteristik yang dimiliki oleh kelas.

   c. Metode (*Operations/Methods*)

      Berisi fungsi atau operasi yang dapat dilakukan oleh kelas.




Gambar 2.3 Struktur Umum Notasi Kelas

2. *Visibility* (Hak Akses)

   Setiap atribut dan metode memiliki hak akses (*visibility*) yang menunjukkan tingkat aksesibilitasnya, yaitu:

   a. *Public* (+): dapat diakses oleh semua kelas
   b. *Private* (-): hanya dapat diakses dalam kelas itu sendiri
   c. *Protected* (#): dapat diakses oleh kelas turunan

   Penggunaan *visibility* ini penting untuk menjaga prinsip enkapsulasi dalam OOP [43].

3. Notasi Relasiantar Kelas

   Relasi antarkelas digunakan untuk menunjukkan keterkaitan antarobjek dalam sistem. Adapun jenis relasi yang digunakan dalam penelitian ini adalah sebagai berikut:

   a. *Association*

      *Association* merupakan hubungan dasar antarkelas yang menunjukkan adanya keterkaitan antarobjek.

      Contoh:

      *User* -------- Resep

      Relasi ini dapat dilengkapi dengan *multiplicity*, seperti:

      i. 1 (satu)
      ii. 0..* (nol atau lebih)
      iii. 1..* (satu atau lebih)

   b. *Aggregation*

      *Aggregation* menunjukkan hubungan kepemilikan lemah, dimana objek dapat berdiri sendiri meskipun tidak terhubung dengan objek lain.

      Simbol: ◇--------

c. *Composition*

*Composition* merupakan hubungan kepemilikan kuat, dimana keberadaan objek bergantung pada objek induknya.

Simbol: ◆--------

d. *Generalization* (*Inheritance)*

*Generalization* menunjukkan hubungan pewarisan antara kelas induk (*superclass*) dan kelas turunan (*subclass*).

Simbol: △--------

e. *Dependency*

*Dependency* menunjukkan hubungan ketergantungan sementara antara satu kelas dengan kelas lainnya.

Simbol: - - - >

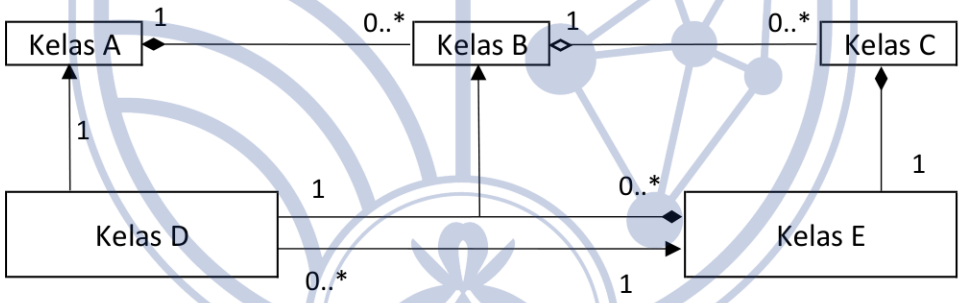
4. *Multiplicity* dan *Navigability*

*Multiplicity* digunakan untuk menunjukkan jumlah objek yang terlibat dalam suatu relasi, sedangkan *navigability* menunjukkan arah hubungan antarkelas.

Contoh:

*User* 1 -------- 0..* *Bookmark*

Artinya, satu *user* dapat memiliki banyak *bookmark*.




Gambar 2.4 Struktur Umum Diagram UML

Keterangan:

1. Kelas A → Kelas B, artinya satu objek A memiliki nol atau banyak B, dan B tidak dapat berdiri sendiri tanpa A.
2. Kelas B → Kelas C, artinya satu objek B berhubungan dengan nol atau banyak C, dan C dapat berdiri sendiri.
3. Kelas C → Kelas E, artinya satu C memiliki tepat satu E, dan E bergantung penuh pada C.
4. Kelas D → Kelas A, artinya D adalah turunan dari A (mewarisi atribut dan metode A).

5. Kelas D → Kelas B, artinya satu D berhubungan dengan satu B.
6. Kelas D → Kelas E, artinya D dapat berhubungan dengan banyak E, tetapi dalam kondisi tertentu minimal ada 1 E.

## 2.8 *Black Box Testing*

### 2.8.1 Pengertian *Black Box Testing*

*Black Box Testing* merupakan salah satu metode pengujian perangkat lunak yang berfokus pada pengujian fungsi sistem tanpa memperhatikan struktur internal atau kode program yang digunakan. Dalam metode ini, pengujian dilakukan dengan memberikan berbagai masukan (*input*) kepada sistem kemudian mengamati keluaran (*output*) yang dihasilkan untuk memastikan bahwa sistem dapat berfungsi sesuai dengan spesifikasi kebutuhan yang telah ditentukan. Pendekatan ini memungkinkan pengujian dilakukan dari sudut pandang pengguna sehingga sistem dapat dievaluasi berdasarkan perilaku fungsional yang terlihat dari luar sistem [45].

Metode *Black Box Testing* banyak digunakan dalam proses pengujian perangkat lunak karena mampu memverifikasi apakah suatu aplikasi dapat menerima *input* dengan benar dan menghasilkan *output* yang sesuai dengan kebutuhan sistem. Pengujian ini dilakukan dengan berbagai skenario pengujian yang dirancang untuk mengidentifikasi kesalahan yang mungkin terjadi pada fungsi aplikasi tanpa harus memahami struktur internal dari program tersebut [46].

Selain itu, *Black Box Testing* juga berperan penting dalam mengidentifikasi berbagai kesalahan yang berkaitan dengan fungsi sistem, seperti kesalahan validasi *input*, kesalahan dalam pengolahan data, maupun kesalahan pada antarmuka pengguna. Melalui pengujian ini, pengembang dapat memastikan bahwa sistem yang dikembangkan dapat berjalan dengan baik serta mampu memenuhi kebutuhan pengguna sebelum aplikasi diimplementasikan secara luas [45].

### 2.8.2 Tujuan *Black Box Testing*

Penggunaan metode *Black Box Testing* bertujuan untuk memastikan bahwa sistem yang dikembangkan telah memenuhi kebutuhan fungsional yang telah ditentukan. Melalui pengujian ini, sistem diuji berdasarkan skenario *input* dan *output* untuk memastikan bahwa aplikasi dapat menghasilkan keluaran yang sesuai dengan kebutuhan pengguna. Pendekatan ini banyak digunakan dalam pengujian perangkat lunak untuk memvalidasi fungsi sistem tanpa melihat struktur internal program [45].

Secara lebih rinci, tujuan dari penerapan *Black Box Testing* dalam pengujian perangkat lunak antara lain sebagai berikut:

1. Memastikan kesesuaian fungsi sistem dengan spesifikasi kebutuhan
   Pengujian dilakukan untuk memastikan bahwa setiap fitur dalam sistem dapat berjalan sesuai dengan kebutuhan yang telah dirancang [45].
2. Mengidentifikasi kesalahan pada proses *input* dan *output* sistem.
   Pengujian *Black Box* dilakukan dengan memberikan berbagai variasi *input* untuk mengetahui apakah sistem mampu menghasilkan *output* yang sesuai dengan spesifikasi sistem [46].
3. Menemukan kesalahan pada antarmuka pengguna (*user interface*)
   Teknik pengujian ini membantu mengidentifikasi kesalahan yang terjadi pada proses validasi data maupun interaksi pengguna dengan sistem [46].
4. Memverifikasi integrasi antarmodul sistem
   Pengujian dilakukan untuk memastikan bahwa seluruh fitur sistem dapat berjalan sesuai dengan fungsinya sebelum aplikasi digunakan oleh pengguna [45].
5. Meningkatkan kualitas dan keandalan perangkat lunak
   Dengan melakukan pengujian terhadap berbagai fitur yang tersedia, pengembang dapat memastikan bahwa aplikasi yang dihasilkan memiliki kualitas yang baik sebelum diimplementasikan kepada pengguna [47].

### 2.8.3 Teknik dalam *Black Box Testing*

Dalam proses pengujian perangkat lunak menggunakan metode *Black Box Testing* terdapat berbagai teknik yang dapat digunakan untuk menyusun skenario pengujian secara sistematis. Teknik-teknik ini membantu penguji dalam menentukan kombinasi *input* dan kondisi yang tepat sehingga kesalahan sistem dapat diidentifikasi secara lebih efektif. Berbagai teknik tersebut dikembangkan dalam bidang rekayasa perangkat lunak untuk meningkatkan kualitas proses pengujian aplikasi [48].

Beberapa teknik yang umum digunakan dalam *Black Box Testing* antara lain *Equivalence Partitioning, Boundary Value Analysis, Decision Table Testing, State Transition Testing, Use Case Testing, Cause-Effect Graphing, Error Guessing,* dan *Comparison Testing*. Setiap teknik memiliki karakteristik yang berbeda dalam proses pengujian perangkat lunak serta digunakan sesuai dengan kebutuhan pengujian sistem yang dikembangkan [49].

Untuk memberikan gambaran yang lebih jelas mengenai perbedaan masing-masing teknik tersebut, berikut disajikan tabel perbandingan teknik *Black Box Testing*.

Tabel 2.6 Perbandingan Teknik *Black Box Testing*

| No. | Teknik Pengujian | Deskripsi | Tujuan Penggunaan |
|---|---|---|---|
| 1 | *Equivalence Partitioning* | Membagi data *input* ke dalam beberapa kelompok yang memiliki karakteristik serupa. | Mengurangi jumlah kasus uji dengan tetap mewakili seluruh kemungkinan *input*. |
| 2 | *Boundary Value Analysis* | Menguji nilai batas minimum dan maksimum dari suatu *input*. | Mengidentifikasi kesalahan yang sering muncul pada nilai batas. |
| 3 | *Decision Table Testing* | Menggunakan tabel keputusan untuk menguji berbagai kombinasi kondisi dan aturan. | Memastikan semua kemungkinan kombinasi kondisi telah diuji. |
| 4 | *State Transition Testing* | Menguji perubahan kondisi atau status sistem berdasarkan suatu peristiwa tertentu. | Memastikan sistem dapat berpindah antarkondisi dengan benar. |
| 5 | *Use Case Testing* | Pengujian dilakukan berdasarkan skenario penggunaan sistem oleh pengguna. | Memastikan sistem berfungsi sesuai dengan proses bisnis yang dirancang. |
| 6 | *Cause-Effect Graphing* | Menggambarkan hubungan antara kondisi *input* (*cause*) dengan *output* (*effect*). | Menentukan kombinasi pengujian yang relevan berdasarkan hubungan sebab-akibat. |
| 7 | *Error Guessing* | Pengujian berdasarkan pengalaman penguji dalam memprediksi kemungkinan kesalahan. | Menemukan *bug* yang mungkin tidak terdeteksi oleh teknik pengujian formal. |
| 8 | *Comparison Testing* | Membandingkan hasil *output* dari beberapa sistem atau implementasi yang berbeda. | Memastikan konsistensi hasil dari sistem yang memiliki fungsi serupa. |

Teknik-teknik tersebut dapat digunakan secara terpisah maupun dikombinasikan dalam proses pengujian perangkat lunak untuk memastikan bahwa seluruh fungsi sistem telah diuji secara menyeluruh. Dengan menggunakan teknik yang tepat, proses pengujian dapat dilakukan secara lebih efisien serta mampu meningkatkan kualitas perangkat lunak yang dikembangkan [48].

### 2.8.4 Penerapan *Black Box Testing* pada Sebuah Aplikasi

Pada penelitian ini, metode Black Box Testing digunakan untuk menguji fungsionalitas sistem tanpa memperhatikan struktur internal atau kode program yang digunakan. Pengujian dilakukan dengan memberikan berbagai variasi input pada sistem, kemudian mengevaluasi output yang dihasilkan untuk memastikan kesesuaiannya dengan spesifikasi kebutuhan perangkat lunak[47].

Sebagai contoh penerapan, pengujian dilakukan pada fitur login pengguna dalam sebuah aplikasi. Fitur ini merupakan salah satu komponen penting yang berfungsi untuk memverifikasi identitas pengguna sebelum mengakses sistem. Oleh karena itu, diperlukan pengujian yang menyeluruh untuk memastikan bahwa proses autentikasi berjalan dengan baik dan sesuai dengan skenario yang telah dirancang.

Adapun beberapa skenario pengujian yang dilakukan adalah sebagai berikut:

1. Pengujian dengan data valid

   Pengujian ini dilakukan dengan memasukkan kombinasi *username* dan *password* yang benar. Hasil yang diharapkan adalah sistem dapat menerima data tersebut dan mengarahkan pengguna ke halaman utama aplikasi.
2. Pengujian dengan data tidak valid

   Pengujian dilakukan dengan memasukkan *username* yang benar namun *password* yang salah. Hasil yang diharapkan adalah sistem menolak proses login dan menampilkan pesan kesalahan yang sesuai.
3. Pengujian dengan data kosong

   Pada pengujian ini, pengguna tidak mengisi salah satu atau seluruh field yang tersedia. Sistem diharapkan dapat memberikan validasi berupa peringatan bahwa data harus diisi sebelum melanjutkan proses login.
4. Pengujian dengan format input tidak sesuai

   Pengujian dilakukan dengan memasukkan data yang tidak sesuai dengan format yang ditentukan, seperti penggunaan karakter yang tidak diperbolehkan. Sistem diharapkan dapat menolak input tersebut dan menampilkan pesan kesalahan yang relevan.

Dalam pelaksanaannya, teknik yang digunakan meliputi *Equivalence Partitioning* untuk mengelompokkan data uji ke dalam kategori valid dan tidak valid, serta *Use Case Testing* untuk memastikan bahwa alur penggunaan fitur login berjalan sesuai dengan skenario yang telah ditentukan.

Melalui penerapan Black Box Testing tersebut, dapat diketahui apakah fitur login dalam aplikasi telah berfungsi secara optimal sesuai dengan kebutuhan sistem. Pengujian ini berperan penting dalam menjamin kualitas perangkat lunak, khususnya dalam aspek fungsionalitas, sehingga aplikasi yang dihasilkan mampu memberikan pengalaman penggunaan yang baik dan sesuai dengan harapan pengguna[50].

## 2.9 *User Acceptance Testing* (UAT)

*User Acceptance Testing* (UAT) merupakan tahap pengujian perangkat lunak yang dilakukan untuk memastikan bahwa sistem yang dikembangkan telah memenuhi kebutuhan, persyaratan, dan harapan pengguna akhir. UAT dilakukan dengan melibatkan pengguna atau pihak yang mewakili pengguna sistem untuk mengevaluasi apakah fungsi dan layanan yang tersedia pada sistem dapat diterima serta berjalan sesuai dengan tujuan pengembangannya[51].

Berbeda dengan pengujian fungsional yang berfokus pada pemeriksaan kesesuaian proses internal maupun fungsi sistem berdasarkan spesifikasi teknis, UAT lebih menitikberatkan pada sudut pandang pengguna (*user perspective*). Pengujian ini bertujuan untuk mengetahui tingkat penerimaan pengguna terhadap sistem sebelum sistem diterapkan secara luas [51][52].

Dalam pelaksanaannya, UAT dilakukan melalui beberapa tahapan, yaitu penyusunan skenario pengujian berdasarkan kebutuhan pengguna, pemberian akses kepada pengguna untuk mencoba sistem, pengumpulan tanggapan pengguna melalui instrumen evaluasi, serta analisis hasil pengujian. Instrumen yang digunakan dalam UAT umumnya berupa kuesioner dengan skala Likert untuk mengukur tingkat persetujuan pengguna terhadap pernyataan yang berkaitan dengan fungsi, kemudahan penggunaan, serta kesesuaian sistem dengan kebutuhan pengguna [52].

UAT memiliki peran penting dalam pengembangan perangkat lunak karena tidak hanya memastikan bahwa sistem telah berjalan sesuai rancangan teknis, tetapi juga memastikan bahwa sistem mampu memberikan manfaat dan pengalaman penggunaan yang sesuai dengan kebutuhan pengguna akhir [52].

## 2.10 *System Usability Scale* (SUS)

*System Usability Scale* (SUS) merupakan metode evaluasi *usability* yang dikembangkan oleh John Brooke pada tahun 1986 untuk mengukur tingkat kemudahan penggunaan suatu sistem[53]. SUS banyak digunakan karena sederhana, cepat, serta mampu memberikan hasil yang reliabel meskipun jumlah responden relatif sedikit. Metode ini dapat diterapkan pada berbagai jenis sistem, termasuk aplikasi web, aplikasi mobile, perangkat lunak, maupun perangkat keras.

SUS menggunakan kuesioner yang terdiri atas 10 pernyataan dengan skala penilaian Likert 1–5, yaitu:

1 = Sangat Tidak Setuju

2 = Tidak Setuju

3 = Netral

4 = Setuju

5 = Sangat Setuju

Sepuluh pernyataan tersebut disusun secara bergantian antara pernyataan positif dan negatif untuk mengurangi bias jawaban responden[53]. Pernyataan bernomor ganjil (1, 3, 5, 7, dan 9) bersifat positif, sedangkan pernyataan bernomor genap (2, 4, 6, 8, dan 10) bersifat negatif.

Perhitungan skor SUS dilakukan dengan tahapan sebagai berikut[53].

1. Untuk pernyataan bernomor ganjil:
   Skor = Nilai Jawaban − 1
2. Untuk pernyataan bernomor genap:
   Skor = 5 − Nilai Jawaban
3. Seluruh skor dari 10 pernyataan dijumlahkan sehingga diperoleh nilai antara 0 hingga 40.
4. Hasil penjumlahan kemudian dikalikan dengan 2,5 sehingga diperoleh skor SUS dengan rentang 0–100.

Rumus perhitungan SUS dituliskan sebagai berikut.

$$\text{Skor SUS} = \left(\sum \text{Skor Pernyataan}\right) \times 2{,}5$$

Nilai SUS yang diperoleh selanjutnya diinterpretasikan untuk mengetahui tingkat usability suatu sistem. Semakin tinggi skor yang diperoleh, semakin baik tingkat kemudahan penggunaan sistem menurut pengguna. Secara umum, skor SUS 68 dianggap sebagai nilai rata-rata (average)[53]. Skor di atas 68 menunjukkan usability yang baik, sedangkan skor di

bawah 68 menunjukkan bahwa sistem masih memerlukan perbaikan dari sisi pengalaman pengguna.

Pada penelitian ini, metode SUS digunakan untuk mengevaluasi tingkat kemudahan penggunaan aplikasi resep masakan berbasis mobile yang telah dikembangkan. Pengujian dilakukan dengan meminta responden menggunakan aplikasi, kemudian mengisi kuesioner SUS. Hasil pengolahan skor digunakan untuk mengetahui tingkat penerimaan pengguna terhadap aplikasi serta menjadi dasar dalam mengevaluasi kualitas usability system