APLIKASI CATALOG RESEP MAKANAN

Disusun untuk memenuhi tugas mata kuliah :

Mobile Programming 2

LAPORAN

Dosen Pengampu:

Niken Riyanti, S.T., M.Kom.

Disusun Oleh:

Sendi Resfiana	232101033
Muhammad Fikri Muhaimin	232101030
Ghefira Zahira Shofa	232101031

Departemen Teknik Informatika
Fakultas Industri Kreatif
UNIVERSITAS TEKNOLOGI BANDUNG 20

DAFTAR ISI

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi digital telah membawa perubahan signifikan dalam berbagai aspek kehidupan manusia, termasuk dalam dunia kuliner. Digitalisasi memungkinkan masyarakat untuk memperoleh akses informasi secara cepat dan efisien, termasuk dalam memperoleh resep makanan. Namun, koleksi resep dalam media cetak dinilai belum mampu memenuhi kebutuhan masyarakat dengan gaya hidup yang dinamis dan serba digital saat ini. Oleh karena itu, pengembangan aplikasi berbasis mobile menjadi solusi inovatif yang menawarkan kemudahan akses, pencarian, hingga implementasi resep secara praktis kapan saja dan di mana saja (Algadrie & Sela, 2024).

Selaras dengan pesatnya pertumbuhan pengguna smartphone, aplikasi katalog resep makanan tidak hanya difungsikan sebagai panduan memasak, tetapi juga sebagai alat edukasi, pengelolaan pola makan sehat, hingga sarana eksplorasi budaya kuliner. Studi sistematik menyimpulkan bahwa aplikasi mobile berperan penting dalam meningkatkan kebiasaan makan sehat secara berkelanjutan, baik melalui penyediaan rekomendasi resep bergizi maupun fitur interaktif lainnya (Agarwal et al., 2024). Dengan integrasi teknologi modern, aplikasi seperti ini mampu memberikan pengalaman pengguna yang lebih personal dan komprehensif.

Penerapan framework pengembangan lintas platform seperti Flutter telah meningkatkan efisiensi dan performa pengembangan aplikasi katalog resep makanan. Flutter mendukung pengembangan aplikasi yang responsif, mudah dikembangkan, serta mendukung berbagai perangkat, sehingga mampu menjawab tantangan fragmentasi platform dalam dunia mobile programming (Algadrie & Sela, 2024; Chatterjee et al., 2021). Melalui pendekatan ini, aplikasi katalog resep dapat memberikan pengalaman antarmuka yang konsisten dan berkualitas tinggi bagi pengguna.

Lebih lanjut, fitur-fitur inovatif seperti pencarian resep berbasis kategori, penyimpanan resep favorit, serta integrasi dengan basis data cloud semakin menambah nilai guna aplikasi katalog resep makanan. Pengalaman pengguna pun semakin ditingkatkan dengan adanya personalisasi menu, saran nutrisi, serta kemampuan berbagi resep antar komunitas pengguna.

Penelitian juga menunjukkan bahwa keterlibatan fitur interaktif dan edukatif dalam aplikasi resep mampu meningkatkan keterampilan memasak dan gaya hidup sehat (Eicher-Miller et al., 2021; Samoggia & Riedel, 2020).

Akhirnya, kebutuhan akan aplikasi katalog resep makanan semakin relevan di tengah tantangan gaya hidup modern, urbanisasi, dan meningkatnya minat masyarakat terhadap kesehatan serta eksplorasi kuliner secara digital. Aplikasi yang dikembangkan berbasis pendekatan user-centered dan teknologi terkini dapat menjadi solusi praktis sekaligus inovatif untuk mendukung kebutuhan masyarakat. Dengan demikian, aplikasi katalog resep makanan tidak hanya berfungsi sebagai alat bantu memasak, namun juga sebagai media edukasi, pelestarian budaya kuliner, dan peningkatan kualitas hidup (Agarwal et al., 2024; Samoggia & Riedel, 2020).

1.2 Rumusan Masalah

1. Bagaimana merancang struktur antarmuka pengguna (user interface) yang sederhana dan intuitif untuk menampilkan katalog resep dalam format yang mudah dibaca dan dipahami oleh pengguna dengan latar belakang yang beragam?
2. Bagaimana mengorganisasi dan menyajikan informasi lengkap setiap resep mencakup nama makanan, gambar makanan, bahan-bahan, dan cara pembuatan dalam halaman detail yang terstruktur dan mudah diikuti?
3. Bagaimana mengimplementasikan sistem navigasi yang efektif sehingga pengguna dapat dengan mudah menjelajahi daftar resep dan mengakses informasi detail resep yang diinginkan?
4. Bagaimana mengelola data resep menggunakan model data yang tepat agar semua informasi resep dapat disimpan, dikelola, dan ditampilkan secara konsisten di seluruh aplikasi?
5. Bagaimana memastikan bahwa aplikasi dapat berfungsi optimal sebagai referensi sederhana bagi pengguna yang ingin memasak dengan menyediakan tata letak yang rapi, pembacaan yang lancar, dan aksesibilitas yang baik?

1.3 Tujuan

1. Merancang dan mengimplementasikan antarmuka pengguna (user interface) yang

sederhana, intuitif, dan user-friendly untuk menampilkan katalog resep sehingga pengguna dapat dengan mudah memahami informasi yang disajikan.

2. Mengembangkan halaman detail resep yang komprehensif dan terstruktur dengan menampilkan informasi lengkap mencakup nama makanan, gambar makanan, bahan-bahan, dan cara pembuatan secara jelas dan terorganisir.
3. Mengimplementasikan sistem navigasi yang efektif menggunakan widget Navigator dan button yang responsif sehingga pengguna dapat dengan lancar berpindah dari daftar resep ke halaman detail resep.
4. Merancang struktur model data (data model) yang tepat untuk mengelola informasi resep secara konsisten agar semua informasi dapat disimpan, dikelola, dan ditampilkan dengan akurat di seluruh aplikasi.
5. Memastikan bahwa aplikasi Catalog Resep Makanan dapat berfungsi sebagai referensi sederhana dan praktis bagi pengguna dengan menyediakan tata letak yang rapi, performa yang optimal, dan aksesibilitas yang baik.

BAB 2
PEMBAHASAN

2.1 Penjelasan Aplikasi “Catalog Resep Makanan”

Aplikasi ini dikembangkan untuk memberikan kemudahan kepada pengguna dalam mencari, memilih, dan mempelajari resep masakan secara digital melalui perangkat smartphone. Dengan menampilkan resep secara sistematis dan visual, pengguna dapat dengan mudah melihat nama makanan, gambar ilustrasi, daftar bahan, hingga tahapan pembuatan secara berurutan dan terstruktur. Setiap halaman pada aplikasi dirancang agar responsif dan mudah digunakan, sehingga memperkuat pengalaman belajar memasak langsung dari perangkat mobile.

Menggunakan framework Flutter dalam pengembangannya, aplikasi ini mendukung pengoperasian lintas platform baik di Android, iOS, maupun Web tanpa perlu penyesuaian berlebih pada kode sumber. Struktur proyek yang lengkap pada Flutter meliputi folder lib, android, ios, web, assets, dan pengelolaan dependencies melalui pubspec.yaml memastikan aplikasi dapat dikembangkan dan di-maintain dengan lebih efisien serta fleksibel untuk penambahan fitur di masa mendatang. Pendekatan ini juga memudahkan pembagian tugas saat pengembangan tim dan meningkatkan kualitas aplikasi dari segi performa maupun tampilan.

Penerapan desain antarmuka yang sederhana dan intuitif memampukan pengguna dari berbagai latar belakang usia dan keahlian untuk mengakses informasi resep tanpa hambatan. Fungsi pencarian, penampilan daftar menggunakan ListView atau GridView, serta navigasi antar halaman menggunakan fitur Navigator, semuanya terintegrasi dalam aplikasi ini untuk menyajikan akses informasi yang cepat dan praktis. Lokasi penyimpanan data resep pada file model dan data dummy memudahkan pengelolaan serta pemanggilan data saat pengguna membutuhkan detail resep tertentu.

Dengan adanya aplikasi ini, diharapkan pengguna tidak hanya memperoleh referensi memasak yang variatif, tetapi juga memperoleh pengalaman digital yang edukatif dan menyenangkan dalam eksplorasi resep-resep kuliner. Adopsi teknologi Flutter pada proyek ini juga dapat menjadi model pengembangan aplikasi katalog digital yang scalable, efisien, dan modern sesuai kebutuhan masyarakat digital saat ini.

2.2 Tujuan Dibuatnya Aplikasi

Aplikasi Catalog Resep Makanan dikembangkan dengan tujuan utama menyediakan katalog resep dalam format yang mudah dibaca dan dapat diakses kapan saja melalui perangkat mobile. Desain antarmuka aplikasi dirancang secara sederhana namun informatif, sehingga pengguna dari berbagai kalangan dapat dengan mudah memahami struktur navigasi dan menemukan resep yang diinginkan tanpa kebingungan. Format penyajian yang rapi dan terorganisir memastikan bahwa informasi resep dapat dikonsumsi dengan nyaman dan efisien.

Setiap resep dalam aplikasi ini ditampilkan dengan informasi yang lengkap dan komprehensif untuk mendukung pengguna dalam proses memasak. Informasi yang disajikan mencakup nama makanan yang jelas dan deskriptif, gambar makanan yang visual dan menarik untuk meningkatkan daya tarik dan motivasi pengguna, daftar bahan-bahan lengkap dengan takaran yang terukur untuk memastikan ketepatan resep, serta cara pembuatan yang diuraikan secara bertahap dan mudah diikuti. Dengan menyajikan semua elemen ini dalam satu platform terpadu, pengguna tidak perlu lagi mencari informasi dari berbagai sumber yang dapat membingungkan atau tidak konsisten.

Fungsi utama aplikasi adalah menjadi referensi sederhana dan praktis bagi pengguna yang ingin memasak, baik untuk pemula maupun yang sudah berpengalaman. Pengguna dapat dengan cepat menemukan resep yang sesuai dengan keinginan mereka, mempelajari bahan-bahan yang diperlukan, memahami langkah-langkah pembuatan dengan jelas, dan langsung menerapkannya di dapur. Dengan demikian, aplikasi ini mengurangi hambatan dalam proses belajar memasak dan membuat aktivitas memasak menjadi lebih terarah, efisien, dan menyenangkan.

Lebih lanjut, pengembangan aplikasi ini bertujuan untuk menunjukkan bahwa teknologi mobile dapat dimanfaatkan sebagai medium edukatif yang efektif dalam mendukung kegiatan sehari-hari masyarakat, khususnya dalam konteks kuliner dan memasak. Aplikasi Catalog Resep Makanan diharapkan dapat menjadi solusi digital yang tidak hanya memenuhi kebutuhan fungsional pengguna, tetapi juga memberikan pengalaman pengguna yang positif, intuitif, dan mudah diakses, sehingga dapat menjadi referensi favorit bagi siapa saja yang ingin mengeksplorasi dan mempelajari berbagai resep masakan.

2.3 Teknologi yang Digunakan

Aplikasi Catalog Resep Makanan dikembangkan dengan memanfaatkan berbagai teknologi dan tools modern yang mendukung pengembangan aplikasi mobile lintas platform. Pemilihan teknologi ini didasarkan pada kebutuhan untuk menciptakan aplikasi yang responsif, efisien, dan mudah dimaintain, sekaligus memastikan konsistensi pengalaman pengguna di berbagai perangkat.

- Framework

Flutter merupakan framework utama yang digunakan dalam pengembangan aplikasi ini. Flutter dipilih karena kemampuannya untuk membangun aplikasi lintas platform (Android, iOS, Web, dan Windows) dengan menggunakan satu basis kode yang sama, sehingga meningkatkan efisiensi pengembangan dan mengurangi duplikasi kode. Framework ini menyediakan widget-widget yang kaya dan responsif, memungkinkan tim pengembang untuk membuat antarmuka pengguna yang menarik dan intuitif tanpa perlu menyesuaikan kode secara signifikan untuk setiap platform. Selain itu, Flutter memiliki performa tinggi dengan hot reload feature yang memudahkan proses debugging dan iterasi pengembangan.

- Bahasa Pemrograman

Dart adalah bahasa pemrograman yang digunakan untuk mengembangkan seluruh logika aplikasi. Sebagai bahasa pemrograman resmi Flutter, Dart menyediakan sintaks yang modern, type-safe, dan mudah dipelajari oleh developer. Dart mendukung pemrograman berorientasi objek (OOP), functional programming, dan async/await patterns yang memudahkan pengelolaan operasi asynchronous dalam aplikasi.

- Struktur Proyek dan Organisasi File

Aplikasi ini mengikuti struktur proyek Flutter yang terorganisir dengan baik untuk memastikan maintainability dan scalability. Berikut adalah penjelasan setiap komponen utama:

1) Folder Assets/Images berisi semua gambar-gambar makanan yang digunakan sebagai visual representasi setiap resep. Gambar-gambar ini disimpan dalam format yang optimal

untuk memastikan loading yang cepat dan tampilan yang berkualitas tinggi di berbagai ukuran layar perangkat.

2) Data (recipe_data.dart) berfungsi sebagai sumber data dummy yang menyimpan koleksi resep lengkap. File ini berisi data resep yang dapat diakses oleh berbagai bagian aplikasi, memudahkan proses pengambilan dan pengelolaan informasi resep secara terpusat.
3) Models (recipe.dart) mendefinisikan struktur data model untuk resep makanan. File ini mengandung class Recipe yang merepresentasikan entitas resep dengan properti seperti nama makanan, gambar, bahan-bahan, dan cara pembuatan. Dengan mendefinisikan model yang jelas, aplikasi dapat mengelola data resep secara konsisten dan tipe-aman di seluruh aplikasi.
4) Providers (recipe_provider.dart) mengimplementasikan state management untuk mengelola data resep dan status aplikasi. Provider memudahkan sharing data antar widget tanpa perlu passing props secara manual, sehingga membuat arsitektur aplikasi lebih clean dan maintainable.
5) Screens berisi halaman-halaman utama aplikasi:
    - add_recipe_screen.dart adalah halaman untuk menambahkan resep baru (jika fitur ini diimplementasikan).
    - home_screen.dart merupakan halaman utama yang menampilkan daftar resep lengkap.
    - splash_screen.dart adalah halaman loading yang ditampilkan saat aplikasi pertama kali dibuka.
6) Widgets berisi komponen-komponen UI yang dapat digunakan kembali (reusable components):
    - Categori_chip.dart adalah widget chip untuk menampilkan kategori resep sebagai filter.
    - Recipe_card.dart menampilkan ringkasan resep dalam format card yang menarik.
    - Recipe_detail_sheet.dart adalah bottom sheet yang menampilkan detail lengkap resep.
    - Recipe_list_item.dart merepresentasikan item individual dalam daftar resep.

7) Main.dart adalah entry point aplikasi yang mengatur konfigurasi awal, navigasi, dan theme aplikasi. File ini juga mendeklarasikan MaterialApp dan mengatur root widget aplikasi.

2.4 Struktur Proyek Utama (Folder Penting)

- /lib

  Folder lib merupakan pusat logika dan kode utama aplikasi Flutter yang berisi semua file Dart untuk fungsionalitas aplikasi. Struktur dalam folder lib diorganisir secara modular dengan pemisahan yang jelas antara berbagai komponen aplikasi. File-file kunci yang terdapat di dalamnya mencakup main.dart sebagai entry point aplikasi, halaman-halaman screen (home_screen.dart, detail_page.dart, add_recipe_screen.dart, splash_screen.dart), model data (recipe_model.dart), sumber data (recipe_data.dart), dan state management (recipe_provider.dart). Selain itu, folder lib juga berisi berbagai widget reusable seperti categori_chip.dart untuk filter kategori, recipe_card.dart untuk menampilkan ringkasan resep, recipe_detail_sheet.dart untuk detail resep dalam bottom sheet, dan recipe_list_item.dart untuk item dalam daftar resep. Meskipun detail implementasi setiap file tidak ditampilkan secara eksplisit, folder lib adalah jantung dari aplikasi yang mengkoordinasikan seluruh logika bisnis, manajemen state, dan interaksi pengguna.

- /assets/images

  Folder assets/images berfungsi sebagai repository untuk menyimpan semua gambar resep makanan yang akan ditampilkan dalam aplikasi. Gambar-gambar ini disimpan dalam format file image standar (seperti .jpg atau .png) dan diorganisir dengan nama file yang deskriptif dan mudah dikenali. Contoh file gambar yang terdapat dalam folder ini antara lain rendang daging.jpg, chocolate pancake.jpg, dan sato ayam.jpg. Penyimpanan gambar secara lokal dalam folder assets memastikan aplikasi dapat menampilkan visual resep dengan cepat tanpa bergantung pada koneksi internet eksternal, sehingga meningkatkan performa dan user experience aplikasi.

- /pubspec.yaml

File pubspec.yaml merupakan file konfigurasi utama proyek Flutter yang mengatur dependency, asset, dan pengaturan proyek secara keseluruhan. File ini digunakan untuk mendeklarasikan library eksternal yang diperlukan aplikasi, mendaftarkan gambar dan asset ke dalam proyek, serta mengatur versi Flutter dan Dart yang digunakan. Dengan mendaftarkan folder assets dalam pubspec.yaml, semua gambar resep dapat diakses dengan mudah oleh aplikasi melalui path yang konsisten.

- /android, /ios, /web, /windows

  Folder-folder platform ini merupakan konfigurasi default Flutter untuk mendukung pengembangan lintas platform. Folder android berisi file konfigurasi Gradle, AndroidManifest.xml, dan resources spesifik Android. Folder ios berisi file Swift/Objective-C native code, Info.plist, dan konfigurasi iOS lainnya. Folder web berisi file HTML dan JavaScript untuk versi web aplikasi. Folder windows berisi C++ runner dan konfigurasi untuk platform Windows. Dengan struktur ini, aplikasi dapat di-build dan dijalankan di berbagai platform dengan konfigurasi yang sudah disiapkan oleh Flutter, memudahkan distribusi aplikasi ke multiple platform.

2.5 Cara Kerja Aplikasi

A. Halaman Utama (Home Page)

Halaman utama atau home page merupakan layar pertama yang dilihat pengguna setelah splash screen, berfungsi sebagai gateway utama untuk mengakses koleksi resep yang tersedia. Halaman ini menampilkan daftar resep dalam bentuk visual yang menarik dan mudah dinavigasi, baik menggunakan card list maupun grid layout yang disesuaikan dengan preferensi desain aplikasi. Setiap item resep ditampilkan dengan gambar resep berkualitas tinggi yang berfungsi sebagai visual hook untuk menarik perhatian pengguna, disertai dengan judul resep yang jelas dan deskriptif. Dengan menampilkan beberapa resep sekaligus dalam satu layar, pengguna dapat dengan cepat menjelajahi berbagai pilihan resep yang tersedia. Fungsionalitas interaktif memungkinkan pengguna untuk memilih atau mengetuk (tap) resep tertentu untuk melihat detail lengkapnya, menciptakan pengalaman navigasi yang intuitif dan user-friendly.

B. Halaman Detail Resep

Ketika pengguna memilih resep tertentu dari halaman utama, aplikasi akan menavigasi pengguna ke halaman detail resep yang menampilkan informasi komprehensif tentang resep yang dipilih. Gambar makanan ditampilkan di bagian atas halaman sebagai header yang menonjol, memberikan visual yang jelas dan menggugah selera pengguna sebelum membaca informasi detail lainnya. Informasi resep lengkap disajikan secara terstruktur dan mudah dipahami, mencakup nama makanan yang jelas, deskripsi singkat resep (jika tersedia) untuk memberikan konteks atau tips khusus, daftar bahan (ingredients) yang terurai dengan lengkap dan terukur untuk memastikan pengguna memiliki semua bahan yang diperlukan, serta langkah-langkah memasak (steps) yang dijabarkan secara bertahap dan mudah diikuti. Semua konten ini ditampilkan dalam bentuk scrollable view yang memungkinkan pengguna untuk dengan nyaman membaca seluruh informasi tanpa layar yang terlalu penuh, memberikan pengalaman membaca yang optimal di berbagai ukuran perangkat. Dengan desain ini, pengguna dapat memahami resep dengan jelas dan siap melaksanakan proses memasak dengan percaya diri.

2.6 Komponen Utama Aplikasi

- ListView / GridView

ListView dan GridView merupakan widget Flutter yang digunakan untuk menampilkan daftar resep dalam format yang scrollable dan responsif. ListView menampilkan item-item secara vertikal dalam satu kolom, sementara GridView menampilkan item-item dalam format grid dengan beberapa kolom, tergantung pada ukuran layar perangkat. Penggunaan widget ini memastikan bahwa daftar resep dapat ditampilkan dengan efisien bahkan ketika jumlah resep cukup banyak, karena kedua widget tersebut mengimplementasikan lazy loading untuk performa optimal.

- GestureDetector / InkWell

GestureDetector dan InkWell adalah widget Flutter yang digunakan untuk menangkap interaksi pengguna, khususnya klik atau tap pada item resep dalam daftar. GestureDetector memberikan

kontrol lebih fleksibel untuk mendeteksi berbagai jenis gesture, sementara InkWell memberikan visual feedback berupa efek ripple yang intuitif saat pengguna menyentuh item. Dengan menggunakan salah satu dari widget ini, aplikasi dapat merespons tindakan pengguna untuk membuka halaman detail resep yang dipilih.

- Navigator.push

Navigator.push merupakan metode Flutter untuk berpindah antar halaman (routing) dalam aplikasi. Ketika pengguna mengetuk item resep di halaman daftar, Navigator.push akan mengarahkan aplikasi untuk menampilkan halaman detail resep dengan mengirimkan data resep yang dipilih sebagai parameter. Dengan menggunakan Navigator, pengguna juga dapat dengan mudah kembali ke halaman daftar melalui tombol back yang disediakan secara default oleh Flutter.

- Model Data (recipe_model.dart)

Model data merupakan struktur fundamental dalam aplikasi untuk merepresentasikan objek resep. Berikut adalah struktur dasar class Recipe yang digunakan:

```
dart
class Recipe {
  String title;
  String image;
  List<String> ingredients;
  List<String> steps;
}
```

Class Recipe di atas mendefinisikan empat properti utama yang merepresentasikan informasi lengkap setiap resep. Properti `title` menyimpan nama makanan, `image` menyimpan path atau URL gambar resep, `ingredients` menyimpan daftar bahan-bahan dalam bentuk List of String, dan `steps` menyimpan langkah-langkah pembuatan juga dalam bentuk List of String. Dengan mendefinisikan model ini, aplikasi dapat mengelola data resep secara konsisten, tipe-aman, dan mudah untuk diakses oleh berbagai bagian aplikasi. Model ini juga memudahkan

proses serialisasi dan deserialisasi data jika aplikasi dikembangkan untuk mengintegrasikan dengan API eksternal atau database cloud di masa depan.

2.7 Kelebihan Aplikasi

Aplikasi Catalog Resep Makanan memiliki beberapa kelebihan yang membuatnya menjadi solusi yang efektif dan bermanfaat baik dari perspektif pengguna maupun dari aspek pengembangan aplikasi. Pertama, user interface (UI) aplikasi dirancang dengan sederhana namun elegan, memastikan bahwa pengguna dari berbagai latar belakang usia dan keahlian teknis dapat dengan mudah memahami dan menggunakan aplikasi tanpa memerlukan tutorial yang rumit. Desain UI yang intuitif mengurangi kurva pembelajaran pengguna dan meningkatkan kepuasan dalam menggunakan aplikasi.

Kedua, struktur proyek Flutter pada aplikasi ini sudah lengkap dengan semua komponen yang diperlukan mulai dari folder lib dengan kode logika aplikasi, folder assets untuk gambar, hingga konfigurasi platform-specific untuk Android, iOS, Web, dan Windows. Kelengkapan struktur ini memungkinkan aplikasi untuk dijalankan langsung tanpa perlu setup atau konfigurasi tambahan yang kompleks, sehingga pengguna dapat langsung melihat hasilnya dalam waktu singkat.

Ketiga, aplikasi ini memiliki potensi besar untuk dikembangkan menjadi aplikasi katalog makanan profesional yang lebih canggih. Dengan fondasi kode yang rapi dan terstruktur, fitur-fitur tambahan seperti sistem pencarian resep berbasis kategori, penyimpanan resep favorit, rating dan review resep, integrasi dengan database cloud, atau bahkan fitur kecerdasan buatan untuk rekomendasi resep dapat ditambahkan dengan relatif mudah tanpa perlu merombak struktur dasar aplikasi.

2.8 Kemungkinan Fitur Tambahan (Jika ingin dikembangkan)

Aplikasi Catalog Resep Makanan memiliki potensi pengembangan yang luas dengan berbagai fitur tambahan yang dapat meningkatkan fungsionalitas dan user experience. Berikut adalah beberapa fitur yang dapat dikembangkan di masa depan:

- **Fitur Pencarian Resep** merupakan fitur yang sangat berguna untuk memudahkan pengguna menemukan resep spesifik tanpa harus scroll melalui seluruh daftar resep. Dengan menambahkan search bar di halaman utama, pengguna dapat mengetikkan nama makanan, bahan, atau kategori resep untuk secara cepat menemukan resep yang dicari. Implementasi fitur ini dapat menggunakan algoritma pencarian sederhana berbasis string matching atau filter pada data lokal.

- **Fitur Favorit (Save Recipe)** memungkinkan pengguna untuk menyimpan resep-resep pilihan mereka untuk akses yang lebih mudah di masa depan. Fitur ini dapat diimplementasikan dengan menambahkan tombol hati atau bintang pada setiap resep, serta membuat halaman tersendiri yang menampilkan daftar resep favorit pengguna. Data favorit dapat disimpan secara lokal menggunakan shared preferences atau database lokal.

- **Login Pengguna** menambahkan personalisasi pada aplikasi dengan memungkinkan setiap pengguna memiliki profil dan preferensi mereka sendiri. Dengan sistem login, data favorit, history resep yang dibaca, atau preferensi pengguna lainnya dapat disimpan dan disinkronkan antar perangkat jika dikombinasikan dengan backend server.

- **Database (Firebase / SQLite)** merupakan infrastruktur penting untuk menyimpan data resep secara persisten dan aman. Firebase menawarkan solusi cloud database yang scalable dengan fitur real-time synchronization, sementara SQLite merupakan pilihan untuk database lokal yang lebih ringan dan tidak memerlukan koneksi internet. Integrasi database ini memungkinkan aplikasi untuk mengelola data resep dalam jumlah yang lebih besar dan menyediakan fitur backup data pengguna.

- **Dark Mode** merupakan fitur modern yang semakin diminati pengguna, terutama mereka yang menggunakan perangkat pada kondisi cahaya rendah. Dengan menambahkan dark mode, aplikasi tidak hanya lebih nyaman digunakan di malam hari, tetapi juga dapat menghemat baterai pada perangkat dengan layar OLED. Flutter menyediakan built-in support untuk theme switching yang memudahkan implementasi fitur ini.

Dengan menambahkan fitur-fitur tambahan ini, aplikasi Catalog Resep Makanan dapat berkembang menjadi platform yang lebih powerful dan kompetitif dalam ekosistem aplikasi

mobile modern, tetap mempertahankan kesederhanaan desain dan kemudahan penggunaan yang menjadi fondasi kekuatan aplikasi.

2.9 Penjelasan Detail Struktur Folder /lib

Meskipun isi file individual tidak ditampilkan secara detail satu per satu dalam laporan ini, struktur proyek Flutter secara umum mengikuti pola organisasi yang terstandarisasi dan best practice dalam industri pengembangan aplikasi mobile. Struktur ini dirancang untuk memastikan maintainability, scalability, dan kemudahan kolaborasi antar developer.

```
lib/
├── main.dart
├── pages/
│   ├── home_page.dart
│   ├── detail_page.dart
├── models/
│   ├── recipe_model.dart
└── data/
    ├── recipe_data.dart
```

Berikut penjelasan fungsi masing-masing

2.9.1 File main.dart – Entry Point Aplikasi

File main.dart merupakan titik pertama dan paling fundamental dalam aplikasi Flutter, berfungsi sebagai entry point yang dijalankan ketika aplikasi diluncurkan. File ini mengatur konfigurasi awal dan struktur dasar aplikasi yang akan menentukan tampilan dan behavior aplikasi secara keseluruhan.

Fungsi main() adalah fungsi yang pertama kali dieksekusi saat aplikasi dijalankan. Fungsi ini bertugas memanggil fungsi runApp() yang menerima widget utama aplikasi sebagai parameter:

```
dart
void main() {
  runApp(MyApp());
}
```

Fungsi runApp() ini menginformasikan framework Flutter untuk membangun dan merender widget yang diteruskan sebagai root dari widget tree aplikasi.

Widget Root → MaterialApp merupakan widget utama yang membungkus seluruh aplikasi dan menyediakan konfigurasi global untuk aplikasi. Widget MaterialApp menerapkan Material Design language dari Google dan menyediakan berbagai fitur penting untuk aplikasi. Beberapa konfigurasi penting dalam MaterialApp mencakup judul aplikasi (title) yang ditampilkan di system-level, tema (theme) yang mengatur warna, tipografi, dan styling global aplikasi, router dan navigasi yang menentukan halaman awal dan alur navigasi, serta berbagai pengaturan lainnya seperti localization dan responsive behavior.

Contoh implementasi sederhana MaterialApp:

dart
```
MaterialApp(
  debugShowCheckedModeBanner: false,
  home: HomePage(),
);
```

Dalam contoh di atas, parameter debugShowCheckedModeBanner: false digunakan untuk menyembunyikan banner debug yang biasanya muncul di sudut aplikasi saat mode debug. Parameter home: HomePage() menentukan halaman awal yang akan ditampilkan saat aplikasi pertama kali dibuka. Dengan struktur ini, aplikasi memiliki fondasi yang kuat dan siap untuk menampilkan halaman utama kepada pengguna.

2.9.2 File home_page.dart – Halaman Daftar Resep

File home_page.dart merupakan halaman pertama yang dilihat oleh pengguna ketika membuka aplikasi setelah splash screen. Halaman ini berfungsi sebagai dashboard utama yang menampilkan koleksi lengkap resep yang tersedia dalam aplikasi. Desain dan fungsionalitas halaman ini dirancang untuk memberikan pengalaman pengguna yang intuitif dan memudahkan pengguna dalam menjelajahi berbagai pilihan resep.

AppBar (Judul Aplikasi) merupakan komponen header di bagian atas halaman yang berisi informasi identitas aplikasi. AppBar memberikan visual hierarchy yang jelas dan memberitahu pengguna bahwa mereka berada di halaman mana. Contoh implementasi AppBar:

dart
```
AppBar(title: Text('Recipe Catalog'))
```

AppBar ini menampilkan teks "Recipe Catalog" sebagai judul aplikasi, dan biasanya juga dapat dilengkapi dengan icon atau tombol tambahan untuk navigasi atau aksi lainnya.

ListView / GridView Resep merupakan widget utama yang menampilkan daftar resep dalam format scrollable. Penggunaan ListView atau GridView memungkinkan aplikasi untuk menampilkan sejumlah besar item resep dengan performa optimal, terutama ketika jumlah resep bertambah di masa depan. Data resep biasanya diambil dari file recipe_data.dart yang berisi koleksi resep lengkap.

Contoh implementasi menggunakan ListView.builder():

```
dart
ListView.builder(
  itemCount: recipeList.length,
  itemBuilder: (context, index) {
    final recipe = recipeList[index];
    return RecipeCard(recipe);
  },
);
```

Dalam implementasi ini, itemCount menentukan jumlah item yang akan ditampilkan berdasarkan panjang daftar resep. Parameter itemBuilder adalah callback function yang dipanggil untuk setiap item, menerima context dan index sebagai parameter. Untuk setiap index, widget RecipeCard dipanggil dengan objek resep yang sesuai untuk menampilkan card individual dengan informasi resep ringkas. Pendekatan ini menggunakan lazy loading, artinya hanya widget yang terlihat di layar yang akan dirender, meningkatkan efisiensi performa aplikasi.

Alternatif lain adalah menggunakan GridView.count() untuk menampilkan resep dalam format grid dengan multiple columns, memberikan visual yang berbeda namun tetap mempertahankan fungsionalitas yang sama. Dengan struktur halaman utama ini, pengguna dapat dengan mudah melihat berbagai pilihan resep dan memilih resep mana yang ingin dilihat detail-nya.

2.9.3 File detail_page.dart – Halaman Detail Resep

File detail_page.dart merupakan halaman yang ditampilkan ketika pengguna memilih salah satu resep dari daftar di halaman utama. Halaman ini menampilkan informasi lengkap dan komprehensif tentang resep yang dipilih, dirancang untuk memberikan semua detail yang diperlukan pengguna untuk memasak. Halaman detail resep biasanya berisi berbagai komponen yang tersusun secara vertikal dalam format yang mudah dibaca.

1) **Gambar Resep** ditampilkan di bagian atas halaman sebagai elemen visual utama yang menarik perhatian pengguna. Gambar ini dimuat dari file yang tersimpan di folder assets/images menggunakan widget Image.asset():

dart
```
Image.asset(recipe.image)
```

Widget ini menampilkan gambar resep berkualitas tinggi yang telah disimpan dalam proyek aplikasi, memberikan visual representation yang jelas tentang hasil akhir masakan.

2) **Nama Makanan** ditampilkan sebagai judul besar di bawah gambar menggunakan widget Text():

dart
```
Text(recipe.title)
```

Nama makanan ini berfungsi sebagai identifikasi jelas dari resep yang sedang dilihat pengguna.

3) **Bahan-bahan (Ingredients)** ditampilkan sebagai list bullet untuk memudahkan pengguna melihat semua bahan yang diperlukan. Implementasinya menggunakan Column dengan map function untuk mengubah setiap bahan menjadi text dengan bullet point:

dart
```
Column(
  children: recipe.ingredients
```

```
    .map((ing) => Text("• $ing"))
    .toList(),
)
```

Dengan format ini, setiap bahan ditampilkan dalam baris terpisah dengan bullet point (-) di depannya, membuat daftar bahan mudah dibaca dan dipahami.

4) **Langkah-langkah Memasak** ditampilkan dengan nomor urut untuk memandu pengguna melalui proses memasak secara sistematis. Implementasinya menggunakan List.generate() untuk membuat teks berformat dengan nomor:

```
dart
Column(
  children: List.generate(recipe.steps.length, (i) {
    return Text("${i+1}. ${recipe.steps[i]}");
  }),
)
```

Setiap langkah ditampilkan dengan nomor urut (dimulai dari 1) diikuti dengan deskripsi langkah, sehingga pengguna dapat mengikuti proses memasak dengan jelas dan terstruktur.

5) **Scrollable View** menggunakan SingleChildScrollView untuk membungkus seluruh konten halaman detail, karena informasi resep yang lengkap sering kali melebihi tinggi layar perangkat:

```
dart
SingleChildScrollView(
  child: Column(...)
)
```

Widget SingleChildScrollView memungkinkan pengguna untuk scroll ke bawah untuk melihat semua informasi resep, termasuk gambar, judul, bahan-bahan, dan langkah-langkah memasak. Dengan menggunakan scrollable view, halaman tetap terorganisir dan mudah dibaca tanpa informasi yang terpotong atau tersembunyi.

2.9.4 File recipe_model.dart – Struktur Data Resep

Berisi model class yang merepresentasikan objek resep. Contoh:

```
class Recipe {
  final String title;
  final String image;
  final List<String> ingredients;
  final List<String> steps;
  Recipe({
    required this.title,
    required this.image,
    required this.ingredients,
    required this.steps,
  });
}
```

Fungsi model ini:

- Merapikan data resep
- Memudahkan pengiriman data antar halaman
- Memudahkan mapping ke widget UI

2.9.5 File recipe_data.dart – Data Dummy Resep

File ini berisi list resep statis (dummy data). Contoh isi:

```
List<Recipe> recipeList = [
  Recipe(
    title: "Nasi Goreng",
    image: "assets/images/nasi_goreng.jpg",
    ingredients: [
      "2 porsi nasi putih",
```

```
      "2 siung bawang putih",
      "1 butir telur",
    ],
    steps: [
      "Panaskan minyak.",
      "Tumis bawang putih hingga harum.",
      "Masukkan telur dan orak-arik.",
      "Tambahkan nasi, bumbui, dan aduk rata.",
    ],
  ),
];
```

Data ini kemudian digunakan di home_page.dart untuk menampilkan daftar resep.

2.9.6 file recipe.dart

```dart
class Recipe {

 final String id;

 final String name;

 final String category;

 final String image;

 final String cookTime;

 final String difficulty;

 final String description;

 final List<String> ingredients;
```

```
Recipe({
  required this.id,
  required this.name,
  required this.category,
  required this.image,
  required this.cookTime,
  required this.difficulty,
  required this.description,
  required this.ingredients,
});


// Copy with method untuk update recipe
Recipe copyWith({
  String? id,
  String? name,
  String? category,
  String? image,
  String? cookTime,
  String? difficulty,
  String? description,
```

```
  List<String>? ingredients,
 }) {
  return Recipe(
    id: id ?? this.id,
    name: name ?? this.name,
    category: category ?? this.category,
    image: image ?? this.image,
    cookTime: cookTime ?? this.cookTime,
    difficulty: difficulty ?? this.difficulty,
    description: description ?? this.description,
    ingredients: ingredients ?? this.ingredients,
   );
  }
}
```

File **recipe.dart** mendefinisikan **Model Data** (atau *blueprint*) untuk setiap resep dalam aplikasi. Dalam pengembangan aplikasi modern, penggunaan Model Data sangat penting untuk menjaga konsistensi dan struktur data.

1. Deklarasi dan Properti (Fields)

```
class Recipe {
  final String id;
```

```
  final String name;

  // ... properti lainnya

  final List<String> ingredients;

// ...

}
```

- **Tujuan:** Blok ini mendefinisikan **atribut-atribut** (data) apa saja yang dimiliki oleh sebuah objek resep.
- **Kata Kunci final:** Semua properti dideklarasikan dengan final. Ini berarti objek Recipe bersifat **immutable** (tidak dapat diubah) setelah dibuat. Jika Anda ingin mengubah satu properti (misalnya, nama), Anda harus membuat objek Recipe baru (menggunakan metode copyWith). Ini adalah praktik terbaik (best practice) dalam Flutter, terutama saat menggunakan *state management* seperti Riverpod.

Properti	Tipe Data	Fungsi
id	String	Kode unik (Identifier) resep. Penting untuk operasi seperti menandai favorit.
name	String	Judul resep.
category	String	Jenis masakan (e.g., 'Indonesian', 'Western').
image	String	**URL gambar** (atau emoji/path lokal) resep. Digunakan oleh widget Image.network.

cookTime	String	Durasi memasak.
difficulty	String	Tingkat kesulitan ('Mudah', 'Sulit').
description	String	Penjelasan detail resep.
ingredients	List<String>	Daftar bahan-bahan yang diwakili sebagai kumpulan string.

2. Constructor (Pembangun Objek)

```
Recipe({

  required this.id,

  required this.name,

  // ... semua properti lainnya

});
```

- **Tujuan:** Blok ini adalah fungsi khusus yang dipanggil untuk **membuat** objek Recipe baru.
- **Parameter Bernama ({ }):** Parameter diletakkan di dalam kurung kurawal, yang memungkinkan Anda membuat objek dengan urutan parameter yang fleksibel (e.g., Recipe(name: '...', id: '...')).
- **Kata Kunci required:** Ini memastikan bahwa saat Anda membuat objek Recipe, Anda **wajib** memberikan nilai untuk semua properti yang ada, sehingga tidak ada data penting yang terlewat.

3. Metode copyWith (Metode Kloning)

```
// Copy with method untuk update recipe

Recipe copyWith({

  String? id, // Parameter bersifat opsional (nullable)

  String? name,

  // ...

}) {

  return Recipe(

    id: id ?? this.id,

    name: name ?? this.name,

    // ...

  );

}
```

- **Tujuan:** Ini adalah metode utilitas standar untuk menangani objek yang *immutable* (final). Fungsinya untuk **membuat salinan baru** dari objek Recipe saat ini, tetapi memungkinkan Anda untuk mengganti nilai dari satu atau beberapa properti saja.
- **Parameter Opsional (String?):** Semua parameter dalam copyWith bersifat *nullable* (String?), artinya Anda boleh tidak memasukkan nilainya.
- **Operator *Null Check* (??):** Ini adalah bagian terpenting. Logika id: id ?? this.id berarti:
    - Jika Anda menyediakan nilai id baru (id bukan null), gunakan nilai tersebut.
    - **JIKA TIDAK** (yaitu, id adalah null, dilambangkan oleh ??), gunakan nilai id yang sudah ada pada objek lama (this.id).

2.9.7 recipe_providers.dart

```
import 'package:flutter_riverpod/flutter_riverpod.dart';

import '../models/recipe.dart';

import '../data/recipe_data.dart';

// State untuk menyimpan resep

final recipesProvider = StateProvider<List<Recipe>>((ref) => initialRecipes);

// State untuk kategori yang dipilih

final selectedCategoryProvider = StateProvider<String>((ref) => 'Semua');

// State untuk resep favorit

final favoritesProvider = StateProvider<Set<String>>((ref) => {});

// State untuk query pencarian

final searchQueryProvider = StateProvider<String>((ref) => '');

// State untuk view mode (grid atau list)

final viewModeProvider = StateProvider<ViewMode>((ref) => ViewMode.grid);

// State untuk filter favorit

final showOnlyFavoritesProvider = StateProvider<bool>((ref) => false);

// Enum untuk view mode

enum ViewMode { grid, list }

// Provider computed untuk resep yang difilter

final filteredRecipesProvider = Provider<List<Recipe>>((ref) {

  final recipes = ref.watch(recipesProvider);
```

```
  final category = ref.watch(selectedCategoryProvider);
  final searchQuery = ref.watch(searchQueryProvider).toLowerCase();
  final favorites = ref.watch(favoritesProvider);
  final showOnlyFavorites = ref.watch(showOnlyFavoritesProvider);
  var filteredRecipes = recipes;
  // Filter berdasarkan kategori
  if (category != 'Semua') {
filteredRecipes = filteredRecipes.where((r) => r.category == category).toList();
  }
  // Filter berdasarkan pencarian
  if (searchQuery.isNotEmpty) {
    filteredRecipes = filteredRecipes.where((r) =>
      r.name.toLowerCase().contains(searchQuery) ||
      r.description.toLowerCase().contains(searchQuery)
    ).toList();
  }
  // Filter hanya favorit
        if (showOnlyFavorites) { filteredRecipes = filteredRecipes.where((r) =>
favorites.contains(r.id)).toList();
  }
  return filteredRecipes;
});
```

File ini berisi kumpulan *Provider* yang berfungsi sebagai "tempat penyimpanan" dan "logika bisnis" utama aplikasi.

1. Provider Penyimpanan Data Utama (StateProviders)

StateProvider digunakan untuk menyimpan data yang bisa berubah (mutabel).

Provider	**Tipe Data yang Disimpan**	**Fungsi**
recipesProvider	List<Recipe>	Menyimpan **seluruh daftar resep** di aplikasi. Diinisialisasi dengan data dari initialRecipes (dari file recipe_data.dart). Provider ini akan diperbarui ketika pengguna **menambahkan resep baru**.
selectedCategoryProvider	String	Menyimpan kategori yang sedang **dipilih** oleh pengguna (misalnya: 'Indonesian', 'Western'). Nilai awal adalah 'Semua'.
favoritesProvider	Set<String>	Menyimpan kumpulan **ID resep** yang telah ditandai sebagai favorit. Menggunakan Set menjamin setiap ID resep hanya ada satu kali.

searchQueryProvider	String	Menyimpan **teks** yang dimasukkan pengguna ke kolom pencarian (search bar).
viewModeProvider	ViewMode (Enum)	Menyimpan preferensi tampilan resep: **ViewMode.grid** (kotak) atau **ViewMode.list** (daftar).
showOnlyFavoritesProvider	bool	Menyimpan status *toggle switch* (saklar) yang menunjukkan apakah aplikasi harus **hanya menampilkan resep favorit** (true) atau semua resep (false).

2. Enum ViewMode

Dart

enum ViewMode { grid, list }

- **Tujuan:** Mendefinisikan dua opsi yang mungkin untuk tampilan resep. Ini membuat kode lebih jelas dan aman (*type-safe*) daripada hanya menggunakan String atau angka untuk mewakili mode tampilan.

3. Provider Logika Bisnis (Computed Provider)

Ini adalah *Provider* terpenting yang berisi logika untuk menggabungkan semua *state* di atas dan menghasilkan daftar resep yang sudah disaring.

filteredRecipesProvider

Dart

```
final filteredRecipesProvider = Provider<List<Recipe>>((ref) {
  // 1. Ambil semua state yang relevan
  final recipes = ref.watch(recipesProvider);
  final category = ref.watch(selectedCategoryProvider);
  // ... state lainnya
  var filteredRecipes = recipes;
  // 2. Logika Filtering (Berurutan)
  // ...
  return filteredRecipes; // Mengembalikan hasil akhir
});
```

- **Tujuan:** Provider ini bertindak sebagai **"mesin filter"**. Ia tidak menyimpan data mentah, melainkan **"mendengarkan" (ref.watch)** perubahan dari semua *state* di atas. Kapan pun recipesProvider, selectedCategoryProvider, searchQueryProvider, atau showOnlyFavoritesProvider berubah, kode di dalam filteredRecipesProvider akan dijalankan ulang (re-run) secara otomatis, dan daftar resep yang ditampilkan di RecipeHomePage akan diperbarui.

Detail Logika Filtering

Logika di dalamnya menerapkan filter secara berurutan pada daftar resep:

Filter Kategori:

Dart

```
if (category != 'Semua') {
  filteredRecipes = filteredRecipes.where((r) => r.category == category).toList();
```

}

1. Jika kategori yang dipilih bukan 'Semua', daftar resep disaring hanya menampilkan resep yang memiliki r.category yang sesuai.

Filter Pencarian:

Dart

```
if (searchQuery.isNotEmpty) {
  filteredRecipes = filteredRecipes.where((r) =>
    r.name.toLowerCase().contains(searchQuery) ||
    r.description.toLowerCase().contains(searchQuery)
  ).toList();
}
```

2. Jika ada teks pencarian, daftar disaring untuk resep yang namanya (r.name) atau deskripsinya (r.description) mengandung teks pencarian tersebut (perbandingan dilakukan dalam huruf kecil/lowercase agar pencarian tidak sensitif huruf besar/kecil).

Filter Favorit:

Dart

```
if (showOnlyFavorites) {
  filteredRecipes = filimport 'package:flutter/material.dart';
```

```
import 'package:flutter_riverpod/flutter_riverpod.dart';
import '../models/recipe.dart';
import '../providers/recipe_providers.dart';
class AddRecipeDialog extends ConsumerStatefulWidget {
final Recipe? recipe;const AddRecipeDialog({super.key, this.recipe});
```

```dart
 @override
 ConsumerState<AddRecipeDialog> createState() => _AddRecipeDialogState();
class _AddRecipeDialogState extends ConsumerState<AddRecipeDialog> {
  final _formKey = GlobalKey<FormState>();
  final _nameController = TextEditingController();
  final _cookTimeController = TextEditingController();
  final _descriptionController = TextEditingController();
  final _ingredientsController = TextEditingController();
  String _selectedCategory = 'Indonesian';
  String _selectedDifficulty = 'Mudah';
  String _selectedEmoji = '🍳';
  final List<String> _categories = ['Indonesian', 'Western', 'Dessert'];
  final List<String> _difficulties = ['Mudah', 'Sedang', 'Sulit'];
  final List<String> _emojis = [
    '🍳',
    '🍝',
    '🍖',
    '🥞',
    '🍲',
    '🥗',
    '🍰',
    '🌮',
    '🍔',
```

'🍕',

'🍜',

'🍱',

'🥘',

'🍛',

'🍣',

'🥟',

'🌮',

'🌯',

'🥪',

'🍩',

'🧁',

'🍪',

'🥧',

'🍦',

'🥓',

'🍗',

'🍤',

'🦐',

'🦞',

'🍢',

'🍡',

'🥮',

```
  ];

  static const Color primaryDark = Color(0xFFE65100);

  static const Color primaryMain = Color(0xFFFF6F00);

  @override

  void initState() {

super.initState();

if (widget.recipe != null) {

_nameController.text = widget.recipe!.name

final      cookTimeParts      =      widget.recipe!.cookTime.split('      ')
_cookTimeController.text        =        cookTimeParts.isNotEmpty        &&
cookTimeParts.first.isNotEmpty

 ? cookTimeParts.first

 : '';



      _descriptionController.text = widget.recipe!.description;

      _ingredientsController.text = widget.recipe!.ingredients.join(', ');

      _selectedCategory = widget.recipe!.category;

      _selectedDifficulty = widget.recipe!.difficulty;

      _selectedEmoji = widget.recipe!.image;

    }

  }



  @override

  void dispose() {
```

```
    _nameController.dispose();
    _cookTimeController.dispose();
    _descriptionController.dispose();
    _ingredientsController.dispose();
    super.dispose();
  }


  void _saveRecipe() {
    if (_formKey.currentState!.validate()) {
      final recipes = ref.read(recipesProvider);


      final ingredientsList = _ingredientsController.text
          .split(RegExp(r'[,\n]'))
          .map((e) => e.trim())
          .where((e) => e.isNotEmpty)
          .toList();


      // Logika Baru untuk Waktu Memasak
      // Pastikan input adalah angka, lalu tambahkan ' menit'
      final rawCookTime = _cookTimeController.text.trim();
      final cookTime = rawCookTime.isNotEmpty
          ? '$rawCookTime menit'
          : '0 menit'; // Pastikan ada nilai default
```

```
      if (widget.recipe != null) {
        // Edit mode
        final updatedRecipe = widget.recipe!.copyWith(
          name: _nameController.text,
          category: _selectedCategory,
          image: _selectedEmoji,
           cookTime: cookTime, // Menggunakan variabel cookTime yang sudah
diformat
          difficulty: _selectedDifficulty,
          description: _descriptionController.text,
          ingredients: ingredientsList,
        );

        final updatedRecipes = recipes.map((r) {
          return r.id == updatedRecipe.id ? updatedRecipe : r;
        }).toList();

        ref.read(recipesProvider.notifier).state = updatedRecipes;

        Navigator.pop(context);

        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
```

```
            content: const Row(
              children: [
                Icon(Icons.check_circle, color: Colors.white),
                SizedBox(width: 8),
                Text('Resep berhasil diperbarui!'),
              ],
            ),
            backgroundColor: Colors.green,
            duration: const Duration(seconds: 2),
            behavior: SnackBarBehavior.floating,
          ),
        );
      } else {
        // Add mode
        // Asumsi Recipe id dihitung berdasarkan panjang list. Ini mungkin tidak ideal untuk aplikasi nyata,
        // tapi sesuai dengan implementasi Anda.
        final newId = (recipes.length + 1).toString();

        final newRecipe = Recipe(
          id: newId,
          name: _nameController.text,
          category: _selectedCategory,
          image: _selectedEmoji,
```

```
          cookTime: cookTime, // Menggunakan variabel cookTime yang sudah
diformat
          difficulty: _selectedDifficulty,
          description: _descriptionController.text,
          ingredients: ingredientsList,
        );

                ref.read(recipesProvider.notifier).state  =  [...recipes,
newRecipe];

        Navigator.pop(context);

        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: const Row(
              children: [
                Icon(Icons.check_circle, color: Colors.white),
                SizedBox(width: 8),
                Text('Resep berhasil ditambahkan!'),
              ],
            ),
            backgroundColor: Colors.green,
            duration: const Duration(seconds: 2),
            behavior: SnackBarBehavior.floating,
```

```
          ),
        );
      }
    }
}


// --- Widget Kustom untuk Segmented Chip Button ---
Widget _buildChipSelector({
  required String label,
  required List<String> options,
  required String selectedValue,
  required ValueSetter<String> onSelected,
}) {
  return Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        label,
        style: const TextStyle(
          fontSize: 14,
          fontWeight: FontWeight.bold,
          color: Colors.black87,
        ),
```

```
        ),
        const SizedBox(height: 8),
        Wrap(
          spacing: 8.0,
          children: options.map((item) {
            final isSelected = item == selectedValue;
            return ChoiceChip(
              label: Text(item),
              selected: isSelected,
              selectedColor: primaryMain.withOpacity(0.1),
              onSelected: (selected) {
                if (selected) onSelected(item);
              },
              backgroundColor: Colors.grey[100],
              labelStyle: TextStyle(
                color: isSelected ? primaryDark : Colors.black87,
                                fontWeight: isSelected ? FontWeight.bold :
FontWeight.normal,
              ),
              side: BorderSide(
                color: isSelected ? primaryDark : Colors.grey.shade300,
                width: 1.2,
              ),
              shape: RoundedRectangleBorder(
```

```
                borderRadius: BorderRadius.circular(20),
              ),
            );
          }).toList(),
        ),
      ],
    );
  }


  @override
  Widget build(BuildContext context) {
    final isEditMode = widget.recipe != null;


    return Dialog(
      shape: RoundedRectangleBorder(
        borderRadius: BorderRadius.circular(20),
      ),
      child: Container(
        constraints: const BoxConstraints(
          maxWidth: 600,
          maxHeight: 750,
        ),
        child: Column(
```

```
        children: [
          // Header
          Container(
            padding: const EdgeInsets.all(20),
            decoration: const BoxDecoration(
              color: primaryDark,
              borderRadius: BorderRadius.vertical(
                top: Radius.circular(20),
              ),
            ),
            child: Row(
              children: [
                Icon(
                  isEditMode ? Icons.edit : Icons.add_circle,
                  color: Colors.white,
                  size: 28,
                ),
                const SizedBox(width: 12),
                Text(
                  isEditMode ? 'Edit Resep' : 'Tambah Resep',
                  style: const TextStyle(
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
```

```
                color: Colors.white,
              ),
            ),
            const Spacer(),
            IconButton(
              icon: const Icon(Icons.close, color: Colors.white),
              onPressed: () => Navigator.pop(context),
            ),
          ],
        ),
      ),

      // Form Content
      Expanded(
        child: Form(
          key: _formKey,
          child: ListView(
            padding: const EdgeInsets.all(20),
            children: [
              // Emoji Selector
              const Text(
                'Pilih Icon Resep',
                style: TextStyle(
```

```
                        fontSize: 14,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                    const SizedBox(height: 8),

                    Container(
                      height: 120,
                      decoration: BoxDecoration(
                        border: Border.all(color: Colors.grey[300]!),
                        borderRadius: BorderRadius.circular(12),
                        color: Colors.white, // Latar belakang putih
                      ),
                      child: GridView.builder(
                        padding: const EdgeInsets.all(8),
                        scrollDirection: Axis.horizontal,
                        gridDelegate:
                                                                const
SliverGridDelegateWithFixedCrossAxisCount(
                          crossAxisCount: 3,
                          childAspectRatio: 1,
                          crossAxisSpacing: 4,
                          mainAxisSpacing: 4,
                        ),
```

```
                    itemCount: _emojis.length,
                    itemBuilder: (context, index) {
                      final emoji = _emojis[index];
                      final isSelected = emoji == _selectedEmoji;
                      return GestureDetector(
                        onTap: () {
                          setState(() {
                            _selectedEmoji = emoji;
                          });
                        },
                        child: AnimatedContainer(
                          duration: const Duration(milliseconds: 200),
                          decoration: BoxDecoration(
                            color: isSelected
                                ? primaryMain.withOpacity(0.2)
                                : Colors.transparent,
                            borderRadius: BorderRadius.circular(8),
                            border: Border.all(
                              color: isSelected
                                  ? primaryDark
                                  : Colors.transparent,
                              width: 2,
                            ),
```

```
                    ),
                    child: Center(
                      child: Text(
                        emoji,
                        style: const TextStyle(fontSize: 28),
                      ),
                    ),
                  ),
                );
              },
            ),
          ),
          const SizedBox(height: 20),

          // Recipe Name
          TextFormField(
            controller: _nameController,
            decoration: InputDecoration(
              labelText: 'Nama Resep',
              border: OutlineInputBorder(
                borderRadius: BorderRadius.circular(12),
              ),
              prefixIcon: const Icon(Icons.restaurant_menu),
```

```
            isDense: true,
          ),
          validator: (value) {
            if (value == null || value.isEmpty) {
              return 'Nama resep tidak boleh kosong';
            }
            return null;
          },
        ),
        const SizedBox(height: 16),

        _buildChipSelector(
          label: 'Kategori',
          options: _categories,
          selectedValue: _selectedCategory,
          onSelected: (value) {
            setState(() {
              _selectedCategory = value;
            });
          },
        ),
        const SizedBox(height: 16),
```

```
                    _buildChipSelector(
                      label: 'Tingkat Kesulitan',
                      options: _difficulties,
                      selectedValue: _selectedDifficulty,
                      onSelected: (value) {
                        setState(() {
                          _selectedDifficulty = value;
                        });
                      },
                    ),
                    const SizedBox(height: 16),


                    // Cook Time (Diubah untuk input angka)
                    TextFormField(
                      controller: _cookTimeController,
                      // HANYA MENGIZINKAN INPUT ANGKA
                      keyboardType: TextInputType.number,
                      decoration: InputDecoration(
                           labelText: 'Waktu Memasak (dalam menit, contoh:
30)',
                               suffixText: 'menit', // Menampilkan 'menit'
sebagai suffix
                        border: OutlineInputBorder(
                          borderRadius: BorderRadius.circular(12),
```

```
                        ),
                        prefixIcon: const Icon(Icons.access_time),
                        isDense: true,
                      ),
                      validator: (value) {
                        if (value == null || value.isEmpty) {
                          return 'Waktu memasak tidak boleh kosong';
                        }
                          // Validasi sederhana: pastikan hanya mengandung
digit
                        if (int.tryParse(value) == null) {
                            return 'Masukkan angka yang valid untuk waktu
memasak';
                        }
                        return null;
                      },
                    ),
                    const SizedBox(height: 16),

                    // Description
                    TextFormField(
                      controller: _descriptionController,
                      decoration: InputDecoration(
                        labelText: 'Deskripsi',
```

```
                    border: OutlineInputBorder(
                      borderRadius: BorderRadius.circular(12),
                    ),
                    prefixIcon: const Icon(Icons.description),
                    isDense: true,
                  ),
                  maxLines: 2,
                  validator: (value) {
                    if (value == null || value.isEmpty) {
                      return 'Deskripsi tidak boleh kosong';
                    }
                    return null;
                  },
                ),
                const SizedBox(height: 16),

                // Ingredients
                TextFormField(
                  controller: _ingredientsController,
                  decoration: InputDecoration(
                    labelText: 'Bahan-bahan',
                    hintText: 'Pisahkan dengan koma atau enter',
                    border: OutlineInputBorder(
```

```
                borderRadius: BorderRadius.circular(12),
              ),
              prefixIcon: const Icon(Icons.list_alt),
              isDense: true,
            ),
            maxLines: 3,
            validator: (value) {
              if (value == null || value.isEmpty) {
                return 'Bahan-bahan tidak boleh kosong';
              }
              return null;
            },
          ),
          const SizedBox(height: 20),

          // Save Button
          ElevatedButton(
            onPressed: _saveRecipe,
            style: ElevatedButton.styleFrom(
              backgroundColor: primaryDark,
              padding: const EdgeInsets.symmetric(vertical: 14),
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(12),
```

```
                        ),
                        elevation: 4, // Menambah elevasi tombol
                      ),
                      child: Text(
                        isEditMode ? 'Perbarui Resep' : 'Simpan Resep',
                        style: const TextStyle(
                          fontSize: 16,
                          fontWeight: FontWeight.bold,
                          color: Colors.white,
                        ),
                      ),
                    ),
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

teredRecipes.where((r) => favorites.contains(r.id)).toList();

}

3. Jika pengguna mengaktifkan filter favorit, daftar disaring lagi, hanya menyisakan resep yang ID-nya ditemukan di dalam favoritesProvider (favorites.contains(r.id)).

Hasil akhir dari proses tiga langkah ini adalah daftar **filteredRecipes**, yang kemudian digunakan untuk membangun tampilan di RecipeHomePage.

2.9.8 AddRecipeDialog.dart

1. **Struktur Widget dan Riverpod**

Dart

```
class AddRecipeDialog extends ConsumerStatefulWidget {
  final Recipe? recipe;
  const AddRecipeDialog({super.key, this.recipe});
  // ...
}
```

- **ConsumerStatefulWidget:** Digunakan karena *widget* ini perlu **mengelola *state* lokal** (input *controller*, kategori yang dipilih, dll.) dan juga perlu **berinteraksi dengan Riverpod** (Consumer).
- **Properti recipe:** Properti ini bersifat *nullable* (Recipe?).
    - Jika recipe adalah **null**, dialog berfungsi sebagai mode **Tambah Resep (*Add Mode*)**.
    - Jika recipe **ada isinya**, dialog berfungsi sebagai mode **Edit Resep (*Edit Mode*)**, dan semua *field* akan diisi dengan data resep tersebut.

2. **State Lokal dan Controller**

Dart

```
class _AddRecipeDialogState extends ConsumerState<AddRecipeDialog> {
```

```
  final _formKey = GlobalKey<FormState>(); // Kunci untuk validasi form
  final _nameController = TextEditingController();
  // ... controller lainnya

  String _selectedCategory = 'Indonesian';
  String _selectedDifficulty = 'Mudah';
  String _selectedEmoji = '🔍'; // Menyimpan icon/gambar (saat ini menggunakan emoji)

  // Daftar pilihan yang tersedia
  final List<String> _categories = ['Indonesian', 'Western', 'Dessert'];
  final List<String> _difficulties = ['Mudah', 'Sedang', 'Sulit'];
  final List<String> _emojis = [ /* ... daftar emoji panjang ... */ ];

  // Warna kustom
  static const Color primaryDark = Color(0xFFE65100);
  // ...
}
```

- **TextEditingController:** Digunakan untuk mengontrol dan mendapatkan input teks dari TextFormField (Nama, Waktu Masak, Deskripsi, Bahan).
- **Variabel State (_selected...):** Menyimpan nilai yang dipilih pengguna dari *dropdown* atau *chip* sebelum disimpan. Perubahan pada variabel ini memicu setState().

- **List Statis:** Daftar kategori, kesulitan, dan emoji yang tersedia untuk dipilih.

3. Fungsi `initState` (Mode Edit)

Dart

```
@override
void initState() {
  super.initState();
  if (widget.recipe != null) {
    // Mode Edit: Mengisi field dengan data resep yang dilewatkan
    _nameController.text = widget.recipe!.name;
    // Logika untuk CookTime: Memisahkan "20 menit" menjadi hanya "20"
    final cookTimeParts = widget.recipe!.cookTime.split(' ');
    _cookTimeController.text = cookTimeParts.isNotEmpty && cookTimeParts.first.isNotEmpty
        ? cookTimeParts.first
        : '';
    _descriptionController.text = widget.recipe!.description;
    // Menggabungkan List<String> ingredients menjadi satu String dengan pemisah koma
    _ingredientsController.text = widget.recipe!.ingredients.join(', ');
    _selectedCategory = widget.recipe!.category;
    _selectedDifficulty = widget.recipe!.difficulty;
    _selectedEmoji = widget.recipe!.image;
```

```
  }
}
```

- Fungsi ini dijalankan satu kali saat *widget* pertama kali dibuat.
- Jika widget.recipe **ada**, ini berarti sedang dalam mode Edit. Semua *controller* dan variabel *state* lokal diisi (*pre-filled*) dengan data resep yang akan diedit.

4. Fungsi `_saveRecipe` (Logika Bisnis)

Fungsi ini dipanggil saat tombol "Simpan Resep" / "Perbarui Resep" ditekan.

A. Validasi dan Pemrosesan Input

Dart

```
void _saveRecipe() {
  if (_formKey.currentState!.validate()) {
    // 1. Ambil list resep saat ini dari Riverpod
    final recipes = ref.read(recipesProvider);
    // 2. Pemrosesan Bahan-bahan (Ingredients)
    final ingredientsList = _ingredientsController.text
        .split(RegExp(r'[,\n]')) // Pisahkan input berdasarkan koma (,) atau baris baru (\n)
        .map((e) => e.trim())     // Hapus spasi di awal/akhir setiap bahan
        .where((e) => e.isNotEmpty) // Hapus string kosong
        .toList();
    // 3. Pemrosesan Waktu Memasak (Cook Time)
    final rawCookTime = _cookTimeController.text.trim();
```

```
    final cookTime = rawCookTime.isNotEmpty
        ? '$rawCookTime menit' // Tambahkan kata ' menit' ke input angka
        : '0 menit';
    // ...
  }
}
```

B. Logika Edit Mode

Dart

```
  if (widget.recipe != null) {
    // 1. Buat objek Recipe BARU dengan data yang diperbarui
    final updatedRecipe = widget.recipe!.copyWith(
      name: _nameController.text,
      // ... semua field diperbarui
      ingredients: ingredientsList,
    );
    // 2. Cari dan Ganti Resep Lama
    final updatedRecipes = recipes.map((r) {
      // Jika ID resep cocok, ganti dengan updatedRecipe, jika tidak, biarkan resep lama (r)
      return r.id == updatedRecipe.id ? updatedRecipe : r;
    }).toList();
```

```
    // 3. Perbarui State Riverpod
    ref.read(recipesProvider.notifier).state = updatedRecipes;
    // 4. Tutup Dialog dan tampilkan Snackbar
    Navigator.pop(context);
    ScaffoldMessenger.of(context).showSnackBar(...);
  }
```

C. Logika Add Mode

Dart

```
  else {
    // 1. Hitung ID baru (berdasarkan panjang list)
    final newId = (recipes.length + 1).toString();

    // 2. Buat objek Recipe BARU
    final newRecipe = Recipe(
      id: newId,
      // ... isi data dari controller dan state lokal
      ingredients: ingredientsList,
    );

    // 3. Perbarui State Riverpod (tambahkan resep baru ke list)
```

```
    ref.read(recipesProvider.notifier).state = [...recipes, newRecipe];
    // Menggunakan spread operator `...` untuk membuat list baru (immutable update)


    // 4. Tutup Dialog dan tampilkan Snackbar
    Navigator.pop(context);
    ScaffoldMessenger.of(context).showSnackBar(...);
  }
```

5. Metode `_buildChipSelector` (Widget Kustom)

```
Dart
Widget _buildChipSelector({
  required String label,
  required List<String> options,
  required String selectedValue,
  required ValueSetter<String> onSelected,
}) {
  // ... kode UI untuk ChoiceChip
}
```

- **Tujuan:** Fungsi ini membuat *widget* yang dapat digunakan kembali (*reusable*) untuk memilih nilai dari daftar opsi menggunakan **`ChoiceChip`** (digunakan untuk memilih Kategori dan Tingkat Kesulitan). Ini bertujuan untuk membuat kode `build` menjadi lebih rapi dan konsisten.

6. Metode `build` (Struktur UI)

Metode ini bertanggung jawab untuk merender tampilan dialog.

- **Header:** Menampilkan judul "Tambah Resep" atau "Edit Resep" dengan warna latar belakang primaryDark (oranye gelap) yang konsisten.
- **Form:** Mengandung ListView yang berisi semua *input field* dan selector.
    - **Emoji Selector:** Menampilkan GridView.builder horizontal untuk memilih ikon resep.
    - **TextFormField:** Digunakan untuk Nama, Waktu Memasak, Deskripsi, dan Bahan-bahan, lengkap dengan InputDecoration yang rapi dan fungsi validator untuk memastikan *field* tidak kosong.
    - **_buildChipSelector:** Dipanggil dua kali untuk Kategori dan Tingkat Kesulitan.
- **Save Button (ElevatedButton):** Tombol yang memicu fungsi _saveRecipe. Teks tombol berubah tergantung pada mode (Tambah atau Edit).

2.9.9 home_screen.dart

A. Ringkasan Fungsionalitas

Kode ini membangun layar beranda yang komprehensif untuk aplikasi katalog resep, termasuk fitur-fitur utama berikut:

1. **App Bar & Branding:** Menampilkan judul aplikasi **'Katalog Resep Masakan'** dengan skema warna yang konsisten (primaryDark).
2. **Toolbar Interaktif:**
    - **Pencarian Resep:** Menggunakan TextField untuk memfilter resep berdasarkan nama atau deskripsi, memperbarui searchQueryProvider.
    - **Toggle Tampilan:** Tombol untuk beralih antara tampilan **Grid** (Icons.grid_view) dan **List** (Icons.list), memperbarui viewModeProvider.
    - **Filter Favorit:** Tombol hati yang dapat di-toggle untuk hanya menampilkan resep yang telah ditandai sebagai favorit, memperbarui showOnlyFavoritesProvider.

3. **Filter Kategori:** Bagian ListView horizontal yang menampilkan CategoryChip untuk memfilter resep berdasarkan kategori (Semua, Indonesian, Western, Dessert), memperbarui selectedCategoryProvider.
4. **Tampilan Konten Utama:** Menggunakan **filteredRecipesProvider** untuk menampilkan resep yang sudah difilter:
    - Jika mode tampilan adalah **Grid**, ia menggunakan GridView.builder dengan RecipeCard.
    - Jika mode tampilan adalah **List**, ia menggunakan ListView.builder dengan RecipeListItem.
5. **Status Kosong (Empty State):** Menampilkan pesan dan ikon yang relevan jika tidak ada resep yang ditemukan (baik karena pencarian/filter kategori atau saat filter favorit aktif).
6. **Tombol Tambah Resep:** FloatingActionButton untuk membuka AddRecipeDialog (AddRecipeScreen.dart), memungkinkan pengguna untuk menambahkan resep baru.

2.9.10 splash_screen.dart

```
import 'package:flutter/material.dart';

import 'home_screen.dart';



class SplashScreen extends StatefulWidget {

  const SplashScreen({super.key});



  @override

  State<SplashScreen> createState() => _SplashScreenState();

}



class _SplashScreenState extends State<SplashScreen>

    with SingleTickerProviderStateMixin {
```

```dart
late AnimationController _controller;

late Animation<double> _fadeAnimation;

late Animation<double> _scaleAnimation;



@override

void initState() {

  super.initState();



  _controller = AnimationController(

    duration: const Duration(milliseconds: 1500),

    vsync: this,

  );



  _fadeAnimation = Tween<double>(begin: 0.0, end: 1.0).animate(

    CurvedAnimation(

      parent: _controller,

      curve: const Interval(0.0, 0.6, curve: Curves.easeIn),

    ),

  );



  _scaleAnimation = Tween<double>(begin: 0.5, end: 1.0).animate(

    CurvedAnimation(

      parent: _controller,
```

```
      curve: const Interval(0.0, 0.6, curve: Curves.easeOutBack),
    ),
  );

  _controller.forward();

  // Navigasi ke home setelah 3 detik
  Future.delayed(const Duration(seconds: 3), () {
    if (mounted) {
      Navigator.pushReplacement(
        context,
        PageRouteBuilder(
          pageBuilder: (context, animation, secondaryAnimation) =>
              const RecipeHomePage(),
          transitionsBuilder:
              (context, animation, secondaryAnimation, child) {
            return FadeTransition(opacity: animation, child: child);
          },
          transitionDuration: const Duration(milliseconds: 500),
        ),
      );
    }
  });
```

```
}


@override
void dispose() {
  _controller.dispose();
  super.dispose();
}


@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      decoration: BoxDecoration(
        gradient: LinearGradient(
          begin: Alignment.topLeft,
          end: Alignment.bottomRight,
          colors: [
            Colors.orange[400]!,
            Colors.orange[700]!,
          ],
        ),
      ),
      child: Center(
```

```
          child: FadeTransition(
            opacity: _fadeAnimation,
            child: ScaleTransition(
              scale: _scaleAnimation,
              child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  // 📖 Ikon tanpa background
                  const Text(
                    '📖',
                    style: TextStyle(
                      fontSize: 100,
                      shadows: [
                        Shadow(
                          blurRadius: 8,
                          color: Colors.black26,
                          offset: Offset(2, 2),
                        ),
                      ],
                    ),
                  ),
                  const SizedBox(height: 30),
                  // Judul
```

```
                  const Text(
                    'MyRecipe',
                    style: TextStyle(
                      fontSize: 32,
                      fontWeight: FontWeight.bold,
                      color: Colors.white,
                      letterSpacing: 1.5,
                    ),
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Bagian ini menjelaskan struktur, fungsionalitas, dan teknologi utama yang digunakan dalam pengembangan antarmuka pengguna (UI) dan manajemen status (state management) aplikasi "MyRecipe".

A. Implementasi Layar Pembuka (Splash Screen)

Layar pembuka (SplashScreen) adalah layar pertama yang dilihat pengguna saat aplikasi dimuat, berfungsi sebagai penarik perhatian sekaligus memberi waktu aplikasi untuk menginisialisasi sumber daya yang diperlukan.

1. Tujuan

- Memberikan pengalaman **transisi yang mulus** dan menarik secara visual saat memuat aplikasi.
- Memberikan **waktu tunda (delay)** selama 3 detik sebelum menavigasi ke halaman utama, memastikan semua inisialisasi Riverpod dan *widget* berjalan dengan baik.

2. Teknik Animasi

Layar ini menggunakan teknik animasi eksplisit pada Flutter dengan:

Komponen	**Implementasi**	**Fungsi**
SingleTickerProviderStateMixin	Digunakan pada _SplashScreenState.	Menyediakan *ticker* untuk menggerakkan **AnimationController**.
AnimationController	Durasi 1500 ms (1,5 detik).	Mengatur alur waktu dan kecepatan animasi.
_scaleAnimation	**Tween<double>(begin: 0.5, end: 1.0)** dengan Curves.easeOutBack.	Membuat ikon dan judul muncul dengan efek **"scale up"** dinamis seolah melompat dari kecil ke ukuran penuh.

_fadeAnimation	**Tween<double>(begin: 0.0, end: 1.0)** dengan Curves.easeIn.	Membuat elemen muncul secara perlahan **(fade in)**.
Interval(0.0, 0.6)	Diterapkan pada kedua animasi.	Memastikan animasi skala dan *fade* selesai dalam 60% durasi *controller* (sekitar 900 ms) untuk efek yang cepat dan energik.

3. Navigasi

Navigasi dari SplashScreen ke RecipeHomePage dilakukan menggunakan:

- **Future.delayed(const Duration(seconds: 3))**: Menjamin minimal 3 detik waktu tunggu total (termasuk durasi animasi).
- **Navigator.pushReplacement**: Digunakan agar pengguna tidak dapat kembali ke layar *splash* menggunakan tombol kembali.
- **PageRouteBuilder**: Digunakan untuk mengimplementasikan transisi kustom (500 ms **FadeTransition**) saat beralih ke RecipeHomePage, menghasilkan perpindahan layar yang lebih halus.

B. Implementasi Halaman Utama Resep (RecipeHomePage)

RecipeHomePage adalah pusat dari aplikasi, yang bertanggung jawab untuk menampilkan, memfilter, dan mengelola resep menggunakan **Flutter Riverpod** sebagai solusi manajemen status.

1. Arsitektur dan Manajemen Status (Riverpod)

Kelas ini merupakan **ConsumerStatefulWidget** yang secara aktif "mendengarkan" (watch) perubahan status dari **recipe_providers.dart**:

Provider yang Di-watch	**Tipe State**	**Keterangan**

filteredRecipesProvider	List<Recipe>	Daftar resep akhir yang sudah disaring berdasarkan semua kriteria.
selectedCategoryProvider	String	Kategori yang sedang aktif difilter ('Indonesian', 'Western', dll.).
viewModeProvider	ViewMode	Mode tampilan yang dipilih pengguna (grid atau list).
showOnlyFavoritesProvider	bool	Status filter favorit (aktif atau non-aktif).

2. Struktur Tata Letak (Layout Structure)

Antarmuka pengguna dibagi menjadi tiga area fungsional utama di bawah *App Bar*:

a) Top Toolbar (Pencarian & Toggle)

Toolbar ini mengelompokkan alat interaktif utama:

- **Search Bar (TextField):** Memperbarui **searchQueryProvider** secara *real-time* melalui onChanged. Tombol *clear* (suffixIcon) disediakan untuk kemudahan penggunaan.
- **Toggle Mode Tampilan:** Tombol dengan ikon yang dapat berganti antara Icons.grid_view dan Icons.list. Klik tombol ini memperbarui **viewModeProvider**, yang secara otomatis memicu penggantian *widget* tampilan utama.
- **Toggle Favorit:** Tombol hati yang mengubah status **showOnlyFavoritesProvider**. Ketika aktif, ia memunculkan *alert banner* (Bagian C) dan memfilter daftar resep.

b) Category Filter Section

Berisi ListView horizontal yang terdiri dari **CategoryChip** *widget* yang dapat diklik. Setiap *chip* mengontrol status **selectedCategoryProvider**, memicu proses filter.

c) Favorite Alert Banner

Sebuah *widget* Container yang hanya ditampilkan (if (showOnlyFavorites)) ketika filter favorit diaktifkan. Memberikan konfirmasi visual kepada pengguna bahwa mereka berada dalam mode filter favorit dan menawarkan tombol pintas **"Tampilkan Semua"** untuk menonaktifkan filter.

d) Main Content (Daftar Resep)

Bagian utama yang menampilkan hasil resep:

- **Mode Grid:** Menggunakan **GridView.builder** dengan **RecipeCard** untuk tata letak kolom ganda yang ringkas.
- **Mode List:** Menggunakan **ListView.builder** dengan **RecipeListItem** untuk tampilan daftar yang lebih detail.
- **Empty State (_buildEmptyState):** Jika filteredRecipes kosong, menampilkan pesan ramah (**'Resep tidak ditemukan'** atau **'Belum ada resep favorit'**) sesuai dengan filter yang aktif, meningkatkan pengalaman pengguna.

3. Fungsionalitas Tambah Resep

FloatingActionButton diposisikan di sudut kanan bawah. Ketika ditekan, ia memanggil showDialog untuk menampilkan **AddRecipeDialog** (add_recipe_screen.dart), memungkinkan penambahan data resep baru ke aplikasi.

2.9.11 category_chip.dart

```
import 'package:flutter/material.dart';



class CategoryChip extends StatelessWidget {

  final String label;

  final bool isSelected;

  final VoidCallback onTap;



  const CategoryChip({

    super.key,

    required this.label,

    required this.isSelected,

    required this.onTap,

  });



  // Definisikan warna yang digunakan
```

```
static const Color primaryColor = Color(0xFFFF6F00); // Oranye utama


@override
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.symmetric(horizontal: 5, vertical: 4),
    child: GestureDetector(
      onTap: onTap,
      child: AnimatedContainer(
        duration: const Duration(milliseconds: 250),
        curve: Curves.easeInOut,
        padding: const EdgeInsets.symmetric(
          horizontal: 16,
          vertical: 8,
        ),
        decoration: BoxDecoration(
          // 1. Latar Belakang Penuh
          color: isSelected ? primaryColor : Colors.white,
          borderRadius: BorderRadius.circular(25),


          // 2. Bayangan/Elevasi
          boxShadow: isSelected
              ? [
```

```
                BoxShadow(
                  color: primaryColor.withOpacity(0.4),
                  blurRadius: 6,
                  offset: const Offset(0, 3),
                ),
              ]
            : [
                BoxShadow(
                  color: Colors.black.withOpacity(0.05),
                  blurRadius: 2,
                  offset: const Offset(0, 1),
                ),
              ],


        border: Border.all(
          color: isSelected ? primaryColor : Colors.grey.shade300,
          width: 1,
        ),
      ),
      child: Text(
        label,
        style: TextStyle(
          fontSize: 13,
```

```
              fontWeight: isSelected ? FontWeight.bold : FontWeight.w500,
              color: isSelected ? Colors.white : Colors.black87,
            ),
          ),
        ),
      ),
    );
  }
}
```

Widget CategoryChip adalah komponen UI fundamental yang digunakan di RecipeHomePage untuk memfilter daftar resep berdasarkan kategori. Implementasi widget ini menekankan pada desain modern, responsif, dan penggunaan animasi halus untuk memberikan feedback visual yang jelas kepada pengguna.

1. Struktur dan Prinsip Dasar

- StatelessWidget: CategoryChip adalah widget tanpa status internal. Status terpilih (isSelected) dan aksi yang terjadi saat diklik (onTap) diproses di luar, tepatnya di provider Riverpod pada RecipeHomePage. Hal ini menjaga prinsip pemisahan perhatian (separation of concerns).
- Properti Wajib:
    - label (String): Nama kategori yang ditampilkan (misalnya, "Indonesian", "Dessert").
    - isSelected (bool): Status apakah chip ini sedang aktif.
    - onTap (VoidCallback): Fungsi yang dipanggil saat chip diklik (digunakan untuk memperbarui selectedCategoryProvider di Riverpod).

2. Desain Visual dan Interaksi Responsif

Desain chip ini sangat responsif terhadap status isSelected, menggunakan warna utama (primaryColor: 0xFFFF6F00 - Oranye) untuk memberikan penekanan pada kategori yang aktif.

Elemen Styling	Ketika isSelected adalah true (Aktif)	Ketika isSelected adalah false (Non-aktif)
Warna Latar Belakang	primaryColor (Oranye Solid)	Colors.white (Putih)
Warna Teks	Colors.white	Colors.black87
Ketebalan Teks	FontWeight.bold	FontWeight.w500
Border	Border berwarna primaryColor	Border tipis Colors.grey.shade300
Box Shadow (Bayangan)	Diberi bayangan oranye yang menonjol untuk efek elevasi 3D yang kuat.	Diberi bayangan hitam tipis yang minim.

Tentu, berikut adalah penjelasan detail mengenai widget CategoryChip untuk dimasukkan ke dalam laporan Anda. Penjelasan ini fokus pada desain, interaksi, dan penggunaan animasi untuk meningkatkan pengalaman pengguna (UX).

3. Penggunaan Animasi (AnimatedContainer)

Untuk transisi visual yang halus antara status aktif dan non-aktif, CategoryChip menggunakan AnimatedContainer dan bukan Container statis.

- Durasi & Kurva: duration: const Duration(milliseconds: 250) dan curve: Curves.easeInOut.
- Fungsi: Ketika status isSelected berubah (misalnya, dari false ke true), AnimatedContainer akan secara otomatis menginterpolasi perubahan properti visual (warna latar belakang, warna border, dan efek box shadow) selama 250 milidetik, menghasilkan transisi yang mulus dan profesional, bukan perubahan yang tiba-tiba.

4. Peningkatan UX

Penggunaan GestureDetector yang membungkus AnimatedContainer memastikan bahwa seluruh area chip dapat diklik. Kombinasi dari kontras warna yang tinggi pada status aktif dan animasi transisi berfungsi sebagai feedback visual instan, menegaskan kepada pengguna bahwa kategori telah berhasil dipilih dan filter sedang diterapkan.

5. Detail Layout

- Padding di dalam ListView kategori: Menggunakan padding: const EdgeInsets.symmetric(horizontal: 5, vertical: 4) untuk memberikan ruang antar chip dan sedikit ruang vertikal.
- padding di dalam AnimatedContainer: Menggunakan horizontal: 16 dan vertical: 8 untuk memberikan padding internal yang cukup, membuat chip terlihat seperti pil yang nyaman dibaca dan diklik.

2.9.12 recipe_card.dart

Widget CategoryChip adalah komponen UI fundamental yang digunakan di RecipeHomePage untuk memfilter daftar resep berdasarkan kategori. Implementasi widget ini menekankan pada desain modern, responsif, dan penggunaan animasi halus untuk memberikan feedback visual yang jelas kepada pengguna.

1. Struktur dan Prinsip Dasar

- StatelessWidget: CategoryChip adalah widget tanpa status internal. Status terpilih (isSelected) dan aksi yang terjadi saat diklik (onTap) diproses di luar, tepatnya di provider Riverpod pada RecipeHomePage. Hal ini menjaga prinsip pemisahan perhatian (separation of concerns).
- Properti Wajib:
    - label (String): Nama kategori yang ditampilkan (misalnya, "Indonesian", "Dessert").
    - isSelected (bool): Status apakah chip ini sedang aktif.
    - onTap (VoidCallback): Fungsi yang dipanggil saat chip diklik (digunakan untuk memperbarui selectedCategoryProvider di Riverpod).

2. Desain Visual dan Interaksi Responsif

Desain chip ini sangat responsif terhadap status isSelected, menggunakan warna utama (primaryColor: 0xFFFF6F00 - Oranye) untuk memberikan penekanan pada kategori yang aktif.

Elemen Styling	Ketika isSelected adalah true (Aktif)	Ketika isSelected adalah false (Non-aktif)
Warna Latar Belakang	primaryColor (Oranye Solid)	Colors.white (Putih)
Warna Teks	Colors.white	Colors.black87
Ketebalan Teks	FontWeight.bold	FontWeight.w500
Border	Border berwarna primaryColor	Border tipis Colors.grey.shade300
Box Shadow (Bayangan)	Diberi bayangan oranye yang menonjol untuk efek elevasi 3D yang kuat.	Diberi bayangan hitam tipis yang minim.

3. Penggunaan Animasi (AnimatedContainer)

Untuk transisi visual yang halus antara status aktif dan non-aktif, CategoryChip menggunakan AnimatedContainer dan bukan Container statis.

- Durasi & Kurva: duration: const Duration(milliseconds: 250) dan curve: Curves.easeInOut.
- Fungsi: Ketika status isSelected berubah (misalnya, dari false ke true), AnimatedContainer akan secara otomatis menginterpolasi perubahan properti visual (warna latar belakang, warna border, dan efek box shadow) selama 250 milidetik, menghasilkan transisi yang mulus dan profesional, bukan perubahan yang tiba-tiba.

4. Peningkatan UX

Penggunaan GestureDetector yang membungkus AnimatedContainer memastikan bahwa seluruh area chip dapat diklik. Kombinasi dari kontras warna yang tinggi pada status aktif dan animasi transisi berfungsi sebagai feedback visual instan, menegaskan kepada pengguna bahwa kategori telah berhasil dipilih dan filter sedang diterapkan.

5. Detail Layout

- Padding di dalam ListView kategori: Menggunakan padding: const EdgeInsets.symmetric(horizontal: 5, vertical: 4) untuk memberikan ruang antar chip dan sedikit ruang vertikal.

- padding di dalam AnimatedContainer: Menggunakan horizontal: 16 dan vertical: 8 untuk memberikan padding internal yang cukup, membuat chip terlihat seperti pil yang nyaman dibaca dan diklik.

2.9.13 recipe_detail_sheet

RecipeDetailSheet adalah komponen kunci yang bertanggung jawab menampilkan semua detail lengkap resep dalam format *modal* yang interaktif. Ia menggunakan DraggableScrollableSheet untuk memberikan pengalaman pengguna yang fleksibel dan mulus pada perangkat seluler.

1. Struktur dan Prinsip Dasar

- **ConsumerWidget:** *Widget* ini adalah ConsumerWidget yang menerima objek **Recipe** tunggal. Meskipun tidak memiliki status lokal, ia dapat "menonton" (watch) status global **favoritesProvider** dari Riverpod untuk menentukan status favorit resep saat ini.
- **Wadah Utama (DraggableScrollableSheet):** *Widget* ini dimuat melalui showModalBottomSheet dengan properti isScrollControlled: true. Hal ini memungkinkan tampilan detail menjadi **setengah layar** (initialChildSize: 0.7) secara default, dan dapat ditarik (draggable) oleh pengguna hingga hampir penuh (maxChildSize: 0.9) atau dikecilkan (minChildSize: 0.5).
- **Visual Handle:** Terdapat indikator tarik (Drag Handle) di bagian atas *sheet* untuk memberikan petunjuk visual kepada pengguna bahwa konten dapat ditarik dan diatur ukurannya.

2. Penyajian Data dan Desain

Tampilan ini menyajikan data resep secara hierarkis dan visual:

Bagian Tampilan	Implementasi	Fungsi

Header	**Emoji dan Nama Resep**	Emoji resep ditampilkan dengan ukuran besar (fontSize: 80) sebagai representasi visual. Nama resep ditampilkan dengan tebal (fontWeight: FontWeight.bold).
Info Chips	Baris horizontal menggunakan sub-widget **_buildInfoChip**.	Menampilkan detail ringkas (Waktu Memasak dan Tingkat Kesulitan) dengan latar belakang dan ikon berwarna (Biru untuk Waktu, Hijau untuk Kesulitan) untuk keterbacaan yang cepat.
Category Badge	**Container** berwarna oranye muda.	Menampilkan kategori resep, memberikan *feedback* visual yang jelas mengenai klasifikasi resep.
Deskripsi	Teks deskripsi resep.	Menggunakan TextStyle yang nyaman dibaca dengan *line height* yang baik (height: 1.5).
Bahan-Bahan	Daftar bahan-bahan (recipe.ingredients) yang di-***map*** menjadi serangkaian Row *widget*.	Setiap bahan dihiasi dengan ikon centang oranye (Icons.check_circle) untuk format *bullet list* yang menarik.

3. Fungsionalitas Interaktif (Aksi)

Terdapat tiga tombol aksi utama di dalam *detail sheet* yang berinteraksi langsung dengan manajemen status Riverpod.

a. Tombol Favorit

- **Implementasi:** Menggunakan IconButton dengan desain yang fleksibel (warna merah saat aktif, abu-abu saat non-aktif).
- **Logika Riverpod:** Sama seperti di RecipeCard, saat ditekan, ia memodifikasi **favoritesProvider** (Set<String>) dengan menambahkan atau menghapus recipe.id. Perubahan ini otomatis diperbarui di seluruh aplikasi.

b. Tombol Edit

- **Fungsi:** Memanggil _editRecipe(context).
- **Mekanisme:**
    1. Menutup RecipeDetailSheet yang sedang aktif (Navigator.pop(context)).
    2. Memunculkan **AddRecipeDialog** (yang merupakan *widget* yang sama untuk menambah resep) dan meneruskan objek recipe saat ini sebagai parameter. Hal ini memungkinkan *form* AddRecipeDialog terisi dengan data resep yang ada, mengubah fungsionalitasnya menjadi **mode edit**.

c. Tombol Hapus

- **Fungsi:** Memanggil _showDeleteConfirmation(context, ref).
- **Mekanisme Penghapusan:**
    1. Memunculkan AlertDialog untuk konfirmasi pengguna, memastikan penghapusan yang tidak disengaja dapat dicegah.
    2. Jika tombol **'Hapus'** ditekan:
        - Resep dihapus dari **recipesProvider** (daftar resep utama) dan **favoritesProvider** (jika resep tersebut merupakan favorit).
        - Dua level navigasi dipicu: menutup *dialog* konfirmasi, dan kemudian menutup *bottom sheet* detail resep.
    3. **Umpan Balik:** Pesan konfirmasi keberhasilan penghapusan ditampilkan melalui **SnackBar** berwarna merah.

2.9.14 recipe_list_item

```
import 'package:flutter/material.dart';

import 'package:flutter_riverpod/flutter_riverpod.dart';

import '../models/recipe.dart';
```

```dart
import '../providers/recipe_providers.dart';
import 'recipe_detail_sheet.dart';


class RecipeListItem extends ConsumerStatefulWidget {
  final Recipe recipe;


  const RecipeListItem({super.key, required this.recipe});


  @override
  ConsumerState<RecipeListItem> createState() => _RecipeListItemState();
}


class _RecipeListItemState extends ConsumerState<RecipeListItem>
    with SingleTickerProviderStateMixin {
  late AnimationController _controller;
  late Animation<double> _scaleAnimation;


  @override
  void initState() {
    super.initState();
    _controller = AnimationController(
      duration: const Duration(milliseconds: 150),
      vsync: this,
```

```
    );
    _scaleAnimation = Tween<double>(begin: 1.0, end: 0.97).animate(
      CurvedAnimation(parent: _controller, curve: Curves.easeInOut),
    );
  }

  @override
  void dispose() {
    _controller.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    final favorites = ref.watch(favoritesProvider);
    final isFavorite = favorites.contains(widget.recipe.id);

    return ScaleTransition(
      scale: _scaleAnimation,
      child: GestureDetector(
        onTapDown: (_) => _controller.forward(),
        onTapUp: (_) => _controller.reverse(),
        onTapCancel: () => _controller.reverse(),
```

```
        onTap: () {
          _controller.reverse();
          showModalBottomSheet(
            context: context,
            isScrollControlled: true,
            backgroundColor: Colors.transparent,
            builder: (context) => RecipeDetailSheet(recipe:
widget.recipe),
          );
        },
        child: Padding(
          padding: const EdgeInsets.only(bottom: 12),
          child: Container(
            padding: const EdgeInsets.all(12),
            decoration: BoxDecoration(
              color: Colors.white,
              borderRadius: BorderRadius.circular(12),
              boxShadow: [
                BoxShadow(
                  color: Colors.grey.withOpacity(0.1),
                  spreadRadius: 1,
                  blurRadius: 5,
                  offset: const Offset(0, 3),
                ),
```

```
            ],
          ),
          child: Row(
            children: [
              // Emoji / Image
              Text(
                widget.recipe.image,
                style: const TextStyle(fontSize: 30),
              ),
              const SizedBox(width: 16),
              // Recipe Info
              Expanded(
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.start,
                  children: [
                    Text(
                      widget.recipe.name,
                      style: const TextStyle(
                        fontSize: 16,
                        fontWeight: FontWeight.bold,
                      ),
                      maxLines: 1,
                      overflow: TextOverflow.ellipsis,
```

```
                ),
                const SizedBox(height: 4),
                // Bagian Rating dihapus

                const SizedBox(height: 4),
                Row(
                  children: [
                    const Icon(Icons.access_time,
                        size: 14, color: Colors.grey),
                    const SizedBox(width: 2),
                    Expanded(
                      child: Text(
                        widget.recipe.cookTime,
                        style: const TextStyle(
                          fontSize: 11,
                          color: Colors.grey,
                        ),
                        overflow: TextOverflow.ellipsis,
                      ),
                    ),
                  ],
                ),
              ],
```

```
                  ),
                ),

                // Favorite Button
                IconButton(
                  icon: Icon(
                    isFavorite ? Icons.favorite : Icons.favorite_border,
                    color: isFavorite ? Colors.red : Colors.grey,
                  ),
                  onPressed: () {
                    final newFavorites = Set<String>.from(favorites);
                    if (isFavorite) {
                      newFavorites.remove(widget.recipe.id);
                    } else {
                      newFavorites.add(widget.recipe.id);
                    }
                    ref.read(favoritesProvider.notifier).state =
newFavorites;},),],),),),); }
}
```

RecipeListItem adalah *widget* interaktif yang menampilkan detail esensial sebuah resep dalam satu baris daftar, memungkinkan pengguna untuk melihat detail lebih lanjut dengan ketukan (*tap*).

1. Struktur Widget dan Riverpod

- **ConsumerStatefulWidget:** RecipeListItem didefinisikan sebagai ConsumerStatefulWidget. Hal ini diperlukan karena *widget* ini mengelola *state* animasi lokal (**_RecipeListItemState**), dan juga perlu mengakses *state* global Riverpod (khususnya **favoritesProvider**).
- **Akses State Global:**
    - Status favorit resep (isFavorite) ditentukan dengan ref.watch(favoritesProvider).
    - Saat tombol favorit ditekan, status diperbarui menggunakan ref.read(favoritesProvider.notifier).state = newFavorites;.

2. Implementasi Animasi Interaktif (Press Effect)

Untuk meningkatkan pengalaman pengguna (UX), *widget* ini mengimplementasikan efek "tekan" (*press effect*) yang menghasilkan *feedback* visual saat disentuh, membuatnya terasa seperti tombol.

Komponen Animasi	**Fungsi**
SingleTickerProviderStateMixin	Diimplementasikan pada _RecipeListItemState untuk menyediakan *ticker* bagi AnimationController.
AnimationController _controller	Durasi singkat 150ms untuk mengontrol kecepatan efek tekan.
Tween<double> (0.97)	Animasi skala yang menyusutkan *widget* menjadi **97%** dari ukuran aslinya.
ScaleTransition	Menggunakan _scaleAnimation untuk membungkus seluruh konten daftar, menerapkan efek skala.

GestureDetector	Mengikat animasi ke interaksi sentuh: onTapDown memicu _controller.forward() (menyusut), dan onTapUp / onTapCancel memicu _controller.reverse() (kembali ke ukuran normal).

3. Fungsionalitas Utama (Navigasi)

Ketika item daftar diketuk (onTap), fungsionalitas utama aplikasi diaktifkan:

1. Animasi skala dipulihkan (_controller.reverse()) untuk memastikan item kembali ke ukuran normal.
2. Fungsi showModalBottomSheet dipanggil.
3. **RecipeDetailSheet** (dari file sebelumnya) dimuat sebagai konten *bottom sheet* dengan properti:
    - isScrollControlled: true: Memungkinkan *sheet* untuk mengambil tinggi layar yang lebih besar (dikelola oleh DraggableScrollableSheet di dalamnya).
    - backgroundColor: Colors.transparent: Agar sudut melengkung pada RecipeDetailSheet dapat terlihat.

4. Struktur Visual dan Desain Item

Konten *list item* disusun dalam sebuah **Row** dengan estetika **"card"** minimalis:

1. **Wadah (Container):** Diberi padding: 12, borderRadius: 12, dan **boxShadow** yang tipis untuk memberikan kesan elevasi atau "mengambang".
2. **Emoji/Image:** Resep ditampilkan di sisi kiri dengan ukuran besar (fontSize: 30).
3. **Informasi Resep (Expanded Column):**
    - **Nama Resep:** Ditampilkan dengan *bold* dan dibatasi satu baris (maxLines: 1) dengan *ellipsis* jika terlalu panjang.
    - **Waktu Memasak (cookTime):** Ditampilkan di bawah nama dengan ikon jam (Icons.access_time) dan teks berwarna abu-abu, memberikan informasi ringkas yang penting.

4. **Tombol Favorit:** Ikon hati di sisi kanan. Warna ikon (merah atau abu-abu) secara dinamis mencerminkan status favorit resep (isFavorite). Menekan tombol ini memicu pembaruan langsung pada **favoritesProvider** Riverpod.

2.10 Alur Kerja Aplikasi (Flow Program)

Aplikasi Catalog Resep Makanan memiliki alur kerja yang sistematis dan sederhana sehingga mudah dipahami oleh pengguna maupun pengembang. Alur kerja aplikasi berikut memetakan perjalanan pengguna dari awal menjalankan aplikasi hingga melihat detail resep dan kembali ke daftar utama.

1. **User membuka aplikasi.**
   Proses dimulai dari file main.dart yang menjalankan aplikasi Flutter dengan memanggil fungsi MaterialApp(). Di sini juga ditentukan halaman awal yaitu HomePage().
   dart

```
void main() {

  runApp(MyApp());

}

MaterialApp(

  home: HomePage(),

);
```

2. **Home Page menampilkan daftar resep.**
   Pada Home Page, aplikasi mengambil data resep dari file recipe_data.dart. Data tersebut digunakan untuk membangun widget daftar resep (ListView atau GridView) agar seluruh koleksi resep tampil di halaman utama.
   dart

```
final recipeList = RecipeData.getAll();
```

```
ListView.builder(
  itemCount: recipeList.length,
  itemBuilder: (context, index) {
    return RecipeCard(recipeList[index]);
  },
);
```

3. **User menekan salah satu resep.**
   Saat pengguna mengetuk salah satu item resep pada daftar, aplikasi menggunakan Navigator.push untuk berpindah ke halaman detail resep.
   dart

```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => DetailPage(recipe)),
);
```

4. **Detail Page muncul.**
   Halaman Detail Page akan menampilkan informasi lengkap resep yang dipilih, mencakup gambar makanan, judul resep, daftar bahan-bahan, serta langkah-langkah memasak, semuanya tersaji dalam satu halaman yang dapat discroll.
   dart

```
Image.asset(recipe.image)
Text(recipe.title)
Column(children: recipe.ingredients.map((ing) => Text("• $ing")).toList(),)
```

Column(children: List.generate(recipe.steps.length, (i) { return Text("${i+1}. ${recipe.steps[i]}"); }))

5. **User kembali ke Home Page.**
   Pengguna dapat kembali ke daftar resep dengan menekan tombol back yang tersedia di AppBar atau tombol back pada device, sehingga alur navigasi tetap lancar dan intuitif.

Dengan alur kerja yang jelas dan logis ini, aplikasi memastikan pengalaman pengguna yang sederhana namun efisien, mulai dari eksplorasi resep hingga mempelajari cara memasak menu yang diinginkan.

2.11 UI/UX Aplikasi

Antarmuka pengguna (UI) aplikasi Catalog Resep Makanan dirancang dengan prinsip kesederhanaan dan kemudahan penggunaan sebagai prioritas utama. Seluruh tampilan menggunakan komponen-komponen dasar Flutter yang telah terbukti efektif dalam menciptakan pengalaman pengguna yang baik.

Komponen-Komponen UI yang Digunakan:

- **Card** adalah widget yang digunakan untuk menampilkan setiap resep dalam bentuk container dengan elevation dan shadow yang memberikan depth pada UI. Setiap Card berisi informasi ringkas resep seperti gambar dan judul, menciptakan visual separation yang jelas antar item resep.
- **ListTile** adalah widget yang menyediakan layout standar untuk item list dengan support untuk leading, title, subtitle, dan trailing elements. Widget ini sangat berguna untuk menampilkan informasi terstruktur dalam format yang konsisten dan mudah dibaca.
- **Image.asset** adalah widget untuk menampilkan gambar-gambar makanan yang tersimpan dalam folder assets/images. Widget ini memastikan gambar dimuat dengan efisien dan dapat disesuaikan ukurannya sesuai kebutuhan layout.
- **Column dan Row** adalah widget fundamental untuk mengatur widget secara vertikal (Column) dan horizontal (Row). Kedua widget ini digunakan secara ekstensif untuk membangun layout yang kompleks dengan cara menggabungkan widget secara modular dan terstruktur.

- **ListView** adalah widget scrollable yang menampilkan daftar item secara efisien dengan support untuk lazy loading. Widget ini ideal untuk menampilkan koleksi resep dalam jumlah besar tanpa mengorbankan performa aplikasi.

Palet Warna dan Styling:

Warna-warna yang digunakan mengikuti default Material Design, sehingga aplikasi tampil clean, minimalis, dan tetap menarik secara visual tanpa mengorbankan keterbacaan maupun aksesibilitas. Material Design color palette memberikan konsistensi visual yang dikenali pengguna dan memudahkan navigasi aplikasi.

- Desain Clean dan Cocok untuk Pemula:

  Desain aplikasi dirancang dengan prinsip "less is more", menghindari kompleksitas UI yang tidak perlu, sehingga aplikasi cocok sebagai referensi pembelajaran bagi pemula yang ingin mempelajari Flutter. Setiap elemen UI memiliki tujuan yang jelas dan berkontribusi pada fungsionalitas aplikasi tanpa menambah kompleksitas yang tidak perlu.

- Fitur-Fitur Utama Aplikasi:

  Aplikasi Catalog Resep Makanan menyediakan daftar menu resep yang ditampilkan secara visual dan mudah dijelajahi, memungkinkan pengguna untuk dengan cepat menemukan resep yang ingin mereka pelajari. Setiap resep dapat diakses untuk melihat detail resep lengkap yang mencakup gambar makanan berkualitas, daftar bahan-bahan lengkap dengan takaran, dan langkah-langkah pembuatan yang terurai secara jelas dan terstruktur.

- Navigasi antar halaman diimplementasikan dengan menggunakan Navigator Flutter yang memberikan transisi mulus antar screen. Pengguna dapat dengan mudah berpindah dari halaman daftar resep ke halaman detail, serta kembali dengan tombol back yang intuitif.
- Aplikasi menggunakan model dan data dummy yang memudahkan proses pengembangan dan testing tanpa perlu koneksi database eksternal. Dengan pendekatan ini, aplikasi tetap sederhana namun tetap dapat mendemonstrasikan semua fungsionalitas dengan baik.

- Struktur proyek rapi dan siap dikembangkan, dengan organisasi file yang mengikuti best practice Flutter. Folder-folder terorganisir dengan baik (pages, models, data, widgets, assets), membuat kode mudah dimaintain dan siap untuk penambahan fitur lanjutan seperti database integration, authentication, atau API integration di masa depan.

Dengan pendekatan UI/UX seperti ini, aplikasi Catalog Resep Makanan tidak hanya fungsional dan mudah digunakan oleh berbagai kalangan pengguna, tetapi juga menjadi contoh excellent dari pengembangan aplikasi mobile Flutter dengan struktur proyek yang professional dan sustainable untuk pengembangan jangka panjang.

BAB 3
PENUTUP

3.1 Kesimpulan

Aplikasi Catalog Resep Makanan telah berhasil merancang antarmuka pengguna yang sederhana dan intuitif untuk menampilkan katalog resep dalam format yang mudah dibaca. Desain UI menggunakan komponen-komponen dasar Flutter seperti Card, ListView, dan Material Design color palette yang meningkatkan usability. Struktur visual yang jelas dengan penempatan strategis AppBar, daftar resep dalam format card yang terorganisir, dan penggunaan whitespace yang tepat memastikan pengguna dari berbagai latar belakang dapat memahami informasi resep tanpa kesulitan. Implementasi Material Design principles memberikan konsistensi visual yang intuitif, sehingga aplikasi ini accessible dan user-friendly untuk semua kalangan.

Halaman detail resep telah berhasil mengorganisasi informasi lengkap mencakup nama makanan, gambar, bahan-bahan, dan cara pembuatan dalam format yang terstruktur dan mudah diikuti. Informasi disajikan secara hierarkis: gambar di bagian atas sebagai visual centerpiece, nama resep, daftar bahan dengan bullet points yang jelas, dan langkah-langkah dengan nomor urut yang membimbing pengguna. Implementasi SingleChildScrollView memastikan semua informasi dapat diakses tanpa terpotong, memberikan pengalaman membaca yang nyaman dan optimal pada berbagai ukuran layar.

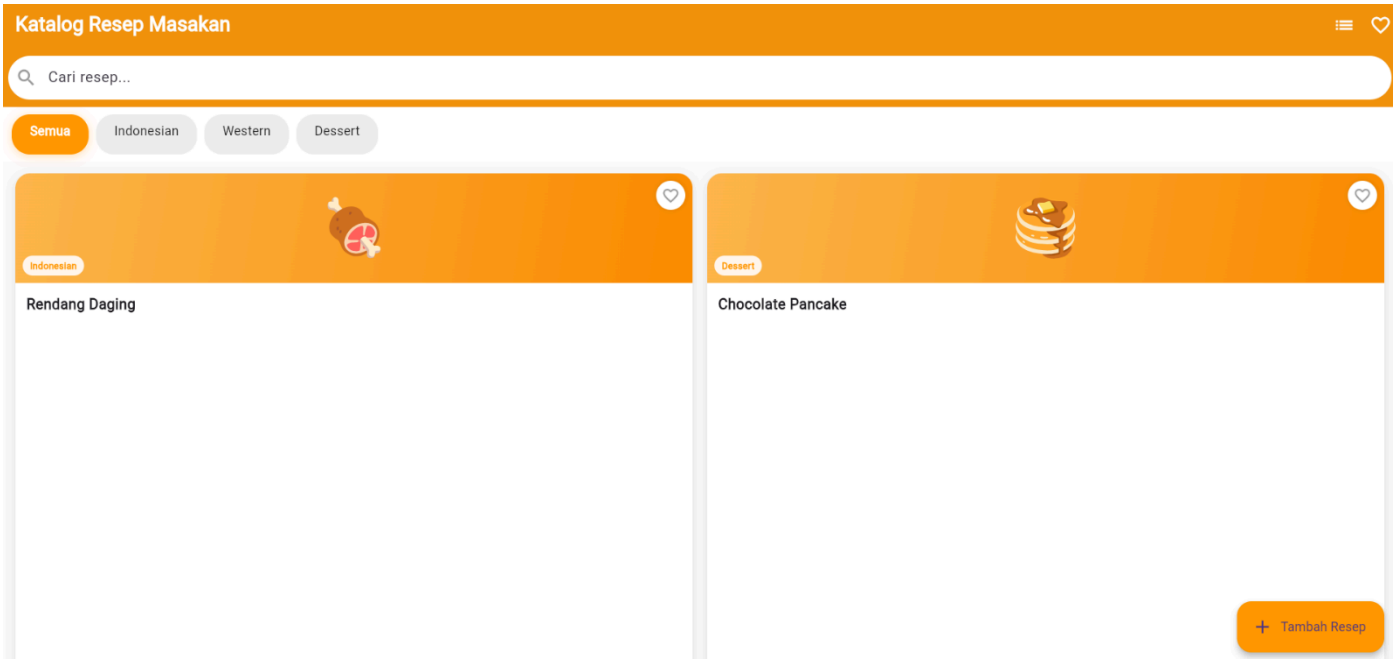
Sistem navigasi memungkinkan pengguna untuk dengan mudah menjelajahi daftar resep dan mengakses detail resep yang diinginkan. Pengguna dapat memilih resep dengan mengetuk item menggunakan GestureDetector, yang kemudian memicu Navigator.push untuk membuka halaman detail. Transisi antar halaman berlangsung mulus dengan visual feedback yang jelas, dan tombol back memudahkan pengguna untuk kembali ke halaman sebelumnya. Sistem navigasi yang responsif ini menciptakan pengalaman yang seamless dan predictable.

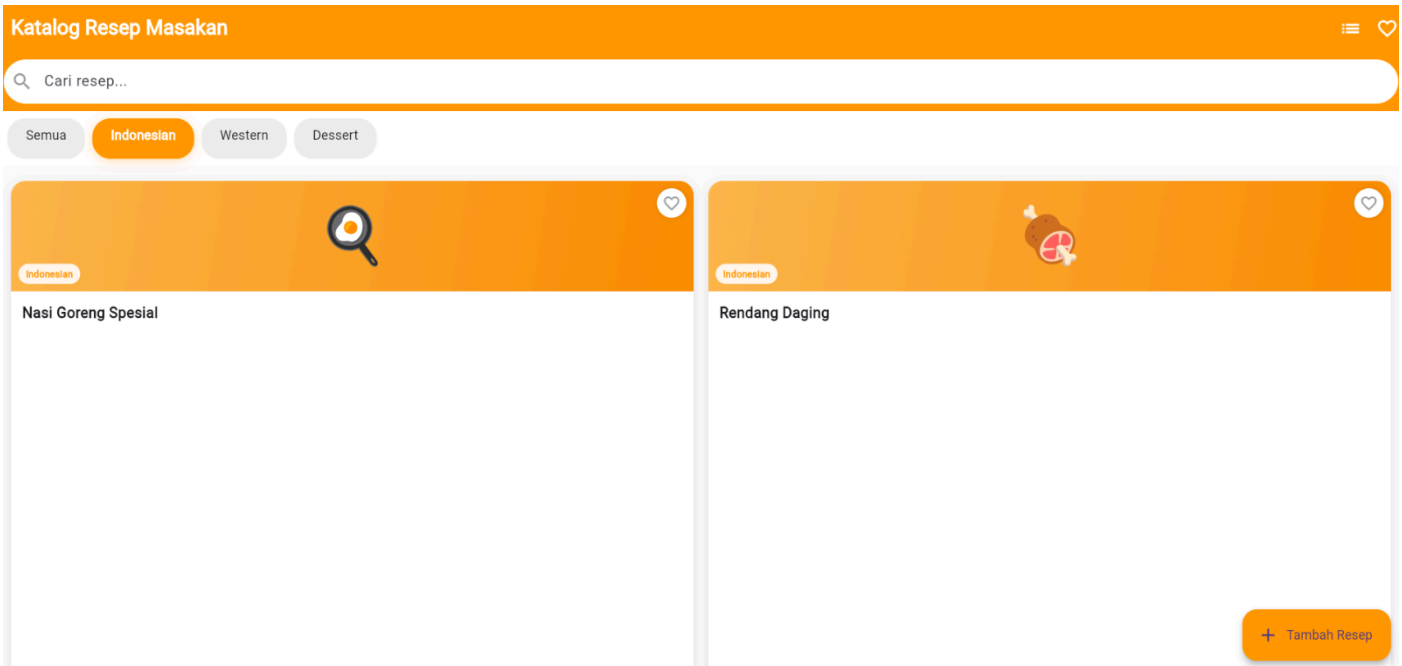
Data resep dikelola menggunakan class Recipe yang mendefinisikan properti-properti fundamental: title, image, ingredients, dan steps. Model data yang type-safe ini memastikan konsistensi informasi di seluruh aplikasi dan menghindari error akibat tipe data yang tidak

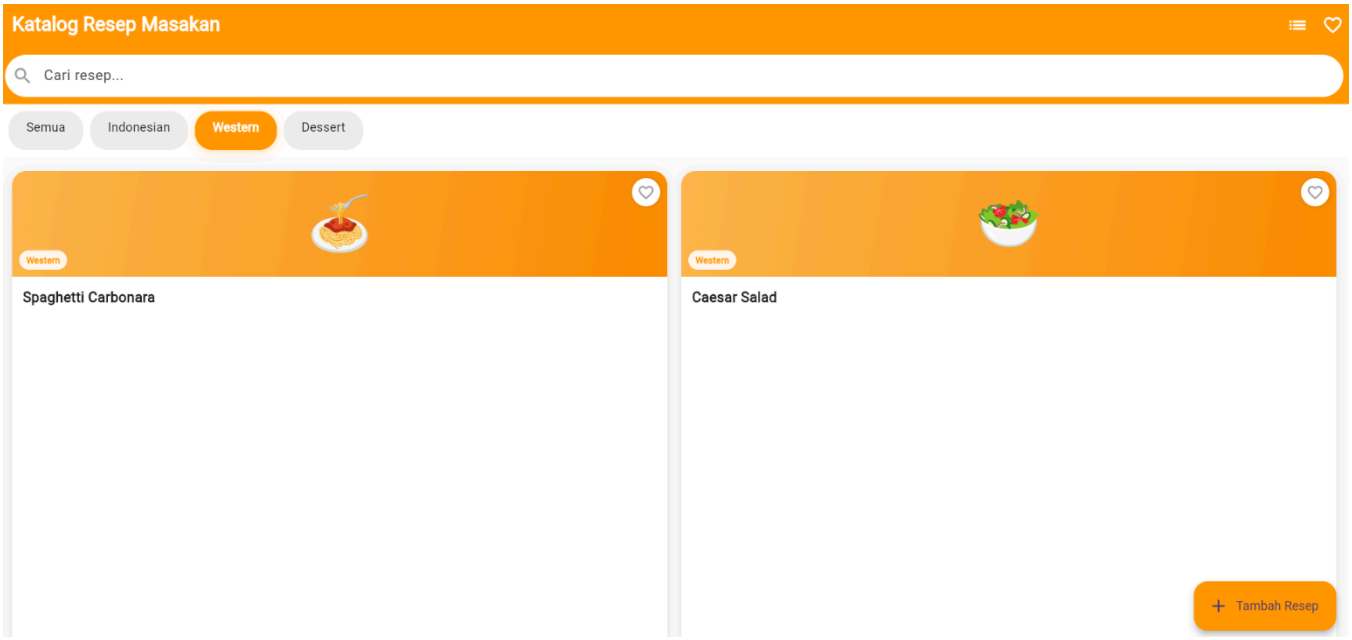
konsisten. Data resep disimpan dalam recipe_data.dart sebagai sumber data terpusat yang mudah diakses dan dikelola, memudahkan maintenance dan pengembangan aplikasi di masa depan.

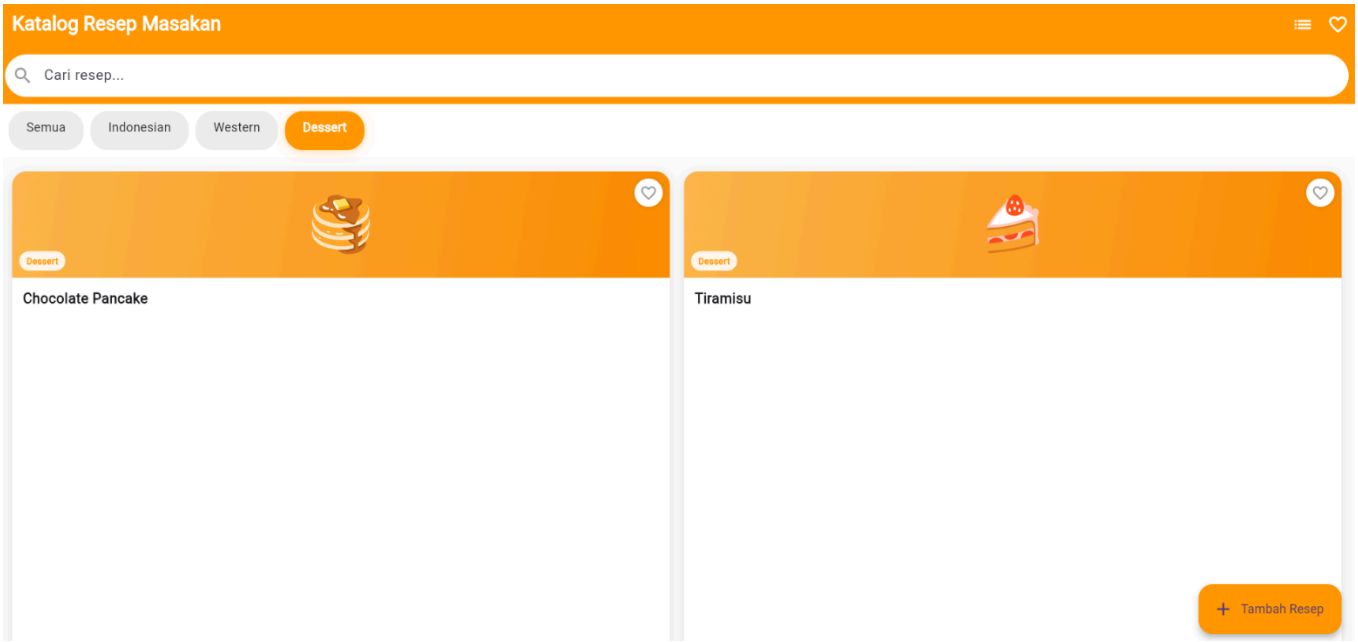
Aplikasi berfungsi optimal sebagai referensi sederhana bagi pengguna yang ingin memasak dengan menyediakan tata letak yang rapi, performa yang optimal, dan aksesibilitas yang baik. Desain visual yang clean menghindari clutter dan fokus pada konten utama. Implementasi ListView.builder dengan lazy loading memastikan performa tetap optimal meski jumlah resep bertambah. Struktur proyek Flutter yang lengkap dan termodular membuat aplikasi mudah dimaintain dan siap dikembangkan dengan fitur-fitur tambahan di masa depan.

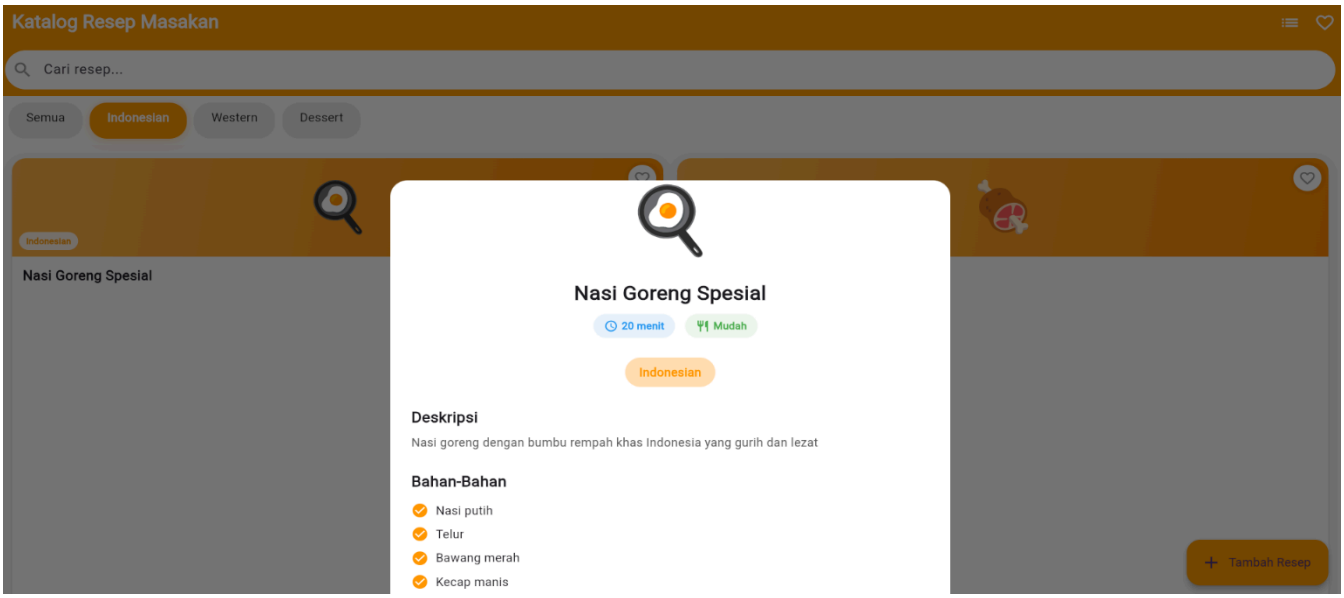
LAMPIRAN

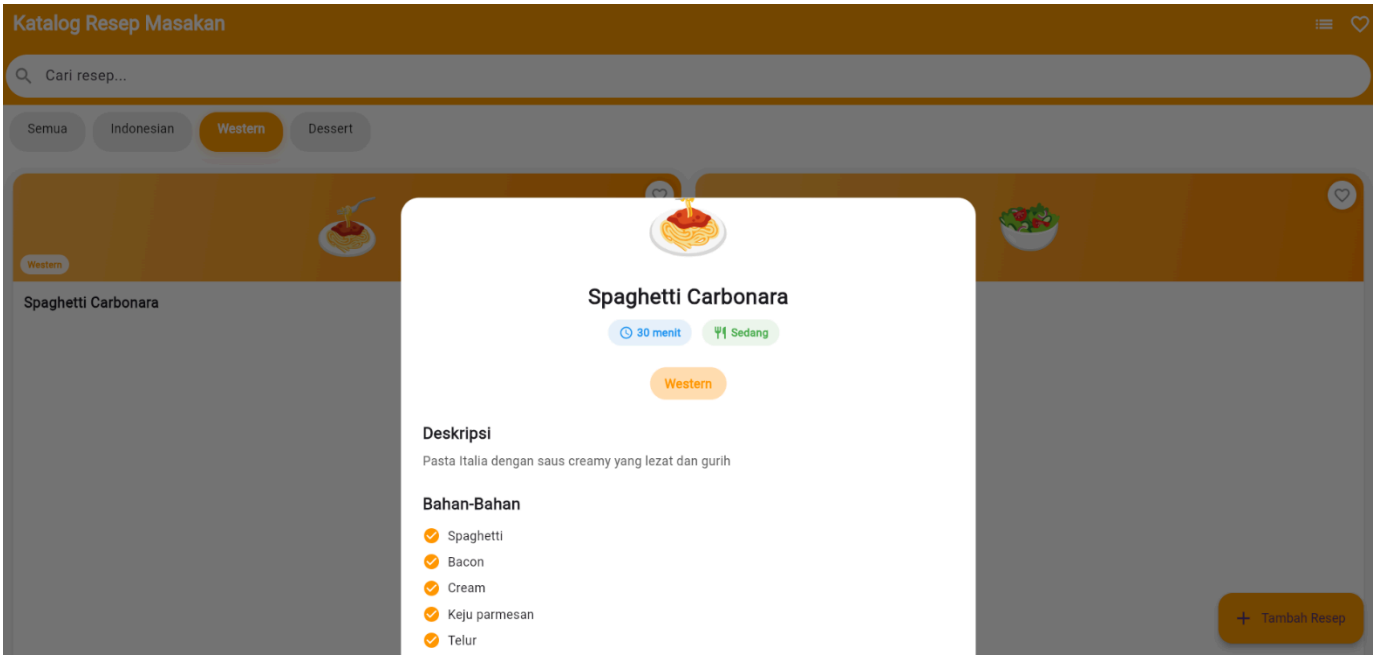

Katalog Resep Masakan
Cari resep...
Semua
Indonesian
Western
Dessert
Indonesian
Rendang Daging
Dessert
Chocolate Pancake
Tambah Resep

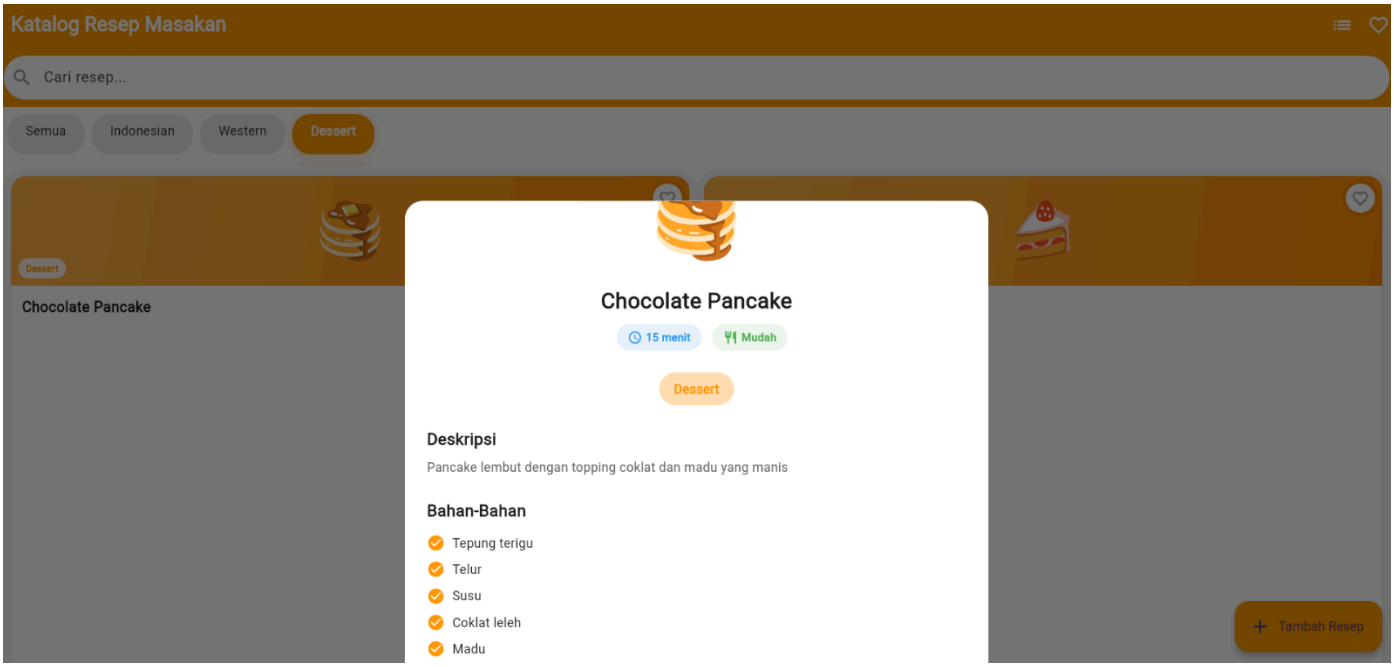

Katalog Resep Masakan
Cari resep...
Semua
Indonesian
Western
Dessert
Indonesian
Nasi Goreng Spesial
Indonesian
Rendang Daging
Tambah Resep

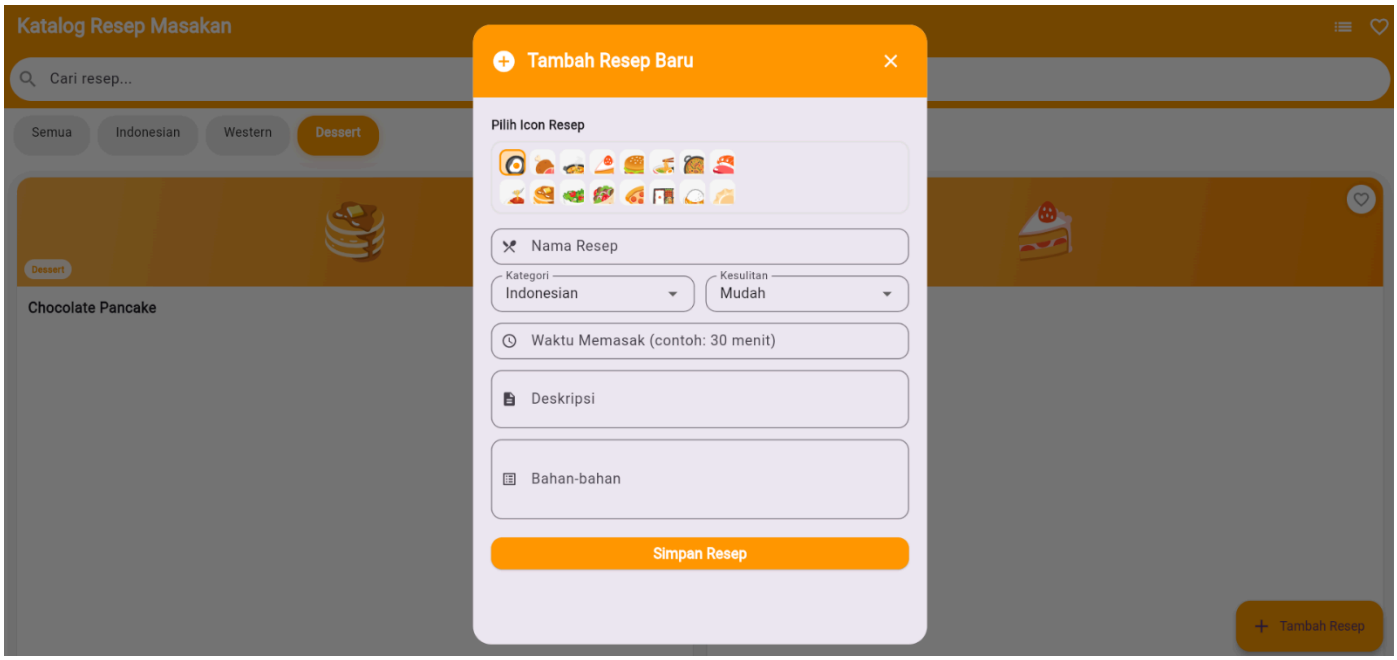

Katalog Resep Masakan
Cari resep...
Semua
Indonesian
Western
Dessert
Western
Spaghetti Carbonara
Western
Caesar Salad
Tambah Resep


Katalog Resep Masakan
Cari resep...
Semua
Indonesian
Western
Dessert
Dessert
Chocolate Pancake
Dessert
Tiramisu
Tambah Resep


Katalog Resep Masakan
Cari resep...
Semua
Indonesian
Western
Dessert
Indonesian
Nasi Goreng Spesial
Nasi Goreng Spesial
20 menit
Mudah
Indonesian
Deskripsi
Nasi goreng dengan bumbu rempah khas Indonesia yang gurih dan lezat
Bahan-Bahan
Nasi putih
Telur
Bawang merah
Kecap manis
Tambah Resep


Katalog Resep Masakan
Cari resep...
Semua
Indonesian
Western
Dessert
Western
Spaghetti Carbonara
Spaghetti Carbonara
30 menit
Sedang
Western
Deskripsi
Pasta Italia dengan saus creamy yang lezat dan gurih
Bahan-Bahan
Spaghetti
Bacon
Cream
Keju parmesan
Telur
Tambah Resep


Katalog Resep Masakan
Cari resep...
Semua
Indonesian
Western
Dessert
Chocolate Pancake
Dessert
Chocolate Pancake
15 menit
Mudah
Dessert
Deskripsi
Pancake lembut dengan topping coklat dan madu yang manis
Bahan-Bahan
Tepung terigu
Telur
Susu
Coklat leleh
Madu
Tambah Resep


Katalog Resep Masakan
Cari resep...
Semua
Indonesian
Western
Dessert
Chocolate Pancake
Dessert
Tambah Resep Baru
Pilih Icon Resep
Nama Resep
Kategori
Indonesian
Kesulitan
Mudah
Waktu Memasak (contoh: 30 menit)
Deskripsi
Bahan-bahan
Simpan Resep
Tambah Resep