Hamdard University
Department of Computing
Final Year Project

FitFeast: Personalized Nutrition and Meal Planning System.

FYP-016/FL24

Software Design Specifications

Submitted by
Rabail Arshad 1397-2021
Syeda Sara Shehnaz (2230-2021)
Syeda Maryam Shehnaz (2231-2021)

Supervisor Name:
Engr. Farooq Iqbal

Spring 2024

Document Sign-off Sheet

Document Information

Project Title	FitFeast: Personalized Nutrition and Meal Planning System
Project Code	FYP-016/FL24
Document Name	Software Requirements Specifications
Document Version	<1.0>
Document Identifier	Software Design Specifications
Document Status	Draft
Author(s)	Rabail, Sara, Maryam
Approver(s)	Engr. Farooq Iqbal
Issue Date	<Date of issuance of this document>

Name	Role	Signature	Date
Rabail Arshad	Project Manager		
Syeda Sara Shehnaz	Developer		
Syeda Maryam Shehnaz	UI/UX Designer		
Engr. Farooq Iqbal	Supervisor		
	Co-Supervisor		
	Project Coordinator		

Revision History

Date	Version	Description	Author
04/12/2024	V-1	Initial SDS discussed	All Members
12/12/2024	V.1.1	SDS doc review & update	All Members
26/12/24	V.1.2	Component / Package diagrams	All Members
7/01/25	V.1.3	Finalized with all requirements	All Members

Definition of Terms, Acronyms, and Abbreviations

Term/Acronym	Description
SDS	Software Design Specification. A document outlining the architectural and design aspects of FitFeast.
SRS	Software Requirements Specification. A document defining the functional and non-functional requirements.
UI	User Interface. The layer through which users interact with the FitFeast application.
API	Application Programming Interface. A set of functions and procedures enabling interaction between different software systems.
ML	Machine Learning. An AI approach is used to analyze food combinations, make recommendations, and personalize user experiences.
Flutter	A UI development framework used to create cross-platform mobile applications for iOS and Android.
AWS	Amazon Web Services. A cloud platform used for hosting FitFeast's backend and database components.
SQL Server	Structured Query Language Server. A relational database management system is used to store user and app data.
Nutritionix	A third-party API providing nutritional data for food and meals.
Edamam	A food and recipe API that integrates with FitFeast to fetch caloric and nutritional information.
OAuth 2.0	An industry-standard protocol for user authentication and authorization, ensuring secure logins.
Calorie Tracker	A module within FitFeast that calculates, tracks, and monitors user calorie intake.
BMR	Basal Metabolic Rate. The number of calories required to keep the body functioning at rest is used for calorie goal calculations.

Term/Acronym	Description
MealKit	A customizable collection of ingredients and recipes tailored to user dietary preferences.
Stripe	A payment gateway integrated for secure online transactions.
Firebase	A cloud platform by Google for sending notifications and managing real-time updates in the app.
TensorFlow	A machine learning framework used in FitFeast for recommendation algorithms and compatibility checks.
PyTorch	Another machine learning framework for implementing food compatibility and calorie prediction models.
JSON	JavaScript Object Notation - A lightweight data-interchange format used for storing preferences, user inputs, and API responses.
ACID	Atomicity, Consistency, Isolation, Durability - Principles ensuring reliable database transactions.
Figma	A design tool used to create mock screens and UI prototypes for the FitFeast application.
CRUD Operations	Basic database operations enable users to create, view, update, and delete data entries.
AI	Artificial Intelligence - Technology that powers recommendation features and ingredient compatibility checks in FitFeast.
VR	Virtual Reality - Excluded feature for immersive user interaction not included in FitFeast's current scope.

1 Table of Contents

2 Introduction

2.1 Purpose of Document

This Software Design Specification (SDS) document serves as a blueprint for the design and implementation of the FitFeast application. It is intended to outline the structural and object-oriented design methodologies employed, providing developers, stakeholders, and testers with a clear understanding of the system's architecture, functions, and implementation strategy. This document ensures consistency in development and serves as a reference throughout the software lifecycle.

2.2 Intended Audience

This document is targeted at:

- **Developers**: To guide implementation based on detailed designs.
- **Project Managers**: For understanding project scope and milestones.
- **Stakeholders**: To provide insights into system capabilities and design approaches.
- **Testers**: To validate system functionalities against the design.
- **Maintenance Teams**: For future enhancements and debugging.

2.3 Document Convention

Font Style: Times New Roman
Font Size: 12pt for body text, 18pt bold for headings.
Diagrams: Unified Modeling Language (UML) notations for all architectural and process diagrams. Made on Draw

2.4 Project Overview

The FitFeast application is a user-centric food planning platform that caters to personalized meal kits and calorie management needs. Using object-oriented design principles, the system integrates advanced machine learning to avoid incompatible ingredient combinations, track caloric intake, and provide meal recommendations. Its layered architecture ensures modularity, scalability, and ease of future enhancements.

2.5 Scope

Included Features:

- Personalized meal kit customization with dietary filters (e.g., vegan, keto).
- Calorie calculation and tracking for each meal with visual progress indicators.
- Machine learning to avoid incompatible ingredient combinations.
- Food recommendations when calorie intake exceeds the set threshold.
- User profile management, including dietary preferences and weight goals.

Excluded Features:

- Real-time kitchen views.
- Delivery logistics and tracking beyond integration with third-party services.
- Support for VR or AR-based interactions.

3 Design Considerations

3.1 Assumptions and Dependencies

Assumptions:

1. **User Competence**: Users are assumed to have a basic understanding of using smartphones and the FitFeast app. The app will be intuitive, but some technical understanding is required to manage calorie goals and customize meal plans.
2. **Consistent Internet Access**: FitFeast requires continuous internet connectivity for real-time calorie tracking, API access for nutrition data, and meal recommendations. A stable internet connection is crucial for uninterrupted app performance.
3. **API Integration**: The system depends on third-party APIs (e.g., Nutritionix for calorie data) and machine learning models for feature functions like meal compatibility analysis and recommendations. These dependencies must remain functional and updated throughout the app's lifecycle.
4. **User Data Accuracy**: Users' dietary preferences, weight goals, and activity levels are assumed to be accurate when entered, as the system uses this data to generate personalized recommendations.
5. **Scalability**: The system will be designed to scale for growing user bases, ensuring that performance remains consistent even during peak traffic times.

Dependencies:

1. **Third-Party Services**: The app depends on external services, such as cloud hosting (AWS, Azure), payment gateways, and nutrition data APIs, for core functionality. The availability of these services is critical to the app’s success.
2. **Machine Learning Models**: The system's ability to suggest personalized meal plans and avoid incompatible combinations depends on the machine learning models used. These models must be trained with sufficient data and continuously updated to maintain accuracy.
3. **Device Compatibility**: The app must be compatible across both iOS and Android platforms, requiring ongoing updates to ensure it works smoothly on a variety of devices.
4. **Regulatory Compliance**: The system may be subject to evolving regulations related to health, nutrition, or data protection. The design should allow easy updates to ensure compliance with any changes in laws or standards.

3.2 Risks and Volatile Areas

Risks:

1. **Dataset Limitations**: Insufficient or biased data may result in inaccurate calorie counts or incompatible combination warnings.
2. **Evolving User Preferences**: User dietary trends (e.g., new popular diets) may necessitate frequent updates.
3. **Third-Party API Downtime**: API interruptions could affect features like calorie calculation and meal recommendations.
4. **Scalability Challenges**: A high user load may impact system performance if not designed for scalability.
5. **Regulatory Changes**: Compliance with local dietary labeling or nutritional data regulations may require system adjustments.

Mitigation Strategies:

1. Incorporate regular updates and retraining for machine learning models with fresh datasets.
2. Design modular systems to easily integrate new dietary preferences or features.
3. Ensure robust fallbacks for API failures, such as local caching of critical data.
4. Employ scalable cloud infrastructure with load balancing to handle peak traffic.
5. Regularly review and update system components to align with new regulations or standards.

4 System Architecture

The system architecture for **FitFeast** is designed to ensure seamless integration of its components, enabling smooth functionality and robust performance. It is partitioned into the following subsystems:

- ## User Interface Layer
    - **Purpose:** This layer is responsible for user interactions. It handles inputs and displays outputs in an intuitive, mobile-friendly design.
    - **Components:**
        - Mobile apps for iOS and Android developed using **Flutter**.
        - User-facing features such as meal customization, calorie tracking, and profile management.
- ## Application/Service Layer
    - **Purpose:** Acts as the business logic layer that processes user inputs and implements application rules.
    - **Components:**
        - **Backend Frameworks:** Developed using Python (Django/Flask) for RESTful API management.
        - **Meal Personalization Module:** Processes user preferences to suggest meal plans.
        - **Calorie Calculation Module:** Retrieves nutritional data to compute meal calories in real-time.
- ## Database Layer
    - **Purpose:** Responsible for storing, retrieving, and managing user and application data.
    - **Components:**
        - **Database Management System:** SQL Server for storing structured data such as user profiles, meal plans, and historical calorie records.
        - **Data Integration Layer:** Links third-party APIs (Nutritionix, Edamam) to fetch nutritional information.
- ## Machine Learning Subsystem
    - **Purpose:** Provides intelligent features like avoiding incompatible combinations and personalized recommendations.
    - **Components:**
        - **ML Models:** Built with TensorFlow/PyTorch for ingredient compatibility checks and caloric suggestions.
        - **Training Dataset:** Includes food pairings, user feedback, and caloric benchmarks.

- **Communication Layer**

  - **Purpose:** Facilitates interactions between system layers and third-party services.
  - **Components:**

    - **APIs:** Integrate with external services for payment processing, food data, and notifications.
    - **Real-time Communication Protocols:** Ensures fast and accurate updates for user modifications.

System Flow Summary

- ✓ **User Interaction:** Users interact with the app to set calorie goals, customize meals, or view recommendations.
- ✓ **Backend Processing:** The input is processed in the application layer, leveraging ML for advanced functionalities.
- ✓ **Database Queries:** Necessary data is fetched from the database or external APIs.
- ✓ **Response Delivery:** Results are displayed to the user, including meal suggestions, calorie counts, and alerts.

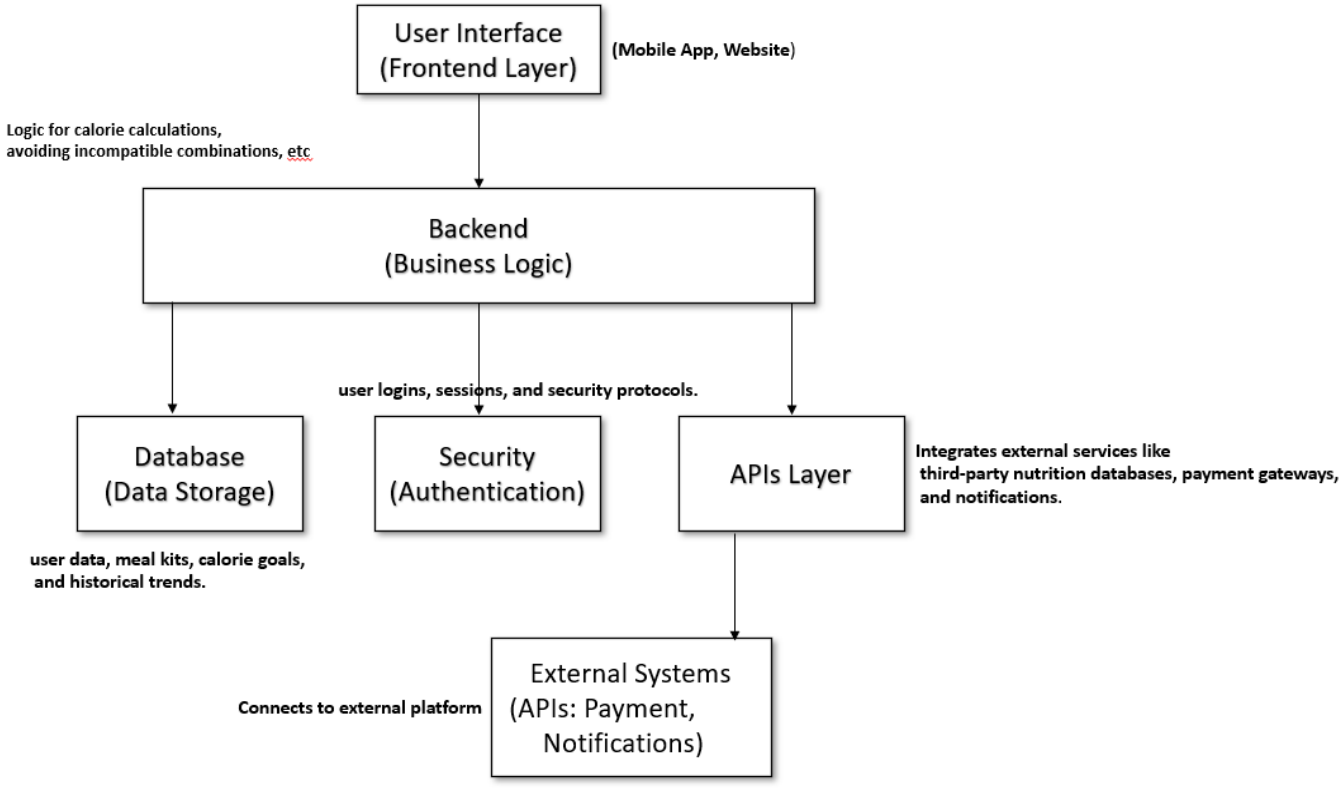


Figure 4.1 : System Flow Diagram

4.1 System Level Architecture

The **FitFeast** system architecture is designed to ensure modularity, scalability, and smooth interactions between its various components. The architecture focuses on decomposing the system into logical layers and subsystems, detailing relationships and external interfaces. Below is the top-level decomposition:

System Decomposition into Elements

- **User Interface (UI) Layer:**
  Handles user interactions through the mobile application interface. Developed using **Flutter**, this layer includes features like meal customization, calorie tracking, and notifications.
- **Application Layer:**
  Implements the business logic and processes requests from the UI. Built using Python frameworks like **Django** or **Flask**, it integrates APIs and ML algorithms.
- **Database Layer:**
  Manages data storage and retrieval. A **MySQL** or **SQL Server** database handles user profiles, dietary preferences, meal kits, and historical records.
- **Machine Learning (ML) Module:**
  Provides AI-driven features such as food recommendations, calorie adjustment, and avoiding incompatible combinations. Built using **TensorFlow** or **PyTorch**.
- **External Interface Layer:**
  Connects the system to third-party APIs like **Nutritionix** and **Edamam** for food data, and **payment gateways** for transaction management.

Relationships between Elements

- The **UI Layer** sends requests to the **Application Layer** via API calls.
- The **Application Layer** processes these requests, retrieves data from the **Database Layer**, and integrates results from the **ML Module** when necessary.
- The **Application Layer** communicates with external systems (e.g., APIs for food data) and sends the responses back to the **UI Layer**.

Interfaces to External Systems

- **Food Databases (Nutritionix, Edamam):** Provide nutritional and caloric data for ingredients and meals.
- **Payment Gateways (e.g., Stripe, PayPal):** Securely process transactions for meal orders.
- **Notification Services (e.g., Firebase):** Send real-time updates and alerts to users.

Major Physical Design Issues

- **Cloud Hosting:** The backend and database components will be hosted on platforms like **AWS** or **Google Cloud** to ensure high availability and scalability.
- **Mobile Execution:** The UI Layer will run on iOS and Android devices, optimized for responsiveness and performance.

Global Design Strategies

- **Error Handling:** Centralized error logging and retry mechanisms ensure smooth recovery from failures. For example, API timeouts are handled gracefully by providing cached data to the user.
- **Security Measures:** Data encryption, secure API keys, and user authentication (OAuth 2.0) are or will be employed to protect user information.
- **Scalability:** The system architecture supports scaling by separating components into microservices, allowing independent scaling of the UI, backend, and database.

UML Diagrams

- ## Component Diagram:

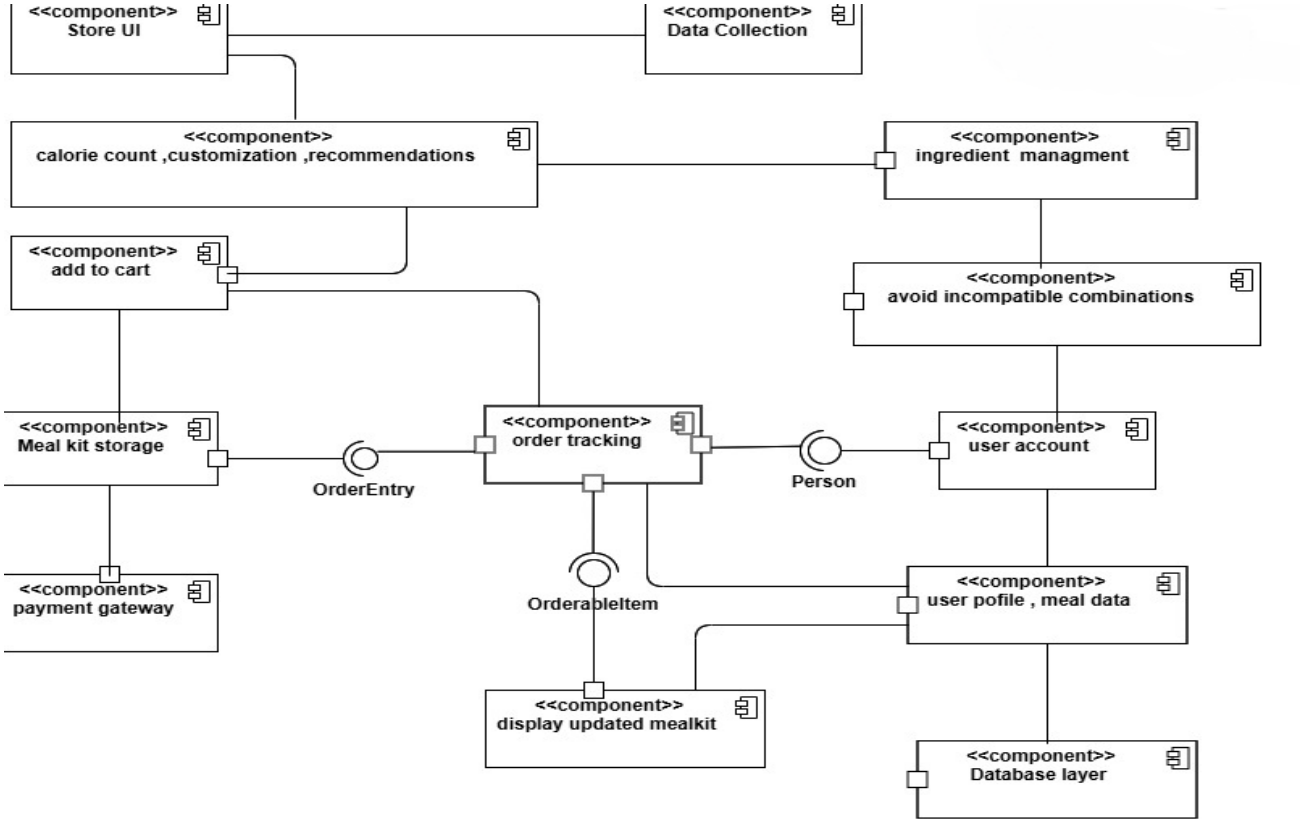


Figure 4.1.1 Component Diagram for FitFeast

- **Package Diagram:**

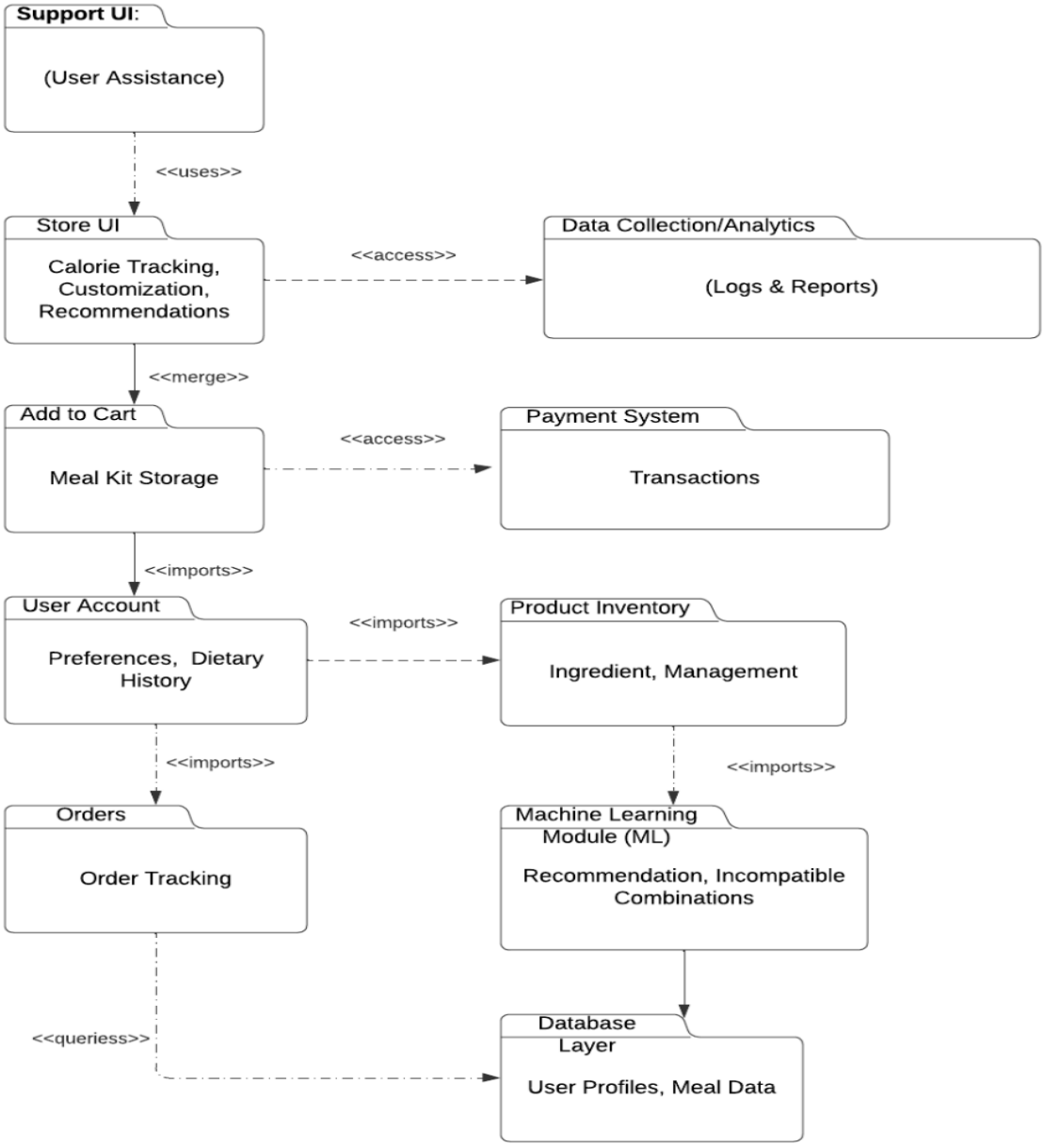


Figure 4.1.2 Package Diagram for FitFeast

4.2 Software Architecture

The **FitFeast** architecture is based on a three-tiered model to ensure modularity, scalability, and separation of concerns. Below is an explanation of the interaction between layers:

User Interface Layer

- **Purpose:** Handles user interactions and presents the results in a friendly manner.
- **Technologies:** Developed using **Flutter** for cross-platform mobile app development.
- **Responsibilities:**
    - Display meal options, calorie counts, and customization features.
    - Collect user inputs such as preferences, dietary restrictions, and portion adjustments.
    - Send user requests (e.g., calculate calorie intake) to the Middle Tier through API calls.

Middle Tier (Business Logic Layer)

- **Purpose:** Processes requests from the UI layer, manages the application logic, and interacts with external systems.
- **Technologies:** Implemented using Python frameworks like **Django** or **Flask**.
- **Responsibilities:**
    - Validate user inputs from the UI Layer.
    - Process calorie calculations and generate meal recommendations.
    - Communicate with the Machine Learning (ML) modules for advanced features like avoiding incompatible combinations.
    - Interact with external APIs (e.g., Nutritionix) to fetch real-time nutritional data.
    - Pass processed data to the Data Access Layer for storage or retrieval.

Data Access Layer (Database Layer)

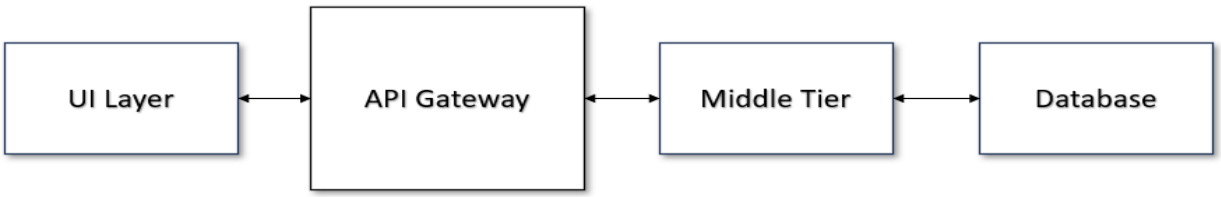
- **Purpose:** Manages data storage, retrieval, and updates.
- **Technologies:** Relational databases like **MySQL** or **SQL Server** are used.
- **Responsibilities:**
    - Store user profiles, dietary preferences, and order history.
    - Maintain tables for ingredients, nutritional values, and meal data.
    - Ensure secure and efficient data transactions with the Middle Tier

Flow of Interaction Between Layers

1. **User Interaction:**
   - Users select meal kits or input preferences through the UI.
2. **Request Handling:**
   - The UI Layer sends requests (e.g., calculate calories) to the Middle Tier via REST APIs.
3. **Business Logic Execution:**
   - The Middle Tier processes requests, performs calculations or communicates with the ML module.
4. **Data Handling:**
   - If required, the Middle Tier queries the Data Access Layer to retrieve or store relevant data.
5. **Response to User:**
   - The Middle Tier sends the processed data (e.g., calorie breakdown) back to the UI Layer, which displays the results to the user.

Diagram Representation

Layers Interaction Diagram:



In a diagram:

1. **User Layer**: Users interact through a Flutter mobile app.
2. **Middle Tier**: Python (Django/Flask) handles business logic, API calls, and ML computations.
3. **Database Layer**: SQL databases manage persistent data for user profiles, preferences, and meal plans.

This architecture ensures scalability, easy maintenance, and an intuitive user experience.

5 Design Strategy

Future System Extension or Enhancement

- **Design Decision:** Modular architecture allows easy addition of new features like AI-powered meal planning or integration with wearable devices.
- **Reasoning:** By separating concerns into distinct layers (UI, Middleware, Data), future modules like a recipe-sharing community or advanced analytics can be seamlessly integrated.
- **Trade-offs:** This may require slightly more initial setup effort to define clear module interfaces.

System Reuse

- **Design Decision:** Use reusable components like APIs for calorie calculations and ingredient compatibility.
- **Reasoning:** Third-party APIs (e.g., Nutritionix) reduce development time and ensure reliability.
- **Trade-offs:** Dependency on third-party systems may require adjustments if APIs are updated or deprecated.

User Interface Paradigms

- **Design Decision:** Intuitive, mobile-first design using Flutter for cross-platform compatibility.
- **Reasoning:** Provides users a seamless experience on both Android and iOS, enhancing accessibility.
- **Trade-offs:** This may require optimized resource management to ensure consistent performance on lower-end devices.

Data Management (Storage, Distribution, Persistence)

- **Design Decision:** Relational databases like MySQL ensure structured storage of user profiles, meal preferences, and calorie data.
- **Reasoning:** SQL databases provide robust querying capabilities and ensure ACID compliance for data consistency.
- **Trade-offs:** As the user base grows, scaling a relational database might require additional optimization or transitioning to hybrid solutions.

Concurrency and Synchronization

- **Design Decision:** Use RESTful APIs with token-based authentication for secure and synchronized user sessions.
- **Reasoning:** Supports simultaneous multi-device access, ensuring users' dietary and calorie data is always up-to-date.
- **Trade-offs:** Requires efficient server resource allocation to handle multiple concurrent API calls without performance degradation.

6 Detailed System Design

6.1 Design Class Diagram

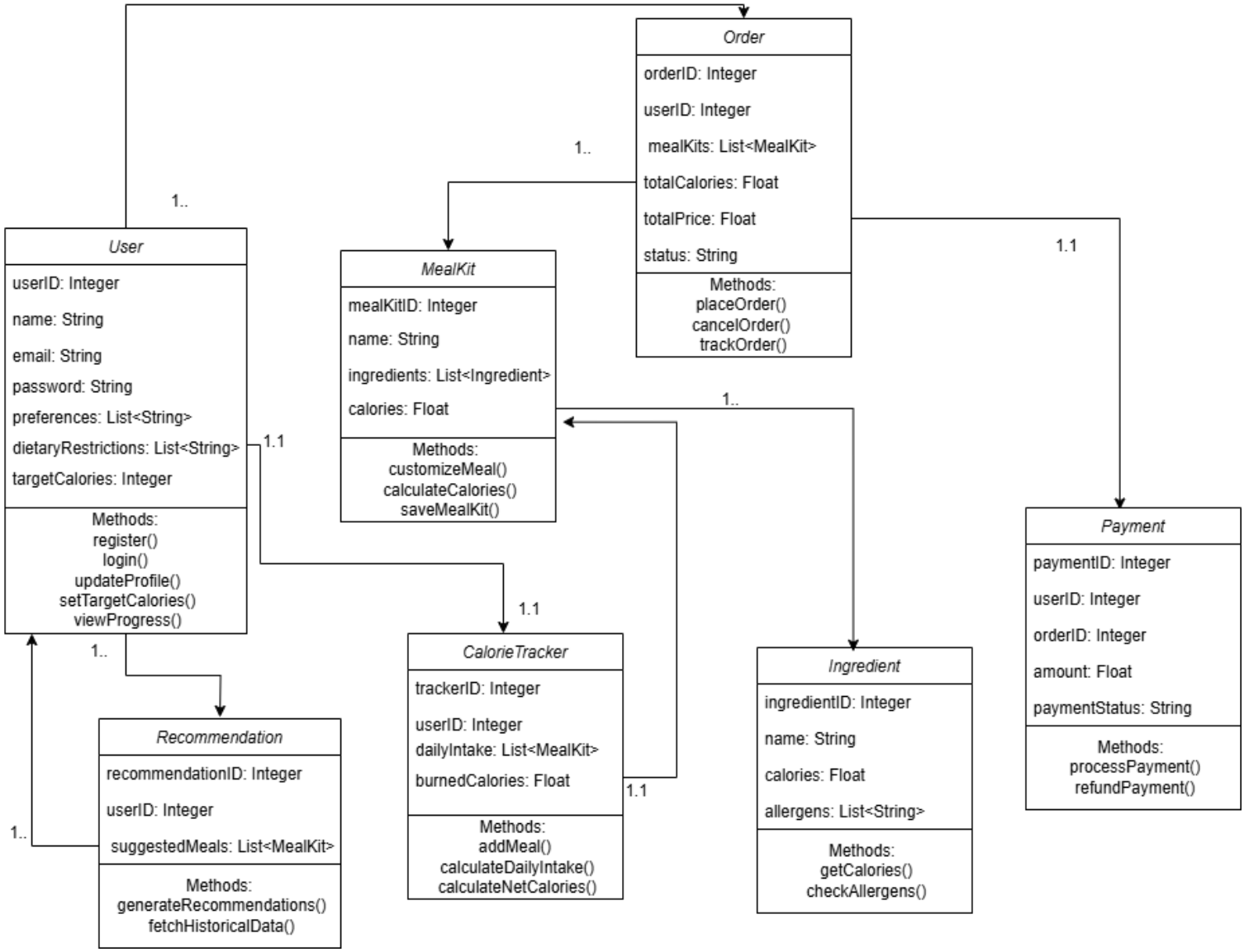


Figure 6.1 Design Class Diagram

6.2 Database Design

ER Diagram

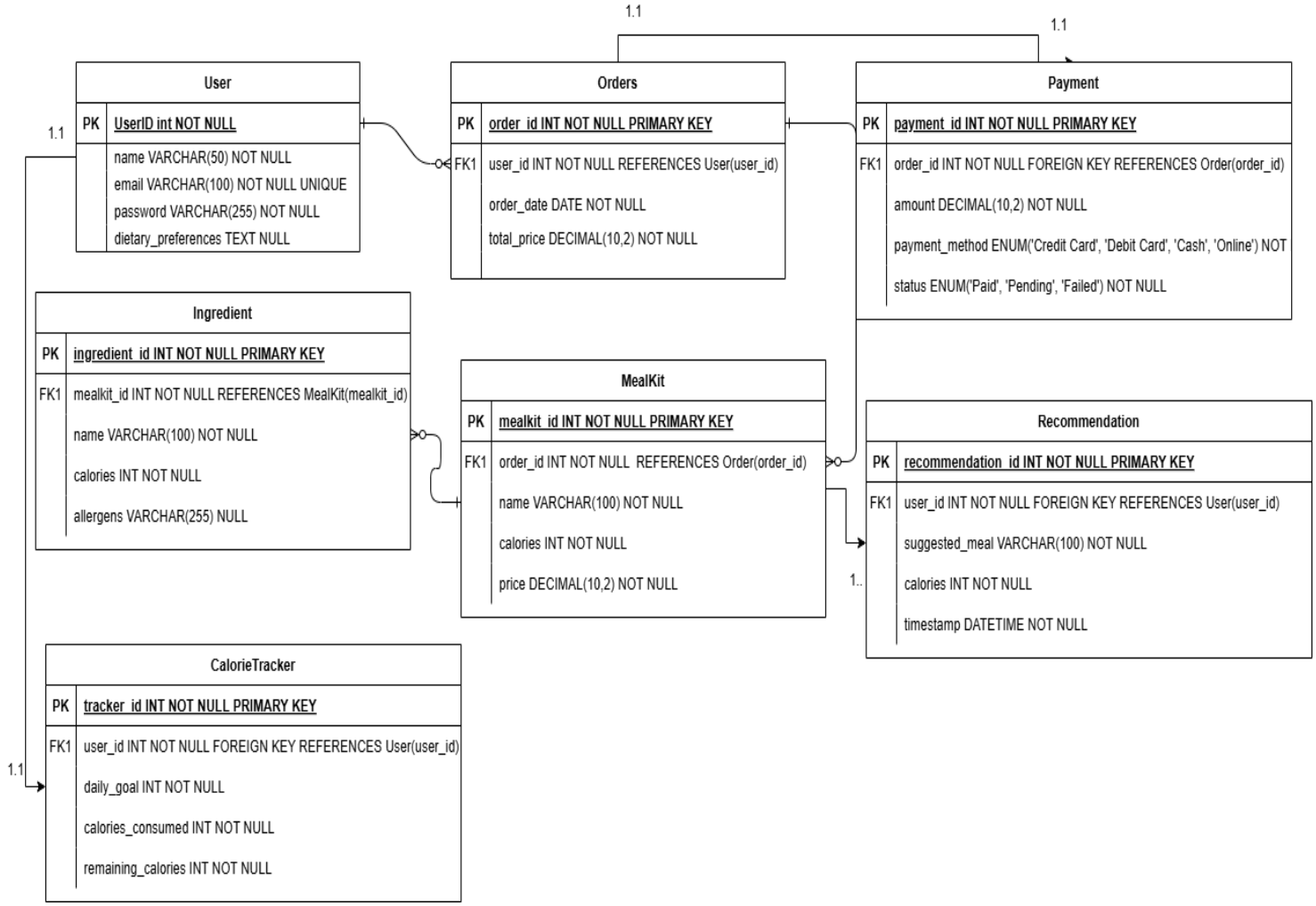


Figure 6.2 Database ER Diagram

6.2 Data Dictionary

6.2.1.1 Data 1: User Table

Column Name	Description,	Type,	Length,	Nullable,	Default Value,	Key Type
userID	Unique identifier for a user	INT	11	No	Auto Increment	PK
email	User's email address	VARCHAR	255	No	NULL	
calorieGoal	Daily calorie goal	INT	11	Yes	NULL	
preferences	Dietary preferences as JSON	TEXT	N/A	Yes	NULL	

6.2.1.2 Data 2 : MealKIT

Column Name,	Description,	Type,	Length,	Nullable,	Default Value,	Key Type
mealID	Unique identifier for a meal	INT	11	No	Auto Increment	PK
mealName	Name of the meal	VARCHAR	255	No	NULL	
calories	Total calorie count	INT	11	No	NULL	
price	Price of the meal	FLOAT	N/A	No	NULL	

6.2.1.3 Data 3: Order Table

Column Name,	Description,	Type,	Length,	Nullable,	Default Value,	Key Type
orderID	Unique identifier for an order	INT	11	No	Auto Increment	PK
userID	ID of the user placing the order	INT	11	No	NULL	FK
totalCost	Total cost of the order	FLOAT	N/A	No	NULL	
totalCalories	Total calorie count for order	INT	11	No	NULL	

6.2.1.4 Data 4: Preference Table

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
preferenceID	Unique identifier for a preference	INT	11	No	Auto Increment	PK
userID	The ID of the user	INT	11	No	NULL	FK
preferenceType	Type of dietary preference	VARCHAR	50	No	NULL	
value	Value of the preference (e.g., vegan)	VARCHAR	50	No	NULL	

6.2.1.5 Data 5: Feedback Table

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
feedbackID	Unique identifier for feedback	INT	11	No	Auto Increment	PK
userID	The ID of the user	INT	11	No	NULL	FK
feedbackText	User feedback	TEXT	N/A	Yes	NULL	
dateSubmitted	Date of feedback submission	DATETIME	N/A	No	CURRENT_DATE	

6.2.1.6 Data 6: Calorie Tracking Table

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
trackingID	Unique identifier for calorie record	INT	11	No	Auto Increment	PK
userID	The ID of the user	INT	11	No	NULL	FK
date	Date of calorie tracking	DATE	N/A	No	NULL	
caloriesConsumed	Total calories consumed	INT	N/A	No	NULL	
caloriesBurned	Total calories burned (exercise)	INT	N/A	Yes	NULL	

6.2.1.7 Data 7: Meal Recommendation Table

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
recommendationID	Unique identifier for recommendation	INT	11	No	Auto Increment	PK
userID	ID of the user	INT	11	No	NULL	FK
mealID	Recommended meal ID	INT	11	No	NULL	FK
recommendationReason	Reason for recommendation	VARCHAR	255	Yes	NULL	

6.2.1.8 Data 8: Allergen Table

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
allergenID	Unique identifier for allergen	INT	11	No	Auto Increment	PK
mealID	ID of the meal	INT	11	No	NULL	FK
allergenName	Name of the allergen	VARCHAR	100	No	NULL	

6.2.1.9 Data 9: Subscription Plan Table

Column Name	Description	Type	Length	Nullable	Default Value	Key Type
planID	Unique identifier for subscription	INT	11	No	Auto Increment	PK
userID	ID of the subscriber	INT	11	No	NULL	FK
planType	Type of plan (weekly, monthly)	VARCHAR	50	No	NULL	
startDate	Start date of subscription	DATE	N/A	No	NULL	
endDate	End date of subscription	DATE	N/A	Yes	NULL	

6.3 Application Design

6.3.1 Sequence Diagram

6.3.1.1 Sequence Diagram 1: User Login

Description: This sequence diagram illustrates how a user logs into the FitFeast application.
Parameter List:

- Input: Email/Password or Social Media Credentials
- Output: Authentication Token

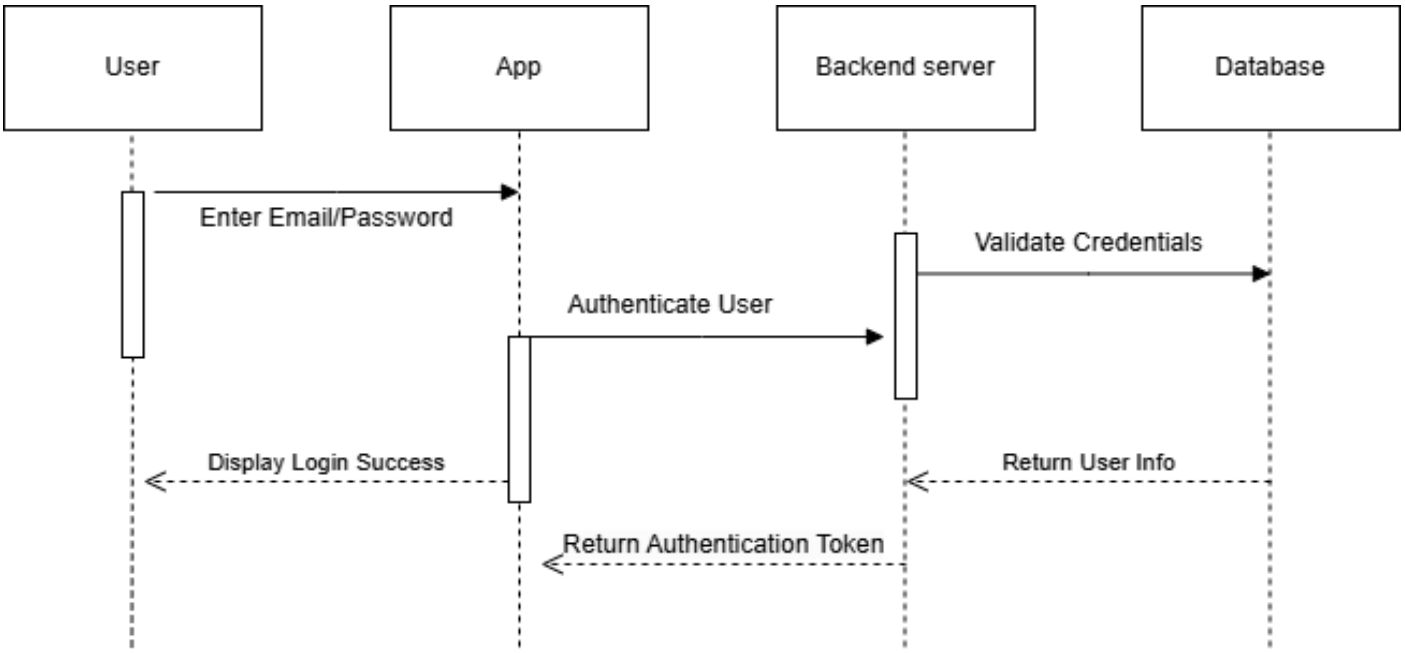


Figure 6.3.1.1 Sequence Diagram 1: User Login

6.3.1.2 Sequence Diagram 2: Meal Kit Customization

Description: This diagram explains how a user customizes a meal kit in the application.
Parameter List:

- Input: Selected Meal Kit, Added/Removed Ingredients
- Output: Updated Meal Kit Details

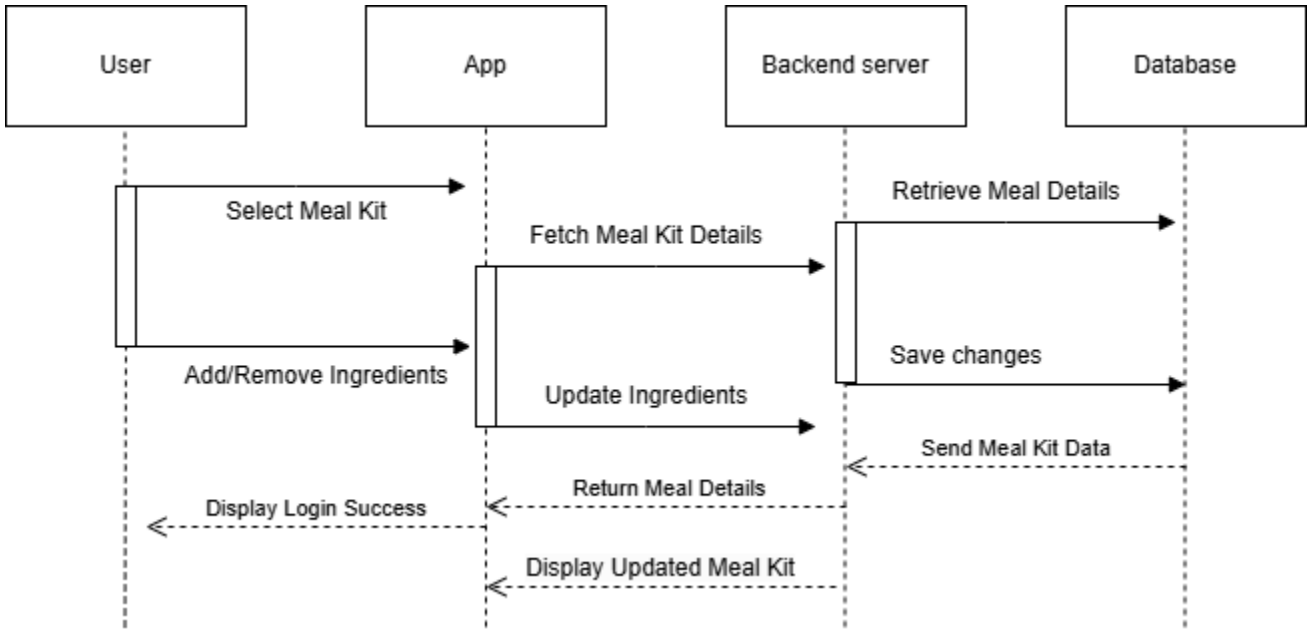


Figure 6.3.1.2 Sequence Diagram 2: Meal Kit Customization

6.3.1.3 Sequence Diagram 3: Calorie Calculation and Recommendation

Description: This diagram shows how the app calculates calories and provides recommendations.
Parameter List:

- Input: Selected Meal
- Output: Calorie Count, Food Recommendation

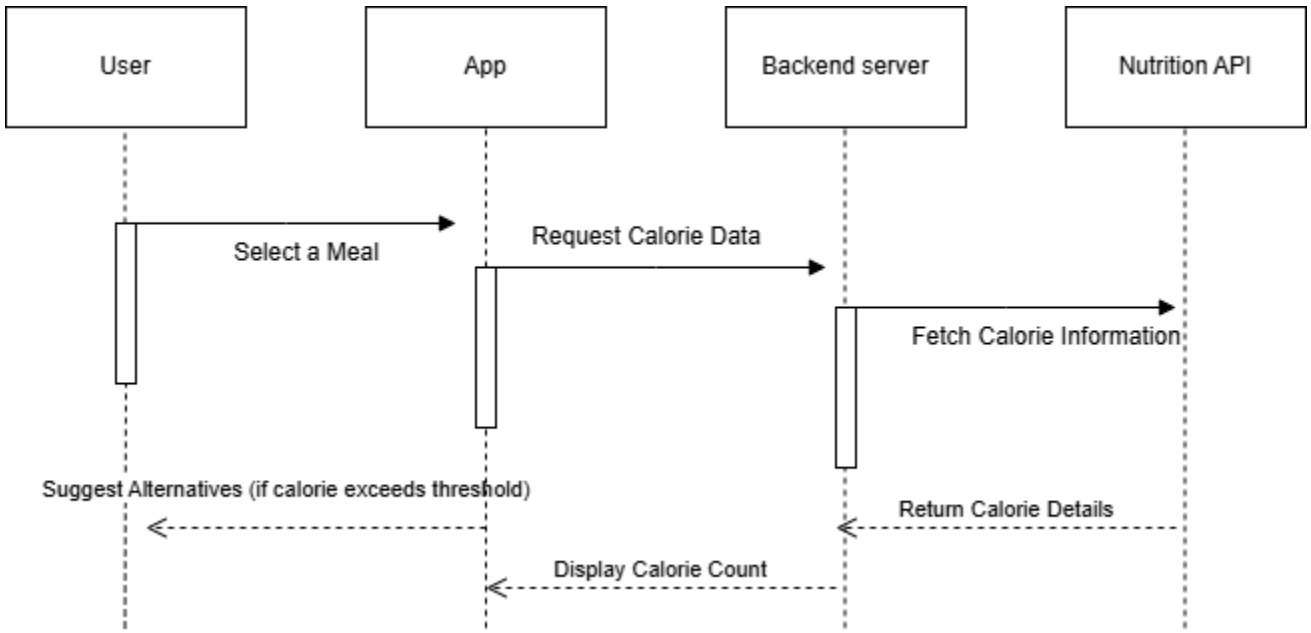


Figure 6.3.1.3 Sequence Diagram 3: Calorie Calculation and Recommendation

6.3.2 State Diagram For FitFeast

- **States:**
  - **Idle State**: The application is not in use (the user has not initiated any interaction).
  - **Login/Registration State**: The user logs in or registers to access their account.
  - **Main Menu State**: The user is presented with the main menu options (e.g., Browse Meals, View Recommendations, Track Calories).
  - **Meal Selection State**: The user browses and selects meals for an order.
  - **Customization State**: User customizes their meal kits or preferences (e.g., portion size, ingredients).
  - **Order Confirmation State**: The user confirms their selected meals and submits the order.
  - **Order Tracking State**: The user tracks the order delivery status.
  - **Calorie Tracking State**: The user checks daily calorie intake or progress.
  - **Error/Exception State**: An error occurs (e.g., login failure, unavailable meal item).

- **Transitions:**
  - **Idle → Login/Registration State**: User launches the app.
  - **Login/Registration State → Main Menu State**: Successful login or registration.
  - **Main Menu State → Meal Selection State**: User selects "Browse Meals."
  - **Meal Selection State → Customization State**: The user chooses a meal and opts to customize it.
  - **Customization State → Order Confirmation State**: The user finalizes the meal customization and proceeds to confirm the order.
  - **Order Confirmation State → Order Tracking State**: The user submits the order and transitions to order tracking.
  - **Main Menu State → Calorie Tracking State**: The user selects "Track Calories" from the main menu.
  - **Any State → Error/Exception State**: System encounters an error.

  - **Error/Exception State → Main Menu State**: User resolves the error or retries.

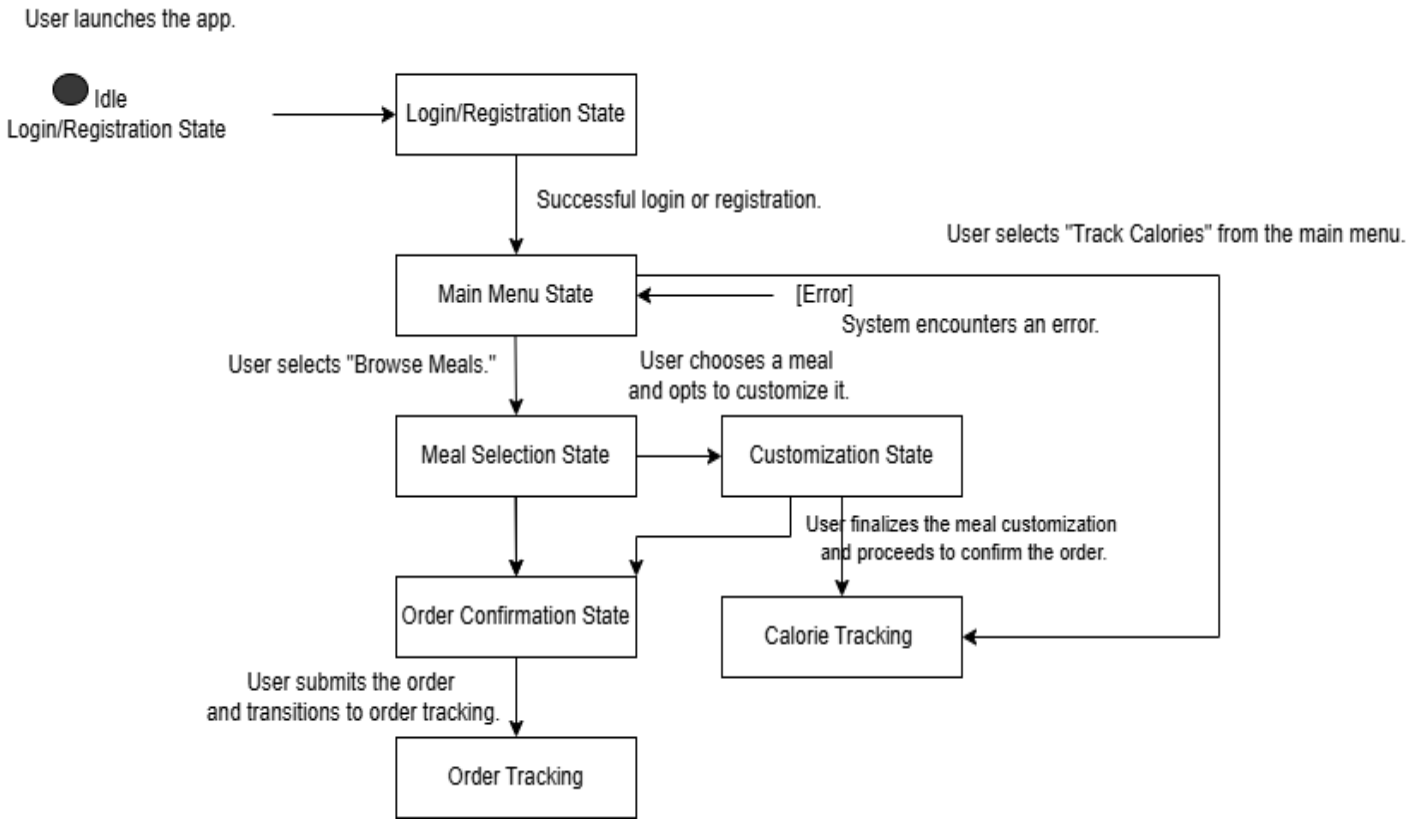


Figure 6.3.2 State Diagram For FitFeast

6.3.2.1 State Diagram for Meal Kit Customization

States:

- **Idle State**: The user has not interacted with the Meal Kit feature.
- **Meal Kit Selection**: The user selects a meal kit.
- **Customization**: The user modifies meal kit ingredients, portion sizes, or preferences.
- **Confirmation**: The user confirms the customized meal kit.
- **Error State**: An issue occurs, such as unavailable ingredients.

Transitions:

- **Idle → Meal Kit Selection**: The user selects the "Customize Meal Kit" option.
- **Meal Kit Selection → Customization**: The user chooses a meal kit.
- **Customization → Confirmation**: User completes modifications and confirms.
- **Any State → Error State**: An error occurs (e.g., unavailable item).
- **Error State → Meal Kit Selection**: User retries the process.

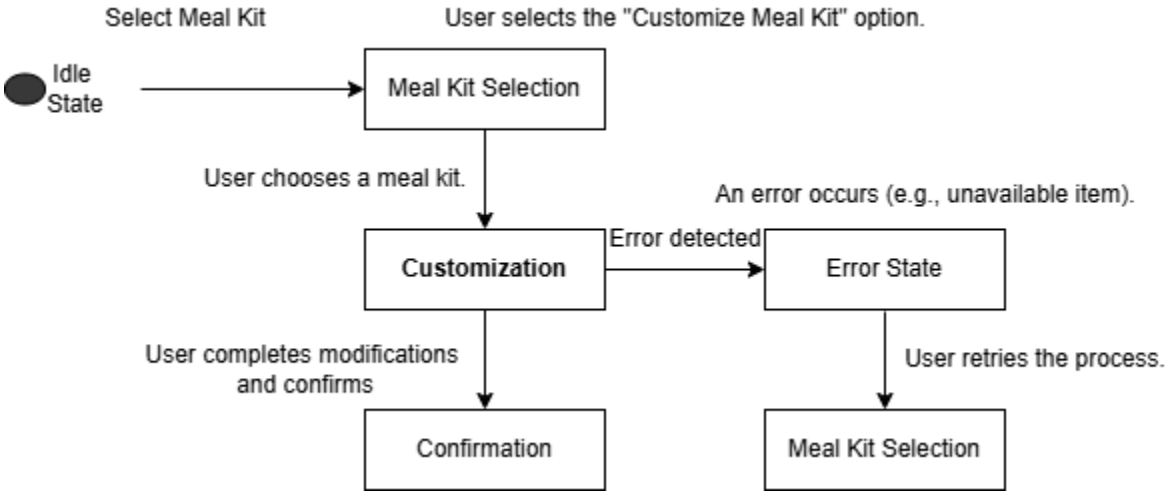


Figure 6.3.2.1 State Diagram for Meal Kit Customization

6.3.2.2 State Diagram for Calorie Count

States:

- **Idle State**: The user has not engaged with calorie tracking.
- **Input Data State**: User provides personal details (e.g., weight, activity level).
- **Daily Calorie Goal Set**: The system calculates daily calorie targets.
- **Calorie Tracking**: The user tracks daily intake and progress.
- **Error State**: System error or invalid user input.

Transitions:

- **Idle → Input Data State**: User selects "Track Calories."
- **Input Data State → Daily Calorie Goal Set**: User submits profile data.
- **Daily Calorie Goal Set → Calorie Tracking**: The system tracks calories as users log meals.
- **Any State → Error State**: Errors such as incorrect inputs or failed calculations.
- **Error State → Input Data State**: User retries.

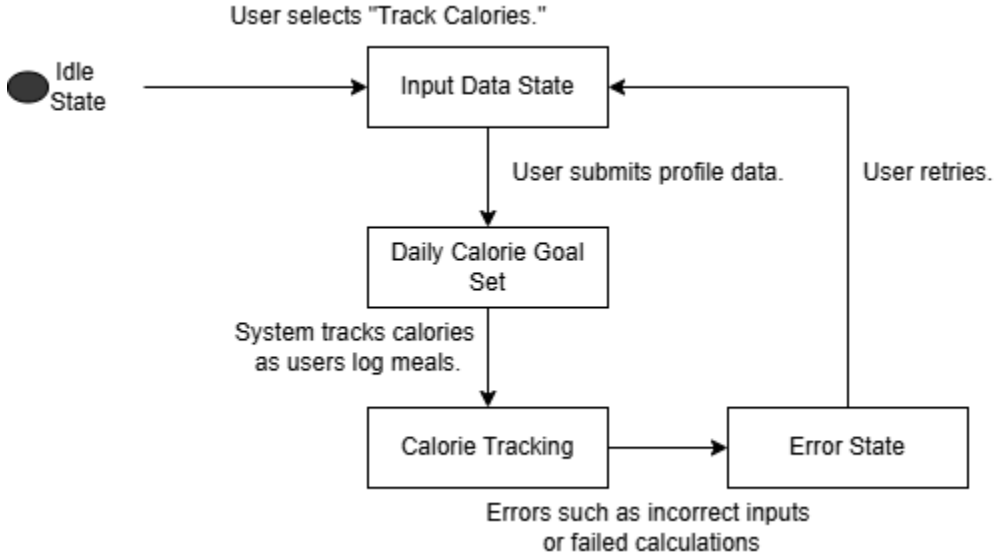


Figure 6.3.2.2 State Diagram for Calorie Count

6.3.2.3 State Diagram for Avoiding Incompatible Combinations

States:

- **Idle State**: The user is not selecting meal components.
- **Ingredient Selection**: The user selects meal ingredients.
- **Validation State**: System checks for incompatible combinations using rules/ML.
- **Alert State**: Incompatible combinations are flagged.
- **Resolution State**: The user modifies the selection or overrides the warning.
- **Confirmation State**: Finalized meal order.

Transitions:

- **Idle → Ingredient Selection**: The user starts adding ingredients.
- **Ingredient Selection → Validation State**: The system validates the selection.
- **Validation State → Alert State**: System flags incompatible combinations.
- **Alert State → Resolution State**: User resolves the flagged issue.
- **Resolution State → Confirmation State**: User confirms the selection.

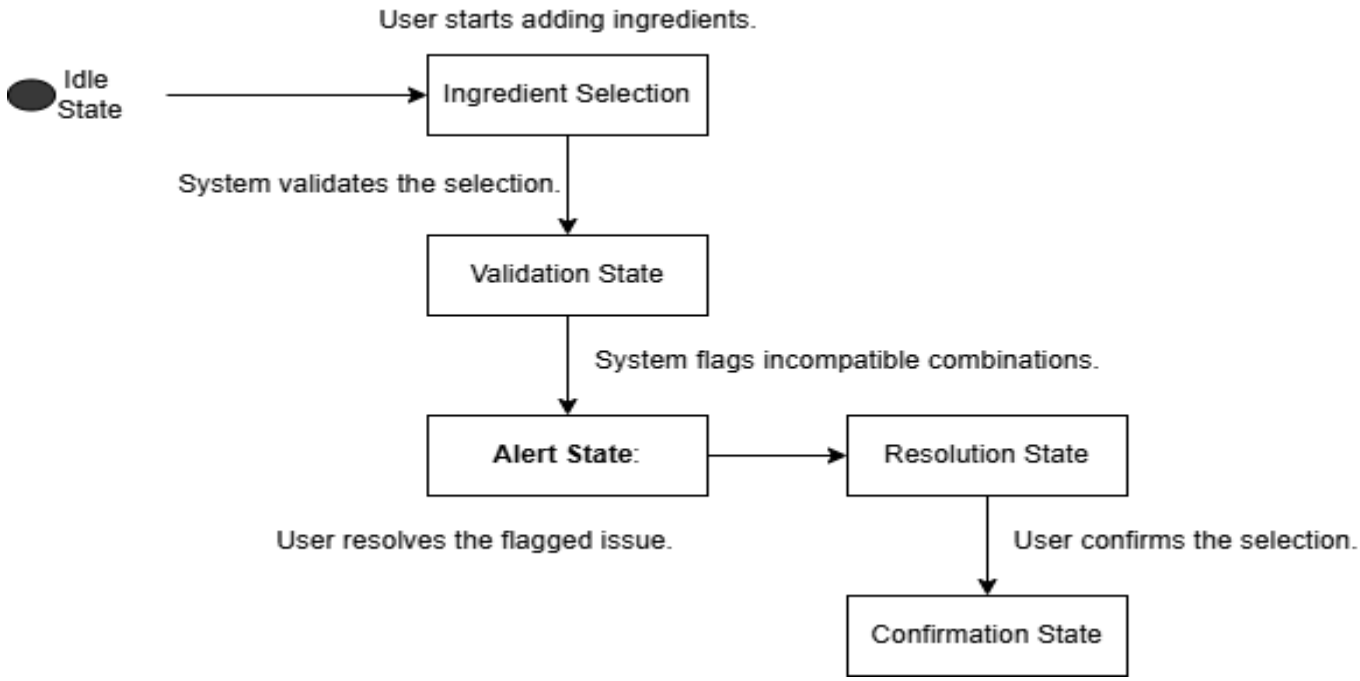


Figure 6.3.2.3 State Diagram for Avoiding Incompatible Combinations

6.3.2.4 State Diagram for Food Recommendations

States:

- **Idle State**: The user is not browsing recommendations.
- **Meal Analysis State**: The system analyzes meals logged by the user.
- **Threshold Exceeded**: System detects calorie surplus based on set thresholds.
- **Recommendation Generation**: The system suggests healthier alternatives.
- **Recommendation Review**: The user views and selects a recommended meal.
- **Error State**: System fails to fetch or process data.

Transitions:

- **Idle → Meal Analysis State**: User logs meals.
- **Meal Analysis State → Threshold Exceeded**: System calculates calorie surplus.
- **Threshold Exceeded → Recommendation Generation**: System generates alternatives.
- **Recommendation Generation → Recommendation Review**: User views the suggestions.
- **Any State → Error State**: An error occurs.

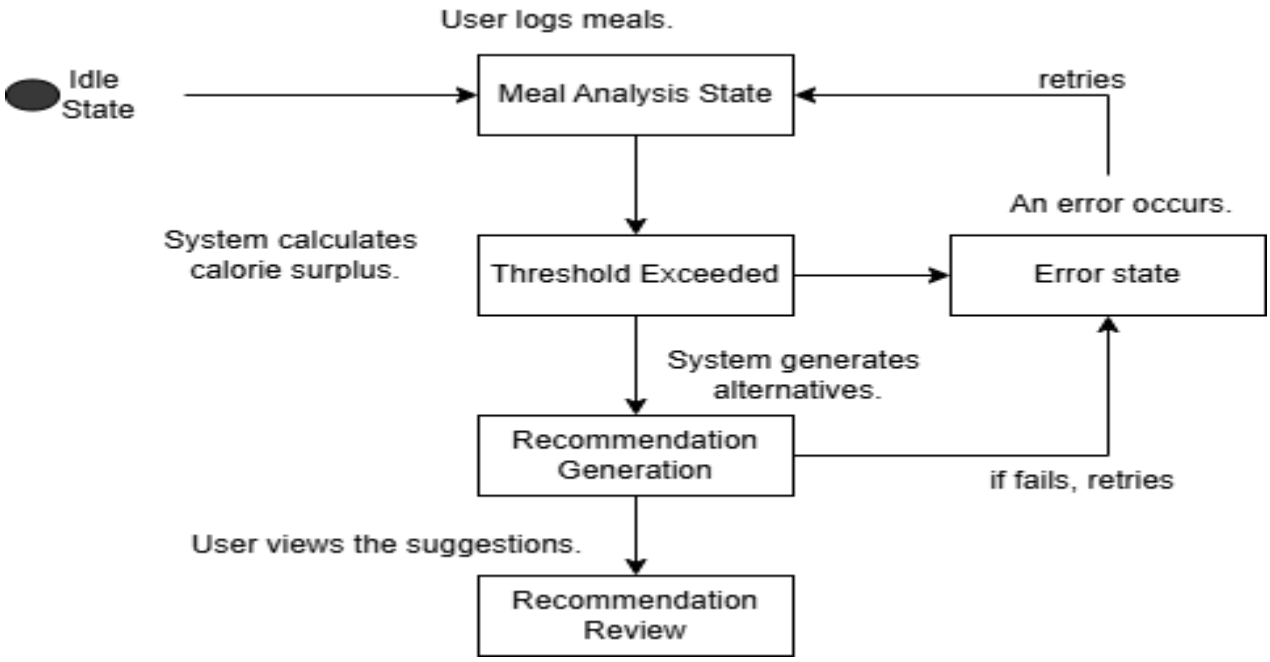


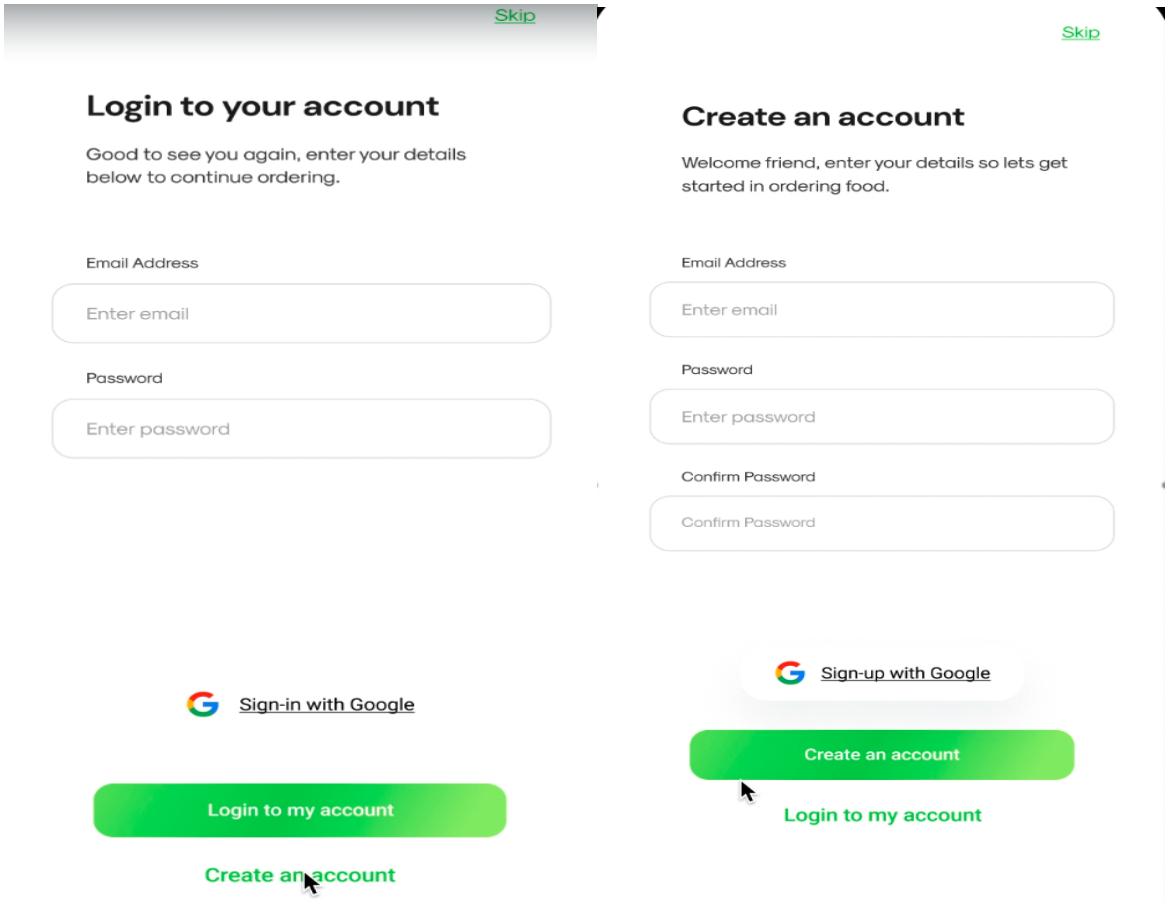
Figure 6.3.2.4 State Diagram for Food Recommendations

6.4 GUI Design

6.4.1 Mock Screen 1: User Login/Sign-Up Screen

Description:
This screen allows users to log in using their email, password, or social media accounts. New users can create an account by entering their personal details, including name, age, email, and dietary preferences.
Features:

- Input fields for email and password.
- "Forgot Password" and "Sign Up" links.
- Social media login buttons.
- Validation errors for invalid credentials.



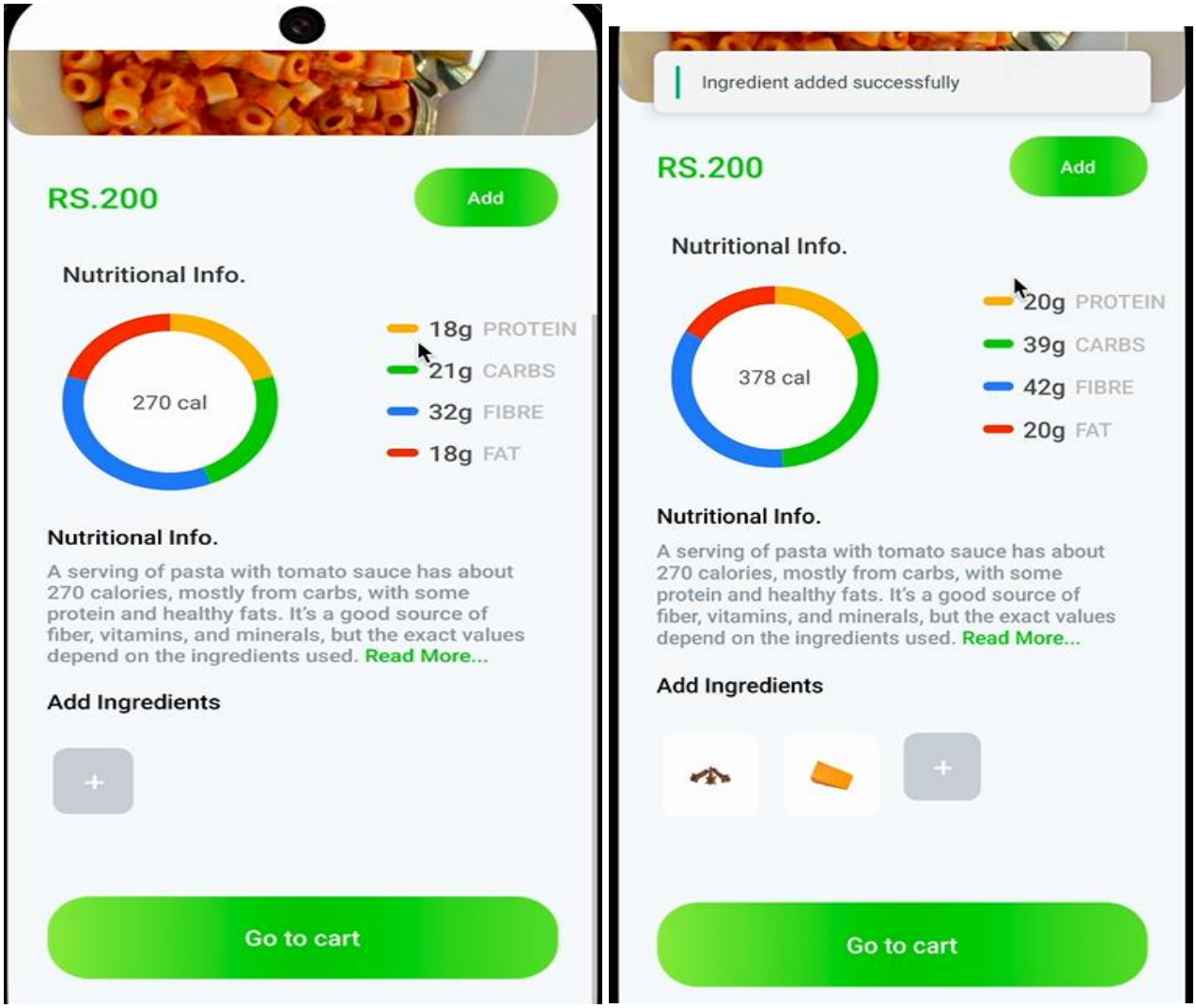
6.4.2 Mock Screen 2: Calorie Info In Meal

Description:

This screen represents the details of a specific meal, It includes a high-resolution image of the meal at the top, followed by key meal attributes, including calorie count and price. Nutritional information is presented in a visually appealing circular graph that shows the breakdown of calories into **Protein, Carbs, Fibre, and Fat**. Below this, a button labeled "Add" allows users to include the meal in their cart or order list. Further down, there is a **Delivery Info** section providing additional information about the meal and a list of **Ingredients** represented using graphical icons.

Key Features:

1. **High-Quality Meal Image** - Visually engaging display of the selected meal.
2. **Calorie Count** - Shows the total calories, emphasizing the health-conscious nature of the app.
3. **Nutritional Breakdown** - Displays macro-nutrients (Protein, Carbs, Fibre, Fat) using a pie chart for quick analysis.
4. **Add Button** - Action button allowing users to add the meal to their cart.
5. **Ingredients Section** - Clear graphical representation of the key ingredients used in the meal.



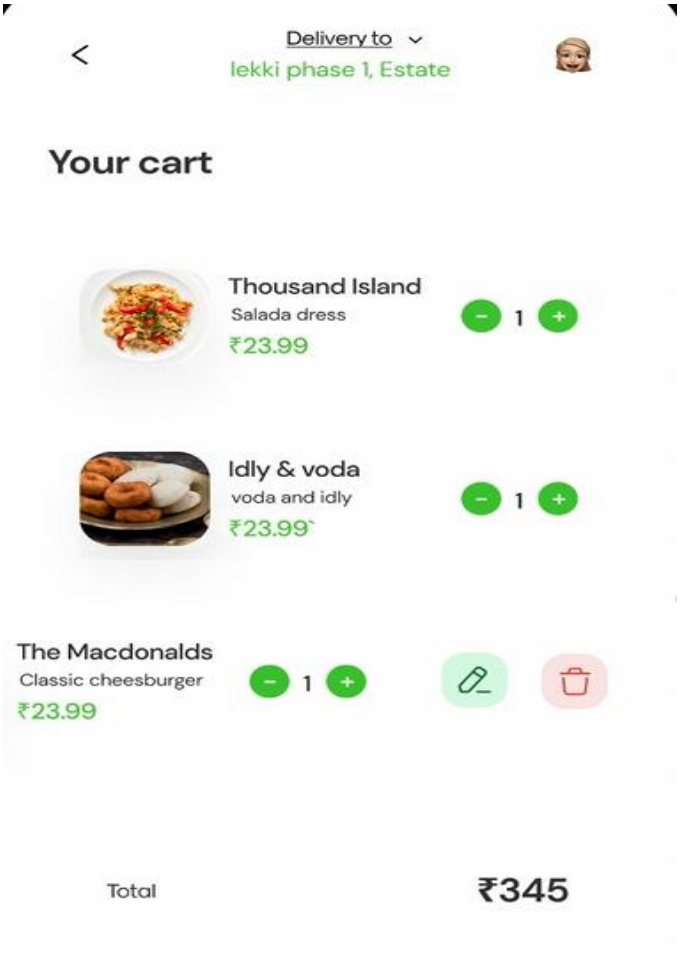
6.4.3 Mock Screen 3: Shopping Cart Screen

Description:

The Shopping Cart Screen is where users can review and manage items they have added for purchase. Each meal is displayed with options to edit quantities, remove items, or view their price details. The cart totals the price of all selected meals and provides a direct pathway to payment through a prominent action button.

Features:

1. **Cart Header**: Includes delivery location and cart title.
2. **Meal Items List**:
    - Displays meal images, titles, and descriptions.
    - Prices are clearly shown for each item.
3. **Quantity Controls**: Allows users to increase or decrease meal quantities.
4. **Edit and Remove Options**: Includes buttons for editing the cart and removing specific items.
5. **Total Price**: Displays the final cart amount prominently.
6. **Proceed to Payment Button**: A green call-to-action button to complete the purchase.

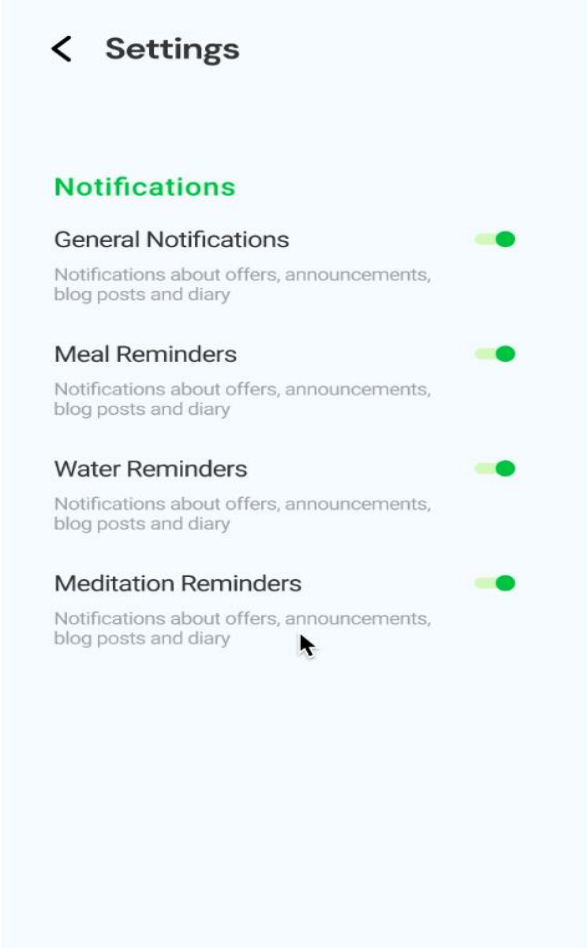


6.4.4 Mock Screen 4: Notification Settings

Description: The Notification Settings Screen enables users to manage and customize alerts for general notifications, meal reminders, water intake, and meditation sessions.

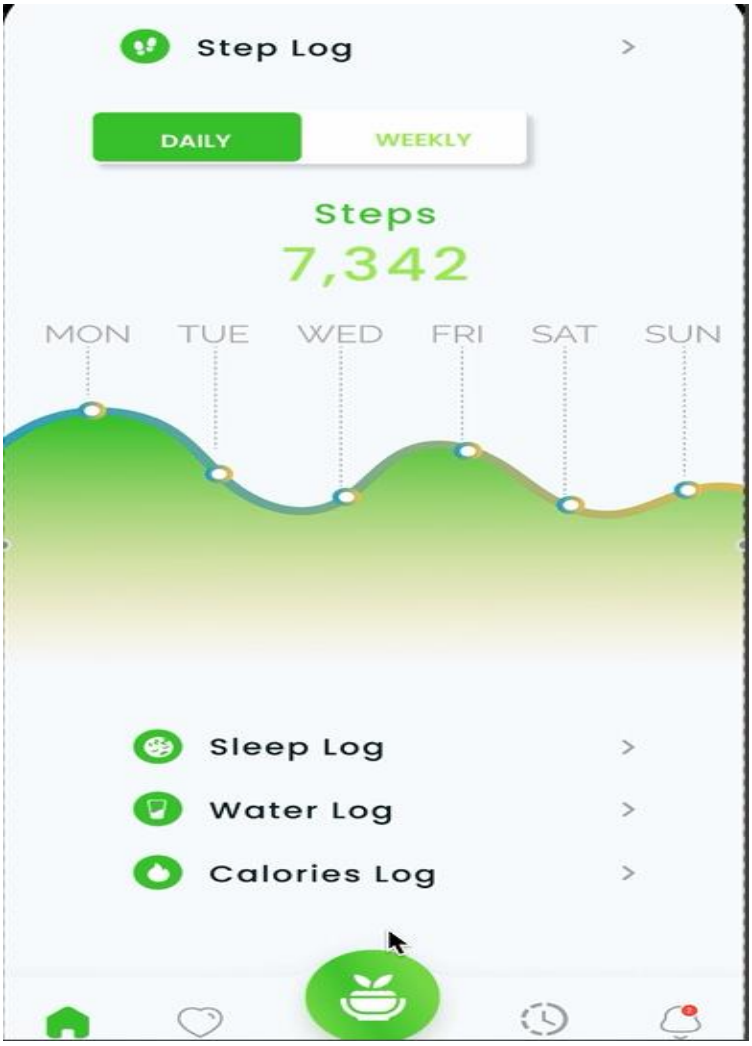
Features:

1. **Header:** Title "Notification Settings" with a back button.
2. **General Notifications:** Toggle for general updates and tips.
3. **Meal Reminders:** Toggles for breakfast, lunch, and dinner reminders with timing options.
4. **Water Reminders:** Toggle and frequency settings for hydration alerts.
5. **Meditation Reminders:** Toggle and time selection for mindfulness sessions.
6. **Save Changes:** Button to apply updated settings.
7. **Reset Defaults:** Option to revert to default notification preferences.



6.4.5 Mock Screen 4: Side Features

This is a GUI for side feature that will allow to keep track of water intake and steps counting like a health maintaining side bar.



References

- **“Food Recommendation System Using Clustering Analysis for Diabetic Patients, Advanced Virtual and Intelligent Computing (AVIC)”** Research Center Department of Mathematics, Faculty of Science, Chulalongkorn University Pathumwan, Bangkok, Thailand, Proc. 6th Int’l Conf Machine Learning and Cybernetics, August 2007
- **“Healthy Personalized Recipe Recommendations for Weekly Meal Planning” By Konstantinos Zioutos, Haridimos Kondylakis , and Kostas Stefanidis** https://www.researchgate.net/publication/376697965_Healthy_Personalized_Recipe_Recommendations_for_Weekly_Meal_Planning **Published on December 2023**
- **"Personalized Nutrition: Principles and Applications"** (2020) Authors: Elissa Epel, Ph.D., et al. https://www.researchgate.net/publication/345679123_Belluschi_Pietro
- **"Machine Learning in Nutrition: A Review"** (2020) Authors: Dimitrios A. Koutelidakis, et al. https://www.researchgate.net/publication/339812652_A_phase_I_study_of_triapine_in_combination_with_irinotecan_in_refractory_tumors
- **"Development of a Personalized Meal Planning System Using Machine Learning" (2019)** Authors: Rui Song. https://www.researchgate.net/publication/334960387_ahkam_altn_balnqd_bamr_khty_fy_alqanwn_alardny
- **"Nutritionix API Documentation"** Authors: Nutritionix Team.
- **"Personalized Diet Plans: Integrating AI and User Preferences"** (2020) Authors: Meng Wang, Ph.D., et al. https://www.researchgate.net/publication/341065798_Entrevista_Armando_Caldeira-Pires
- "Machine Learning in Dietary Assessment: Challenges and Opportunities" (2021) Authors: George P. Tzimas, https://www.researchgate.net/publication/348296957_Labor_Market_Consequences_of_the_Tenant_Purchase_Scheme_20_Years_On
- **Personalized Flexible Meal Planning for Individuals With Diet-Related Health Concerns:** System Design and Feasibility Validation Study https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10436119/ Published online **2023 Aug 3.** Authors: Maryam Amiri, Juan Li & Wordh Hasan.
- Naw Naw and Ei Ei Hlaing. **"Relevant Words Extraction Method for Recommendation System."** Buletin Teknik Elektro dan Informatika Vol. 2, No. 3 (2013).
- "Food recommendation system using clustering analysis for diabetic patients." By Phanich, Maiyaporn, Phathrajarin Pholkul, and Suphakant Phimoltares. 2010 International Conference on Information Science and Applications. IEEE, 2010.
- Gaikwad, D. S., et al. **"Food Recommendation System."** International Research Journal of Engineering and Technology” 4.01 (2017).
- **"Interaction design in a mobile food recommender system."** By Elahi, Mehdi, CEUR Workshop Proceedings. CEUR-WS, 2015

……..THANKYOU……

Appendices

Appendix A: Glossary

- **Calorie Count**: A feature that tracks the user's daily calorie intake.
- **Personalized Meal Kit**: Allows users to customize meals by adding or removing ingredients.
- **Incompatible Food Combinations**: Alerts users about harmful or strange food pairings.
- **Food Recommendations**: Provides suggestions when calorie intake exceeds predefined limits.

Appendix B: Dataset Sample

Food Items and Nutritional Values

Food Item	Calories (kcal)	Protein (g)	Carbohydrates (g)	Fats (g)	Category
Grilled Chicken	165	31	0	4	Protein
Brown Rice	216	5	45	2	Carbohydrates
Broccoli	55	4	11	0	Vegetables
Olive Oil	119	0	0	14	Fats

Food Compatibility Table

Food Item A	Food Item B	Compatibility Score	Notes
Grilled Chicken	Broccoli	95%	High compatibility
Brown Rice	Olive Oil	90%	Recommended for cooking
Milk	Lemon Juice	0%	Avoid – Causes curdling

User Preferences and Calorie Goals

User ID	Name	Daily Calorie Goal	Preferred Foods	Avoid Foods
001	Alice	1800	Chicken, Broccoli	Dairy
002	Bob	2200	Rice, Olive Oil	High Sugar Items

Food Recommendation Examples

Current Calorie Intake	Recommendation	Reason
1500 kcal	Add a serving of grilled chicken	Protein intake insufficient
2000 kcal	Add vegetables like broccoli	Balance overall nutrition

Appendix C : References

1. USDA FoodData Central. (2023). *Food and Nutrient Database.* Retrieved from https://fdc.nal.usda.gov
2. Smith, J., & Johnson, K. (2020). *Machine Learning Techniques for Food Recommendation Systems.* Journal of Artificial Intelligence Research, 45(3), 123-145.
3. Flutter Documentation. (2024). *Build Beautiful UIs with Flutter.* Retrieved from https://flutter.dev/docs
4. Python Software Foundation. (2024). *Python 3.10 Documentation.* Retrieved from https://docs.python.org/3
5. Brown, L., & Green, P. (2019). *Health and Wellness through Nutrition Tracking Applications.* International Journal of Nutrition Science, 12(4), 85-102.
6. National Institutes of Health. (2023). *Dietary Guidelines for Caloric Intake.* Retrieved from https://www.nhlbi.nih.gov
7. OpenAI. (2023). *Natural Language Processing in Recommendation Systems.* Retrieved from https://www.openai.com/research
8. MySQL Documentation. (2024). *Database Management for Applications.* Retrieved from https://dev.mysql.com/doc