Hamdard University
Department of Computing
Final Year Project

FitFeast: Personalized Nutrition & Meal Planning System

FYP-016/FL24

Software Requirements 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	<1.4>
Document Identifier	FYP-016/FL24 SRS
Document Status	Final
Author(s)	Rabail, Sara, Maryam
Approver(s)	Engr. Farooq Iqbal
Issue Date	

Name	Role	Signature	Date
Rabail Arshad	Team Lead – 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
3/Oct/2024	V-1	Making of SRS	All Members
6/Oct/2024	V-1.1	Initial document with requirements	All Members
30/Oct/2024	V-1.2	Descriptive Req Gathering	All Members
5/Nov/2024	V-1.3	Refined functional requirements	All Members
21/Nov/2024	V-1.4	Finalized SRS	All Members

Definition of Terms, Acronyms, and Abbreviations

Term	Description
FitFeast	The food ordering application provides customized meal kits, calorie management, and recommendations.
SRS	Software Requirements Specification, is a document detailing the system's requirements, features, and functionalities.
ML	Machine Learning, is a branch of AI used for avoiding incompatible food combinations in the system.
Personalized Meal Kit	A feature allowing users to customize meals according to their dietary preferences.
Calorie Count	A feature to manage and display the total calories of selected meals.
Incompatible Combinations	The detection and prevention of unsuitable food combinations using machine learning algorithms.
Recommendation	A feature that provides alternative meal suggestions when calorie intake exceeds user preferences.
UserPreferences Table	Database table storing user-specific dietary preferences and restrictions.
REST API	Representational State Transfer Application Programming Interface, used for communication.
Nutritional API	External API provides nutritional data for ingredients and meals.
MySQL	A relational database management system is used to store and retrieve application data.

Table of Contents

1. Introduction

FitFeast is a food ordering app that aims to provide users with personalized nutrition, and calorie control, and recommend healthy alternatives when appropriate. The platform uses machine learning to prevent food mix-ups, ensure consistency, and promote healthy eating. The software aims to make healthy eating easier, more flexible, and more personalized to fit customers' food preferences and nutritional needs. This document outlines the requirements and specifications needed to build the FitFeast platform.

1.1 Purpose of Document

The purpose of this document is to outline the system requirements and specifications for the development of the *FitFeast* food ordering platform. It serves as a comprehensive reference for all stakeholders involved, detailing the features, functionalities, and constraints of the system to ensure proper understanding and successful implementation.

1.2 Intended Audience

This document is intended for software developers, project managers, testers, and stakeholders involved in the design, development, and deployment of *FitFeast*.

1.3 Abbreviations

Term	Description
FitFeast	The food ordering application provides customized meal kits, calorie management, and recommendations.
SRS	Software Requirements Specification, is a document detailing the system's requirements, features, and functionalities.
ML	Machine Learning, is a branch of AI used for avoiding incompatible food combinations in the system.
Personalized Meal Kit	A feature allowing users to customize meals according to their dietary preferences.
Calorie Count	A feature to manage and display the total calories of selected meals.
Incompatible Combinations	The detection and prevention of unsuitable food combinations using machine learning algorithms.
Recommendation	A feature that provides alternative meal suggestions when calorie intake exceeds user preferences.

2. Overall System Description

2.1 Project Background

FitFeast is a food delivery application aimed at providing users with personalized meal options, calorie management, and food recommendations using machine learning. The platform ensures healthy eating habits and avoids incompatible food combinations, offering a tailored nutrition experience.

2.2 Problem Statement

Many food delivery platforms lack personalization and proper nutritional guidance. FitFeast solves this by offering meal customization, calorie tracking, and food recommendations to meet specific dietary needs.

2.3 Project Scope

FitFeast will enable personalized meal plans, calorie tracking, and machine learning-based food recommendations, with a web-based and mobile application interface.

2.4 Not In Scope Virtual reality tours, live kitchen streams, and mood-based recommendations are not included.

2.5 Project Objectives

- Provide users with personalized meals based on dietary preferences.
- Enable calorie tracking for meals.
- Suggest meal alternatives when calorie intake exceeds the user's normal range.
- Avoid incompatible food combinations using machine learning.

2.6 Stakeholders & Affected Groups

- **Primary Stakeholders:** Customers, app developers, nutritionists.
- **Secondary Stakeholders:** Restaurant partners, and delivery personnel.

2.7 Operating Environment

FitFeast will be available on iOS and Android platforms, with backend services hosted in a cloud environment using MySQL/SQL Server for database management.

2.8 System Constraints

- Must operate on mobile and web platforms.
- Integration with machine learning libraries for food recommendations.
- Limited budget for third-party integrations.

2.9 Assumptions & Dependencies

- Access to real-time food databases and nutrition APIs.
- Users will have basic internet connectivity for app use.

3. External Interface Requirements

3.1 Hardware Interfaces

FitFeast will operate on mobile devices (iOS, Android) and desktops. It requires a device with at least 2GB RAM and stable internet connectivity. The backend will be hosted on cloud servers, supporting databases like MySQL/SQL Server. No specific physical addresses are required, but the hardware should support REST API-based operations.

3.2 Software Interfaces

FitFeast will interact with the following applications:

- **MyFitnessPal API** (External) for calorie tracking. Optional
- **Google Maps API** (External) for location-based services. Optional
- **Payment Gateway** (External) for transaction processing. Optional

3.3 Communications Interfaces

FitFeast will use HTTPS for secure data communication over mobile and web platforms. It will connect with remote databases via a RESTful API and use standard encryption protocols for secure transactions.

4. System Functions / Functional Requirements

4.1 System Functions

Ref #	Functions	Category	Attribute	Details & Boundary Constraints
R1.1	Calorie Calculation based on the user's order	Evident	Response Time	Display calorie count within 2 seconds of meal selection.
R1.2	Avoid incompatible combinations (e.g., chocolate in biryani)	Hidden	Algorithm Accuracy	Detect inappropriate combinations with at least 95% accuracy using machine learning algorithms.
R1.3	Personalized meal kits	Evident	Customization Options	Allow users to customize meal kits based on dietary preferences. Show updated meals within 3 seconds.
R1.4	Recommendation for healthier alternatives	Evident	Suggestion Response Time	Display alternative meal recommendations if calorie intake exceeds the user’s average by more than 10%, with results appearing within 2 seconds.
R1.5	User Profile Management	Evident	Security, Scalability	Enable users to create and update profiles, including dietary preferences and restrictions. Data should be saved securely with encryption standards and handled for up to 1,000 concurrent users.
R1.6	Mobile App Performance	Hidden	Response Time, Efficiency	Ensure the app can handle at least 500 requests per second with less than 3-second page loading time across mobile and web platforms.
R1.7	ML-based ingredient compatibility check	Hidden	Algorithm Response Time	Incompatible ingredients were flagged within 2 seconds during the meal customization process.

4.1.1 Functional Requirements:

1. Calorie Count

- **Daily Calorie Goal**:
  - **Process**: Users set their daily calorie goal on the profile setup screen or within settings. They select from goals like weight loss, maintenance, or gain.
  - **How It Works**: The goal is saved in a UserPreferences table in the database, and the app references this daily to track progress. APIs fetch this data for each calorie-related feature, like calorie suggestions.
- **Meal-wise Breakdown**:
  - **Process**: When users select a meal, they can view a calorie breakdown for each ingredient.
  - **How It Works**: Each ingredient's calorie count, stored in an Ingredients table, is pulled via a query and displayed as a total and per-ingredient breakdown on the meal page.
- **Track Daily Progress**:
  - **Process**: The app shows users their calorie intake in real-time, visualizing daily goals in a progress bar.
  - **How It Works**: A UserActivity table logs calorie intake for each meal, and the data is visualized on a progress bar or chart, updating dynamically with each meal.
- **Historical Data**:
  - **Process**: Users navigate to a history screen to view calorie intake over days, weeks, or months.
  - **How It Works**: Past intake data is stored in a CalorieHistory table, and the app fetches it with queries for the chosen date range. Trends are displayed using line or bar charts.
- **Calories Burned**:
  - **Process**: Users log exercises in a dedicated section, adjusting net calorie totals based on calories burned.
  - **How It Works**: Exercise data is saved in a UserExercise table, and calorie totals are recalculated by factoring in calories burned for the day.
- **Custom Caloric Suggestions**:
  - **Process**: The app suggests meals that align with the remaining daily calories.
  - **How It Works**: Using the Meals table and the user's remaining calorie data from UserPreferences, a query filters meal suggestions based on remaining calorie limits.

- **Real-time Updates**:
  - **Process**: When users add or change ingredients, calorie totals automatically update.
  - **How It Works**: The Ingredients table calculates the total calorie count, and the app displays this dynamically as users modify their meals.
- **Overconsumption Alerts**:
  - **Process**: If users select meals that might exceed their daily limit, the app sends an alert.
  - **How It Works**: If the total intake approaches the daily goal from UserPreferences, the app triggers a notification to warn users.
- **Macronutrient Breakdown**:
  - **Process**: The app displays macronutrient values for each meal.
  - **How It Works**: Macronutrient data is stored alongside calorie data in the Ingredients table, and it is visualized as percentages for each selected meal.
- **Personalized Adjustments**:
  - **Process**: Recommendations are tailored based on users' physical attributes, such as weight and activity level.
  - **How It Works**: User metrics are stored in the UserProfile table, and an algorithm uses this data to adjust calorie suggestions shown on the dashboard.
- **Allergen Impact**:
  - **Process**: Users are notified if selected ingredients contain allergens.
  - **How It Works**: Allergen information is stored in the Ingredients table. When users select a meal, the app checks for allergens and flags ingredients as necessary.
- **Dynamic Meal Adjustments**:
  - **Process**: Users can change portion sizes or ingredients, and calorie counts update accordingly.
  - **How It Works**: Portion and ingredient changes are recalculated on the fly, pulling data from Ingredients, which refreshes totals in real time.
- **Nutrition Recommendations**:
  - **Process**: The app provides tips on low-calorie alternatives and healthier ingredients.
  - **How It Works**: A recommendation engine suggests alternative ingredients based on calorie reduction goals, pulling data from an AlternativeIngredients table.

- **Caloric Range Categories**:
  - **Process**: Users can filter meal options based on a calorie range.
  - **How It Works**: The app uses data from Meals to apply filters based on low, medium, or high calorie categories.
- **Daily Caloric Deficit Goal**:
  - **Process**: The app tracks users' caloric deficit for weight loss goals.
  - **How It Works**: A user's caloric intake, adjusted by exercise data from UserExercise, is displayed against their set daily goals in the UserPreferences table.
- **Caloric Intake Warnings**:
  - **Process**: The app flags meals that may exceed daily intake significantly.
  - **How It Works**: If a meal's calorie count greatly surpasses the user's remaining goal, an alert is triggered, offering lower-calorie meal options.
- **Calorie-based Discounts**:
  - **Process**: FitFeast offers discounts for calorie-controlled meal options.
  - **How It Works**: Discount data linked to low-calorie meals is stored in DiscountOffers. The app presents these offers when users view qualifying meals.
- **Multi-meal Calorie Syncing**:
  - **Process**: Calories logged across devices sync in real-time.
  - **How It Works**: CalorieHistory data syncs across devices using API-based synchronization, allowing users to access data across multiple platforms.
- **In-app Guides**:
  - **Process**: Users can access educational content on calorie tracking and nutrition.
  - **How It Works**: Content is stored in a Guides database and available on a help or resource section, guiding users with tips and explanations.
- **Personalization Engine**:
  - **Process**: Users see calorie recommendations based on age, gender, and metabolic rate.
  - **How It Works**: The app's algorithm pulls user-specific data from the UserProfile table to personalize calorie needs and updates suggestions accordingly.

2. Personalized Meal Kits:

Users must be able to customize meals based on dietary preferences.

- **Customizable Ingredients**:
  - **Process**: Users select a preset meal kit and personalize it by adding or removing ingredients.
  - **How It Works**: Ingredient modifications are saved to a UserMealKit table, where the app retrieves ingredient data from Ingredients and recalculates nutritional values in real-time.
- **Dietary Preferences**:
  - **Process**: Users set dietary preferences in their profile (keto, vegan, etc.).
  - **How It Works**: The app filters meal kits to match preferences, checking UserPreferences and MealCategories to show only relevant options.
- **Meal Component Suggestions**:
  - **Process**: Users see suggested meal components based on past choices.
  - **How It Works**: Past meal components from OrderHistory inform suggestions in MealComponents. These recommendations adjust dynamically for each user.
- **User-defined Macros**:
  - **Process**: Users define macros they want in a meal kit.
  - **How It Works**: Macro targets are saved in UserMacros, and the system calculates ingredient contributions to match the user's defined goals.
- **Meal Variety**:
  - **Process**: If an ingredient is unavailable, the app suggests similar meal kits.
  - **How It Works**: Using data from Inventory, unavailable ingredients trigger alternate suggestions in the MealKitOptions.
- **Seasonal Ingredients**:
  - **Process**: Users see seasonal ingredients based on their location.
  - **How It Works**: The app queries LocalIngredients based on location and season, using the UserLocation table for filtering options.
- **Save Favorite Kits**:
  - **Process**: Users save and reorder favorite meal kits.

- **How It Works**: Saved kits are stored in FavoriteKits, and users can access these with a single tap for reordering.

➢ **Real-time Price Calculation**:

- **Process**: Users see real-time price adjustments as ingredients are added or removed.
- **How It Works**: Ingredient prices in Ingredients are recalculated for each modification, showing an updated total.

➢ **Portion Size Adjustment**:

- **Process**: Users adjust portion sizes for any meal.
- **How It Works**: Serving sizes in UserMealKit dynamically adjust nutritional values and ingredient amounts, reflecting accurate data for different portions.

➢ **Meal Recommendations Based on Time**:

- **Process**: The app recommends meals based on the time of day.
- **How It Works**: A MealRecommendation function queries suitable options for breakfast, lunch, or dinner based on local time settings in the user's profile.

➢ **Personalization Based on Order History**:

- **Process**: Users receive meal suggestions aligned with their order history.
- **How It Works**: OrderHistory data drives recommendations in SuggestedMeals, helping users discover similar meals they may enjoy.

➢ **Ingredient Availability Alerts**:

- **Process**: Users are notified when new ingredients match their preferences.
- **How It Works**: New ingredient availability in Ingredients triggers notifications to users with relevant dietary preferences.

➢ **Meal Plan Subscription**:

- **Process**: Users subscribe to weekly/monthly meal plans.
- **How It Works**: Subscription data in UserSubscription drives automated order generation, based on personal preferences and set schedules.

➢ **Cross-meal Compatibility**:

- **Process**: The app ensures meal kits align with past preferences and restrictions.
- **How It Works**: The UserPreferences table cross-checks dietary restrictions across multiple meals, avoiding any conflict.

- **Personalized Recipe Adjustments**:
  - **Process**: Recipes automatically adjust based on user modifications.
  - **How It Works**: The app pulls personalized recipes from RecipeInstructions with updated ingredients and portion sizes.
- **Nutritional Impact Display**:
  - **Process**: When ingredients are swapped, the app shows nutritional changes.
  - **How It Works**: Updated ingredient data in Ingredients reflects the total nutrition impact, ensuring users see accurate values.
- **Meal Preparation Tips**:
  - **Process**: Users view customized prep tips.
  - **How It Works**: PreparationGuides are tailored to modifications made to meal kits, providing relevant cooking instructions.
- **Guest Mode**:
  - **Process**: Users create customized meal kits for guests with dietary needs.
  - **How It Works**: Guest data in GuestPreferences allows users to tailor meal kits for specific dietary restrictions temporarily.

3. Avoid Incompatible Combinations:

Use machine learning to avoid inappropriate food combinations (e.g., chocolate in biryani).

- **Algorithm/ML Integration**:
  - **Process**: When users select meal components, an ML algorithm scans for conflicts (e.g., chocolate with biryani).
  - **How It Works**: The ML model, integrated within the MealCombinationRules database, uses data on incompatible foods. Each ingredient is evaluated against the model's learned incompatibilities before final confirmation.
- **Dynamic Updates**:
  - **Process**: Compatibility rules are periodically refreshed based on new ingredient data or scientific research.

- **How It Works**: The system retrains the ML model, pulling new data from the FoodScience dataset to ensure it accounts for the latest dietary insights.

➢ **Manual Override**:

- **Process**: Users receive a warning for incompatible choices and may choose to override it.
- **How It Works**: The override choice, saved in UserSettings, allows flexibility, letting users proceed with selections even if flagged by the system.

➢ **Learning from Feedback**:

- **Process**: Users provide feedback on flagged combinations, which refines compatibility rules.
- **How It Works**: Feedback is stored in the UserFeedback table and used in periodic ML model retraining, making compatibility assessments more accurate over time.

➢ **Real-time Suggestions**:

- **Process**: Users receive immediate suggestions for better ingredient matches when conflicts are detected.
- **How It Works**: As users select ingredients, real-time queries against MealComponents suggest alternatives if incompatibilities are detected.

➢ **Customization**:

- **Process**: Users set personal sensitivity levels for detecting incompatible foods.
- **How It Works**: Preferences saved in UserPreferences allow users to adjust the strictness of compatibility checks, catering to individual needs.

➢ **Combination Validation**:

- **Process**: During ingredient selection, the system continually validates combinations for conflicts.
- **How It Works**: Each ingredient choice is verified against the IncompatiblePairs table to ensure no conflicts before the order is finalized.

➢ **User Education**:

- **Process**: When an incompatibility is flagged, users see a brief explanation.
- **How It Works**: The system retrieves relevant explanations from FoodEducationContent for flagged combinations, helping users understand potential issues.

➢ **Cross-meal Compatibility**:

- **Process**: Ingredients are checked for compatibility across multiple meals in the order.
- **How It Works**: The system verifies combinations for all meals in an order, referencing UserOrderHistory to prevent conflicts in multi-course meals.

➢ **Periodic Updates**:

- **Process**: ML models and rules are regularly updated with new data.
- **How It Works**: Data from culinary research and user feedback informs updates to the FoodCompatibilityModel, refining the system's accuracy in identifying incompatibilities.

Each step in this design supports a seamless user experience, backed by ML-based conflict detection, real-time feedback, and periodic updates for enhanced personalization and accuracy.

4. Food Recommendations:

Recommend alternative meals when calorie intake exceeds a set threshold.

➢ **Calorie Threshold Monitoring**:

- **Process**: System tracks calorie intake against user-defined daily or meal-based limits.
- **How It Works**: Calorie data from each meal is stored in UserIntakeHistory and checked against UserCalorieThresholds to ensure intake stays within set limits.

➢ **Recommendation Triggers**:

- **Process**: When calorie limits are exceeded, recommendation algorithms activate.
- **How It Works**: The system's event listener, linked to UserMealIntake, initiates recommendations when the calorie count surpasses the threshold.

➢ **Alternative Meal Suggestions**:

- **Process**: System provides lower-calorie options that meet dietary needs.
- **How It Works**: Queries to the MenuDatabase retrieve meals with compatible preferences and lower calorie counts.

➢ **User Preferences Integration**:

- **Process**: System factors in preferences like dietary restrictions.
- **How It Works**: Stored in UserPreferences, these details refine the recommendation algorithm to ensure personalized suggestions.

➢ **Caloric Adjustments**:

- **Process**: System suggests minor modifications to lower meal calories.
- **How It Works**: Suggested ingredient swaps or portion reductions are calculated, then shown to the user with updated calorie counts.

➢ **Nutritional Balance**:

- **Process**: The system maintains a balanced macronutrient profile in all recommendations.
- **How It Works**: Macronutrient analysis from the MealNutrients database ensures each alternative maintains the right balance.

➢ **Historical Data Use**:

- **Process**: Tailored suggestions based on the user's historical choices.
- **How It Works**: Recommendations pull from UserOrderHistory, factoring in frequently ordered items for a more personalized experience.

➢ **Real-Time Recommendations**:

- **Process**: Recommendations appear during meal selection.
- **How It Works**: The app provides real-time alerts and alternative options as users build their meals.

➢ **Post-Order Suggestions**:

- **Process**: Offers lighter options for future meals if limits are exceeded.
- **How It Works**: If an order goes over the limit, UserIntakeHistory updates to suggest low-calorie meals later in the day.

➢ **Learning Algorithm**:

- **Process**: System refines recommendations based on user feedback.
- **How It Works**: ML model analyzes choices stored in UserFeedback and UserOrderPatterns to improve future meal suggestions.

5. User Profile Management:

Users must be able to manage their dietary preferences and profiles.

- **User Registration and Login**

  - **Registration**: Users can register via **email**, **social media** (Google, Facebook), or **phone number** (OTP).
  - **Login**: Secure login with **username/password** and **multi-factor authentication**. Password recovery is available.

- **Profile Information**

  - **User Inputs**: Users enter **age**, **weight**, **height**, **activity level**, and **dietary preferences** (e.g., vegan, keto).
  - **Personalization**: This data is used to calculate **calorie needs** and generate **personalized meal plans**.

- **Current and Target Weight Tracking**

  - **Weight Input**: Users enter their **current weight** and **target weight goal**.
  - **Dynamic Plan**: The system adjusts **meal plans** and **calories** based on progress toward weight goals.

- **Calorie Intake Management**

  - **Calorie Calculation**: Based on user data, the system sets a **daily calorie goal**.
  - **Tracking**: Users track meals, and the system updates **calorie intake** in real-time, ensuring users stay within their target range.

- **Dietary Preferences**

  - **Preference Setup**: Users specify dietary preferences (e.g., vegan, gluten-free).
  - **Meal Filtering**: Meal suggestions are filtered based on these preferences.

- **Food Allergens**

  - **Allergen Input**: Users list allergens (e.g., nuts, dairy).
  - **Meal Exclusion**: The system excludes any meal containing these allergens from recommendations.

- **Goal Progress Monitoring**

  - **Dashboard**: Visual graphs track progress toward weight and nutrition goals.
  - **Real-time Updates**: The system updates meal plans and calories based on goal progress.

- **Profile Analytics**

  - **Insights**: Users view analytics on **caloric intake**, **favorite meals**, and how dietary choices impact goals over time.

- **Profile Customization**

  - **Notifications**: Users can customize **notifications** for calorie updates, meal suggestions, and weight progress.

- **Account Security**

  - **Data Protection**: The system offers **multi-factor authentication**, **data encryption**, and **regular updates** for account security.

- **Multi-Device Syncing**

  - **Cross-Device Sync**: User data (dietary preferences, meal logs) syncs across multiple devices for seamless access.

- **Meal Preferences**

  - **Cuisine/Ingredient Setup**: Users can set preferred **cuisines** or **ingredients**.
  - **Meal Prioritization**: The system prioritizes meals based on user tastes.

- **Nutrition Plan Adjustments**

  - **Plan Flexibility**: Users can switch between **nutrition plans** (e.g., weight loss, muscle gain) based on their progress.

- **Daily Notifications**

  - **Reminders**: The system sends **daily notifications** on calorie intake, meal timings, and upcoming meal suggestions. This will help user to keep themselves aware of their diet. They will set the timings and other information according to themselves

System Attributes / Nonfunctional Requirements

Attribute	Details and Boundary Constraints	Category
Response Time	All critical interactions (calorie count, recommendations) are processed in under 2-3 seconds.	Mandatory
Scalability	The system must support at least 1,000 concurrent users without downtime.	Mandatory
Security	Data encryption (SSL) and secure API access must be implemented.	Mandatory
Usability	The interface must be intuitive, mobile-friendly, and easy to use for all users.	Optional

Nonfunctional Requirements

1. **Performance:** The system must handle 500+ concurrent users with response times under 3 seconds.
2. **Scalability:** The platform should scale to accommodate growth in user base and data volume.
3. **Availability:** Ensure 99.9% uptime with minimal downtime for maintenance.
4. **Security:** Implement secure authentication, data encryption, and protection against cyber threats.
5. **Compatibility:** Support iOS, Android, and web browsers with responsive design.
6. **Maintainability:** Ensure modular code structure for easy updates and bug fixes.
7. **Usability:** Provide an intuitive, user-friendly interface to ensure high user satisfaction.
8. **Reliability:** The system should consistently provide accurate calorie tracking and food recommendations.

4.2 Use Cases

4.2.1 List of Actors

- **User**: Interacts with the application to order meals and customize preferences.
- **Administrator**: Manages user accounts, oversees system updates, and handles issues.
- **Nutritionist**: Provides meal recommendations based on user dietary needs.
- **Delivery Person**: Delivers meals to users' specified locations.
- **System**: The backend that processes data, manages orders, and performs calculations.

4.2.2 List of Use Cases

- Order Meal: Allows users to browse the menu and place orders for meals.
- Calculate Calories: Provides nutritional information and calorie counts for selected meals.
- Customize Meal Kit: Enables users to tailor meals to their preferences and dietary restrictions.
- Get Recommendations: Suggest healthier food alternatives when users exceed their calorie limits.
- Avoid Incompatible Combinations: Utilizes machine learning to ensure ingredient compatibility in meals.
- Manage User Profile: Users can update personal information and dietary preferences within their profiles.

4.2.3 Use Case Diagram

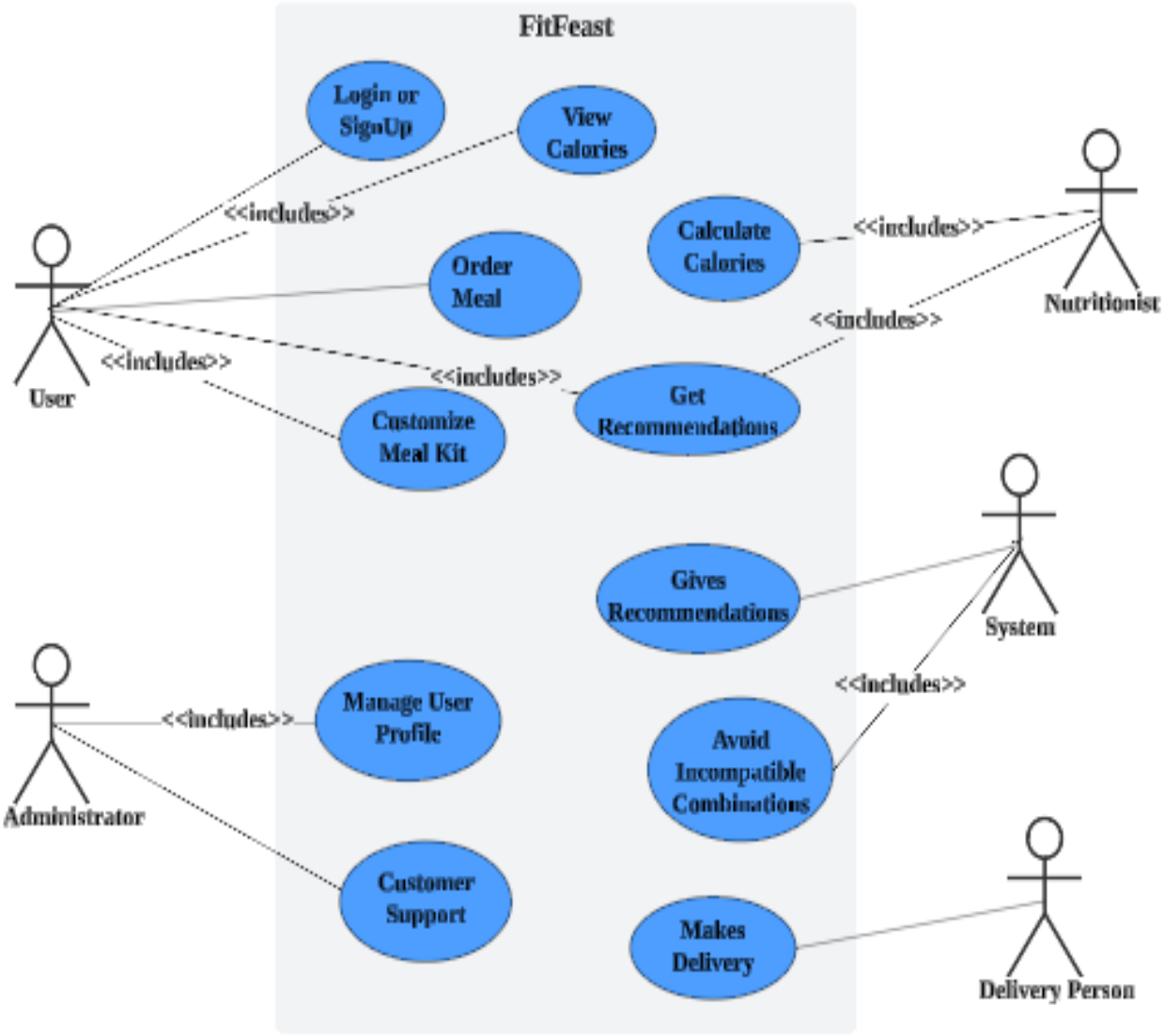

Figure 4.2.3.1 Use Case for FitFeast

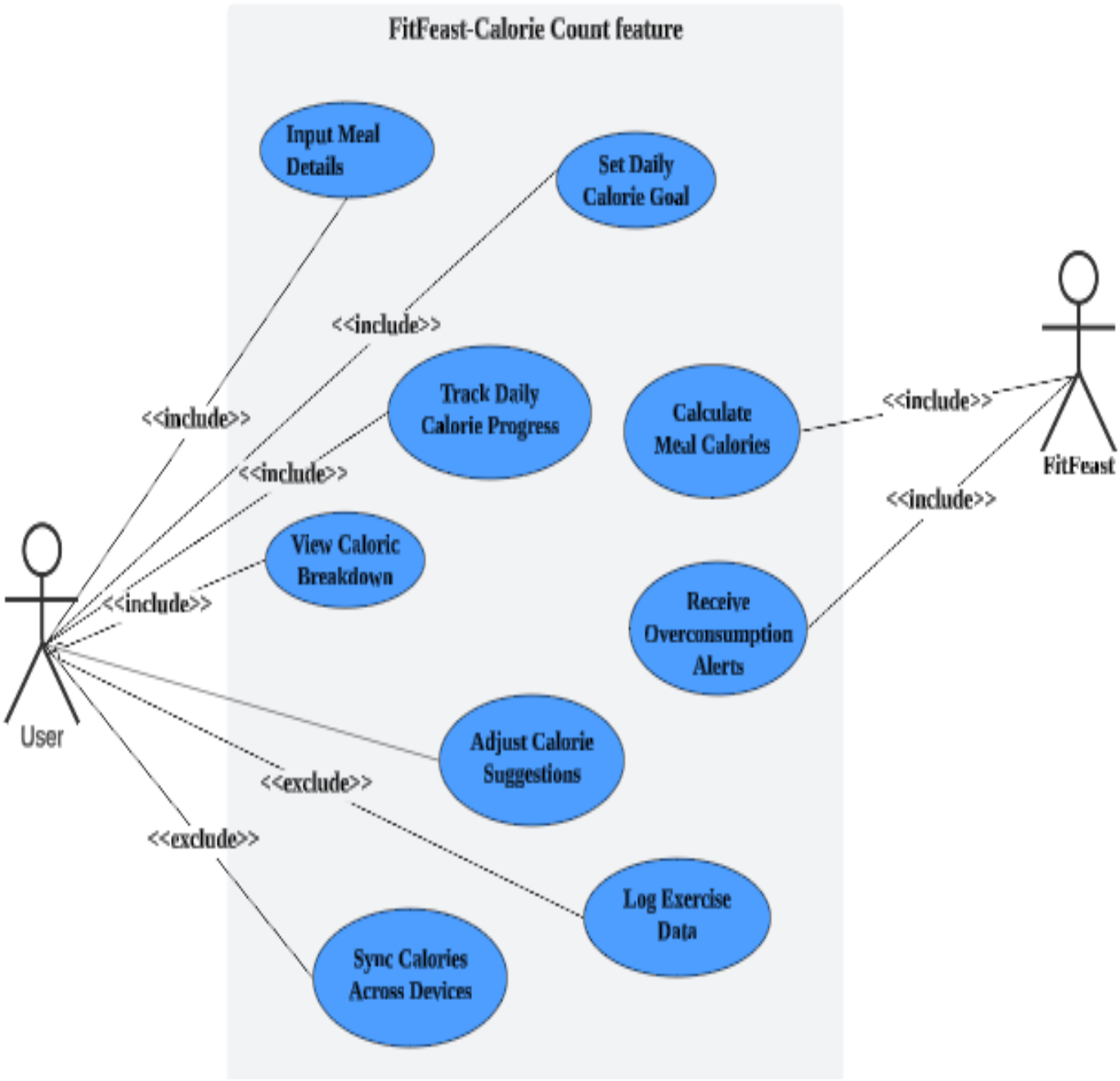

Figure 4.2.3.2 Use Case for Calorie Count

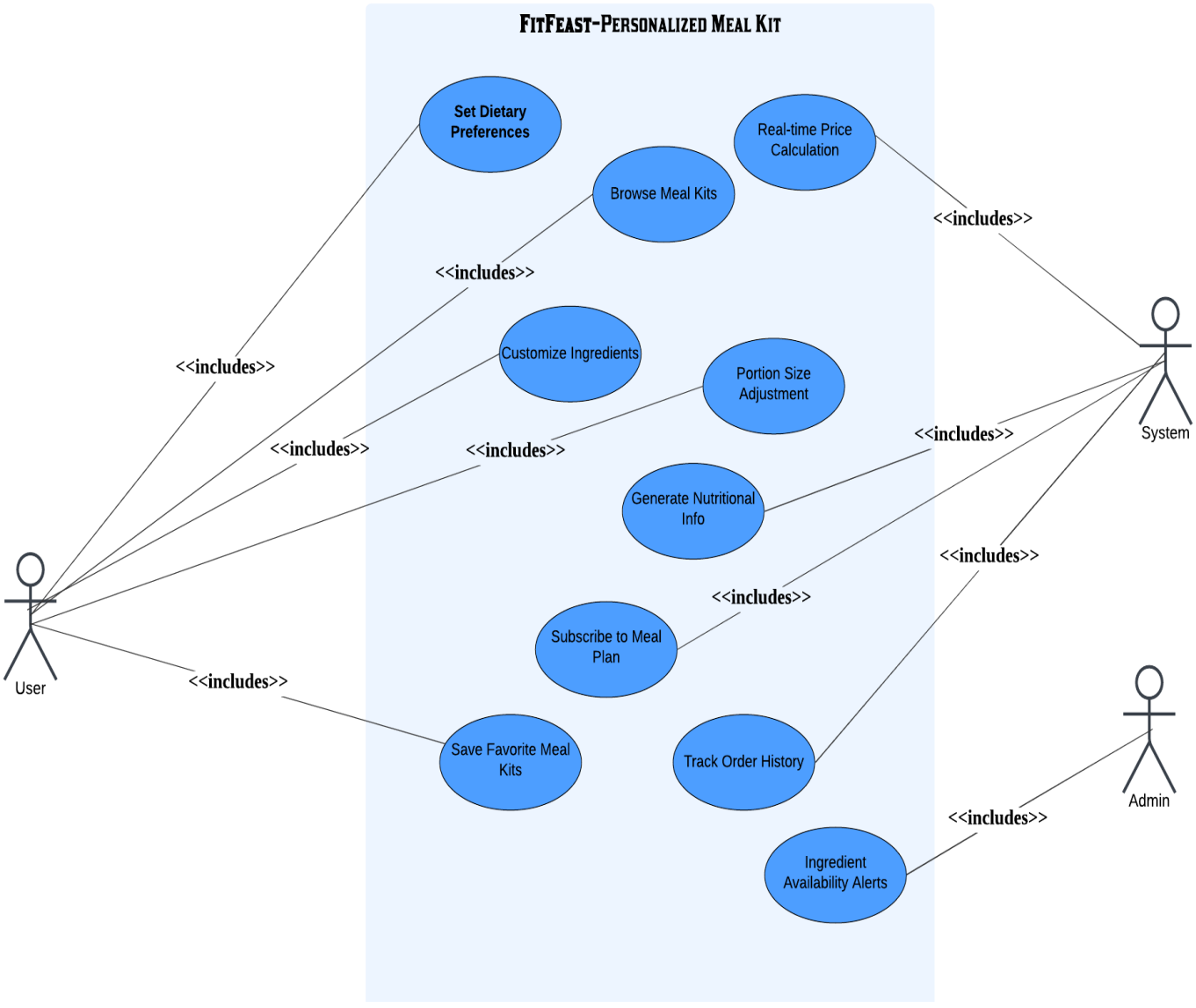

Figure 4.2.3.3 Use Case for Personalized Meal Kit

4.2.4 Description of Use Cases

Use Case: Order Meal

Main Section

- **Name:** Order Meal
- **Actors:** User, System
- **Purpose:** To allow users to select and customize their meals.
- **Description:** A user logs into the application, selects their meal preferences, customizes the meal, and places an order. The system processes the order and provides confirmation.
- **Cross References:** Functions: R1.1, R1.2

Pre-Conditions

- The user must be logged into the application.

Successful Post-Conditions

- The order is successfully recorded, and a confirmation is sent to the user.

Failure Post-Conditions

- The order is not recorded, and the user is notified of the failure.

Typical Course of Events

Actor Action	System Response
1. User logs into the application.	2. The system validates user credentials.
3. The user selects the "Order Meal" option.	4. System displays available meal options.
5. The user customizes the meal (e.g., size, ingredients).	6. The system updates the meal selection.
7. User confirms the order.	8. The system processes the order and generates a receipt.
9. User receives order confirmation.	10. The system logs the order for future reference.

Alternative Course

- Step 3: The user selects an unavailable meal; the system prompts the user to choose another option.

Use Case: Manage Caloric Intake

Main Section

- **Name**: Manage Caloric Intake
- **Actors**: User, System
- **Purpose**: To help users monitor and manage their daily caloric intake.
- **Description**: A user inputs their meal data, and the system calculates and displays the total calories consumed.
- **Cross References**: Functions: R1.1, R1.2

Pre-Conditions

- User must have an active meal entry.

Successful Post-Conditions

- User receives updated caloric information.

Failure Post-Conditions

- User receives an error message if the input is invalid.

Typical Course of Events

Actor Action	System Response
1. The user navigates to the caloric intake section.	2. System displays current caloric intake.
3. User inputs meal data.	4. System calculates total calories consumed.
5. User reviews caloric intake.	6. The system provides feedback on exceeding caloric limits.

Alternative Course

- Step 3: User enters invalid meal data; the system prompts for valid input.

Use Case: Avoid Incompatible Combinations

Main Section

- **Name**: Avoid Incompatible Combinations
- **Actors**: User, System
- **Purpose**: To prevent users from selecting incompatible meal ingredients.
- **Description**: As a user selects ingredients for a meal, the system analyzes compatibility and alerts the user if an incompatible combination is chosen.
- **Cross References**: Functions: R1.1, R1.2

Pre-Conditions

- User is customizing a meal.

Successful Post-Conditions

- User is notified of incompatible combinations.

Failure Post-Conditions

- User continues with incompatible selections without notification.

Typical Course of Events

Actor Action	System Response
1. The user selects ingredients for a meal.	2. System checks for compatibility.
3. System identifies incompatible combinations.	4. The system alerts the user to remove incompatible items.
5. User adjusts selection accordingly.	6. System confirms valid selections.

Alternative Course

- Step 3: No incompatible combinations found; user continues to finalize meal.

5. Non - Functional Requirements

5.1 Performance Requirements

- **Response Time**: The system should process user requests within 3 seconds.
- **Concurrent Users**: Support a minimum of 100 simultaneous users without performance degradation.
- **Load Handling**: Capable of handling peak loads of up to 500 transactions per minute.

5.2 Safety Requirements

- Ensure data encryption during transmission to prevent data breaches.
- Implement secure payment processing to protect users' financial information.

5.3 Security Requirements

- User authentication must use two-factor authentication.
- All sensitive data must be encrypted at rest and in transit.
- Regular security audits and vulnerability assessments should be conducted.

5.4 Reliability Requirements

- The system should have 99.9% uptime.
- Automatic failover mechanisms should be in place to ensure continuous service availability.

5.5 Usability Requirements

- The application should have an intuitive user interface that is easy to navigate.
- Provide accessibility features for users with disabilities.
- User feedback should be incorporated into design iterations.

5.6 Supportability Requirements

- The application should include logging and monitoring features for troubleshooting.
- Documentation should be available for both users and developers.
- Support should be provided via email, chat, and phone.

5.7 User Documentation

- User manuals and FAQs should be available online.
- Provide video tutorials for key features.
- Create a troubleshooting guide for common issues.

5. 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……S