

**Software requirement specification (SRS) document:**

**CSCI317 - Software Engineering – Dr. Ibrahim Dhaini**

# Project Name:

Omnifood

# Date:

March -22-2025

# Version:

1.0

# By:

Shahd Saleh Al-Sayel \_ Zainab Adnan Darwech

ID: 1123059 ID:1123246

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Version description** | **Date completed** |
|  |  |  |  |

# Review History

|  |  |  |  |
| --- | --- | --- | --- |
| **Approving party** | **Version approved** | **Signature** | **Date** |
|  |  |  |  |

# Approval History

|  |  |  |  |
| --- | --- | --- | --- |
| **Reviewer** | **Version reviewed** | **Signature** | **Date** |
|  |  |  |  |

# Table of contents:

* **1. Introduction**
* 1.1 Product Scope
* 1.2 Product Value
* 1.3 Intended Audience
* 1.4 Intended Use
* 1.5 General Description
* **2. Functional Requirements**
* **3. External Interface Requirements**
* 3.1 User Interface Requirements
* 3.2 Hardware Interface Requirements
* 3.3 Software Interface Requirements
* 3.4 Communication Interface Requirements
* **4. Non-functional Requirements**
* 4.1 Security
* 4.2 Capacity
* 4.3 Compatibility
* 4.4 Reliability
* 4.5 Scalability
* 4.6 Maintainability
* 4.7 Usability
* 4.8 Other Non-functional Requirements
* **5. Definitions and Acronyms**

# Feasibility study:

# Technical feasibility:

* Website’s frontend is built on React/Angular and backend on Node.js/Django/Flask.
* Includes AI meal suggestions, payments ,and order status updates.
* Infrastructure can support one million users with encryption and authentication in place for security.

# Economic feasibility:

* Obtained from subscriptions, one-off orders, and chargeable AI features.
* Expenditure on development, hosting, security, and API amalgamation.
* Dependent upon from customer acquisition, retention, and charge offer.

# Operational feasibility:

* Website that allows users to create custom meal plans and track deliveries is user-friendly.
* Proactive notifications and feedback system for progressive enhancement.
* Requires engagement with food suppliers and delivery companies.

# Legal feasibility:

* Complies with GDPR in data protection.
* Fulfils required standards for payments – PCI DSS.
* Needs to comply with food safety regulations and needs to clearly state terms and conditions.

# Marketing feasibility:

* Focus on the health wary, busy professionals, and competitive sports people.
* AI assisted meal personalization, sustainability, and flexible meal planning give competitive edge.
* Includes SEO, social media paid adverts, and referral campaigns.

# Scheduling feasibility:

* Complete development is estimated to take 4-6 months.

# Beta testing will happen prior to launch, followed. **The Software Development Life Cycle (SDLC):**

Agile software development life cycle (SDLC)Model.

Why Agile?

# Frequent iteration and updates:

 Omnifood needs continuous updates based on user feedback, dietary trends, and AI improvements.

 Agile’s **iterative approach** ensures **regular releases** with new features and refinements

# Customer satisfaction and flexibility:

 Agile emphasizes **customer involvement**, which is critical for personalized meal services.

 Users can request **new features (e.g., allergy filters, meal preferences)** and see them implemented in short cycles.

# Rapid delivery of features:

 Omnifood relies on AI-powered meal planning, **which needs frequent testing and optimization**.

 Agile allows **incremental improvements** to enhance AI recommendations over time

# Better risk management:

 If a **meal recommendation system fails** or user engagement drops, Agile allows **quick adjustments**.

 Continuous testing helps **identify and fix** issues before they impact users.

# Scalability and market expansion:

 Agile supports **scaling up Omnifood** by gradually expanding meal options, delivery areas, and features.

 New dietary preferences or regional cuisines can be added **without disrupting the entire system.**

# Introduction

**1.1-Product scope:**

Omnifood is an AI-powered food subscription service that delivers healthy meals based on the user's preferences and personal nutrition needs. Its goal is to make meal planning easier so you can consistently eat a variety of healthy meals. Users can get personalized meal recommendations, and track their orders all in one place. This document lists functional and non-functional requirements of the Omnifood system.

**1.2-Product value:**

* Omnifood: An AI-Powered Personalized Meal Delivery Service
* It gives you an entirely customized meal plan with recommendations.
* It has access to organic, produced locally, and fresh food in an effort to motivate healthier decisions.
* It saves your time by simplifying meal planning and cooking.
* Users can easily change delivery schedules and meal plans.
* Using reusable packaging helps to be more sustainable.

**1.3- Intended Audience:**

* People with dietary needs looking for appropriate options.
* Busy life workers who would like healthy meals and easy to use.
* Athletes who need meals tailored to what they want to achieve.
* people who are interested in sustainable, locally grown, nutritious food.

**1.4-Intended Use:**

* Give users the ability to adjust their meal plans when necessary to accommodate changing schedules or goals.
* Individualized meal subscription–with web and mobile access.
* Allow users (people with dietary needs, busy professionals, athletes, health-conscious individuals, etc.) to manage preferences, approve meal plans, track deliveries.

**1.5-General description:**

* Meals are prepared daily, ensuring quality and convenience.
* user-friendly web and mobile interfaces to track orders and manage subscriptions.
* dynamic periods of availability based on user availability and need.
* tailored menu offerings to suit a variety of tastes and adjustments.

# Functional Requirements

 The functional requirements define the core features and capabilities that the Omnifood system must provide to meet user needs and business goals. These functions are categorized by key areas of functionality.

**2.1 User Management**

Users can:

Register for an account with email verification.

Log in and log out securely.

Reset forgotten passwords.

Update their profile information (e.g., name, allergies, dietary goals).

Manage subscription status (pause, resume, or cancel).

Role-based access control ensures appropriate permissions for:

Admins: Full access to manage users, meals, analytics, and support tickets.

Chefs: Ability to add/edit/delete meals and recipes, view feedback, and collaborate with AI.

Customers: Personalized meal planning, ordering, and community interaction.

**2.2 Meal Planning**

The system enables users to:

Create, modify, or delete weekly/monthly meal plans.

Receive AI-powered meal suggestions based on preferences, past orders, and health goals.

Filter meals by dietary preferences (e.g., vegan, gluten-free) and allergens.

View detailed nutritional breakdowns (calories, macros, etc.).

**2.3 Ordering System**

The system supports:

Selection and customization of meals from a curated list.

Setting preferred delivery windows.

Real-time order tracking for deliveries.

Reordering meals from previous selections.

Managing order history and status updates.

**2.4 Payment Processing**

The system includes:

Secure online payment options via Stripe , PayPal , or other trusted gateways.

Automated billing for recurring subscriptions.

Access to payment and transaction history for users.

Integration with third-party services for secure transactions.

**2.5 Community Features**

To foster user engagement and sharing, the system allows:

Rating and reviewing meals.

Uploading personal recipes with images, ingredients, and instructions.

Commenting on and liking recipes.

Following other users for meal inspiration and shared ideas.

**2.6 AI & Analytics**

The system provides:

Smart meal recommendations using AI based on user goals, feedback, and eating habits.

Nutritional tracking tools including calorie intake and macro insights.

Weekly summary reports on meals and health progress.

Chef collaboration with AI to refine meal offerings and improve performance.

**2.7 Support & Feedback**

The system offers:

An in-app chat support system for real-time assistance.

A form-based system for submitting feedback or reporting issues (e.g., late or incorrect deliveries).

Admin ability to respond to support requests and feedback.

**2.8 Admin Functionality**

Admins have access to:

Add, edit, or remove meals and recipes.

Manage user accounts and subscription statuses.

Monitor system analytics and customer behavior.

Respond to user feedback and support tickets.

**2.9 Chef Functionality:**

Manage their profile (name, specialty, certifications).

Add, edit, or delete official meals and recipes.

View ratings and feedback on their meals.

Collaborate with the AI engine to enhance meal suggestions.

Track performance metrics (e.g., top-rated meals, popular dishes).

External Interface Requirements

**3.1-User Interface Requirements:**

* **Intuitive Design:**The interface will be designed in a simple way to enhance intuitive flow of users.
* **Screen Layouts:** The home page should show meal suggestions, order status and diet need in a structured layout. Users can tap through tabs for subscriptions, meals, and delivery.
* **Responsive:** The treated places will be responsive and therefore will work on mobile, tablet and desktop.
* **Style guide:** App should be simple with clean desired to utilize modern design and colors to reflect health. Fonts are also easy to read, and buttons are well-labelled for quick actions.
* **Swiping, drag-and-drop features**, and **tapping** are used to allow users to customize their meal plans. **Instant feedback** through pop-up modals for saving preferences or confirming orders.

**3.2-Hardware Interface Requirements:**

* **Supported Devices:** Compatible with desktops, smartphones, and tablets
* **Network Requirements:**Needs a stable internet connection (Wi-Fi or mobile data plan) for updating meal plans, processing orders, and real-time notifications.
* **Communication Protocols:** Implements HTTPS (Hypertext Transfer Protocol Secure) for secure data transfer, along with RESTful APIs (Representational State Transfer) to enable the client to interact with the backend architecture

**3.3-Software Interface Requirements:**

* **AI:** Uses AI powered algorithms to create personalized meal plans based on the user's taste and diet.
* **Backend System:** A database like PostgreSQL, MySQL, or MongoDB that securely stores user data, order history, and meal preferences.
* **Frontend Framework:** React, or Angular (used to develop an interactive and responsive user interface).
* **Backend Framework:** Node.Try to develop. js, Django or Flask to manage business logic and API communications.
* **APIs & Third-Party Services:** Integrates with **payment gateways (e.g., Stripe, PayPal),** **delivery tracking APIs**, and **authentication services (OAuth, Firebase Auth, etc.)** for secure access control.

**3.4-Communication Interface Requirements:**

* **Email and SMS Notifications:** Users receive automated updates about order confirmations, delivery status, and subscription reminders.
* **In-App Notifications:** Alerts for meal plan updates, promotions, and important service announcements.
* **Feedback and Review System:** Users can provide ratings and feedback on meals and delivery service.

# Non-Functional Requirements

**4.1-Security:**

* + Implement industry-standard encryption algorithms, such as **AES-256** for data encryption and **RSA** for secure key exchange.
  + Ensure secure payment processing using trusted third-party services (e.g., **Stripe**).

**4.2-Capacity:**

* Load balancing should be in place to distribute user requests efficiently across servers.
* The system must support up to **1 million users** for meal recommendations and order processing during peak traffic.

**4.3-Compatibility:**

* Ensure cross-browser support for major browsers, including **Chrome**, **Firefox**, **Safari**, and **Edge**, with responsive design
* The mobile app should support **iOS** and **Android** with full functionality across these versions

**4.4-Reliability:**

* Using a network of backup servers with load-balancing techniques, ensures high availability and keeps your operations running smoothly even during unexpected challenges.
* Achieve a system uptime of 99.9% to ensure continuous availability for users.

**4.5-Scalability:**

* Design the system to expand effortlessly, making it capable of supporting new markets and additional features as the project grows.

**4.6-Maintainability:**

* Utilize a modular structure in the architecture to streamline updates and efficiently resolve any issues or bugs.

**4.7-Usability:**

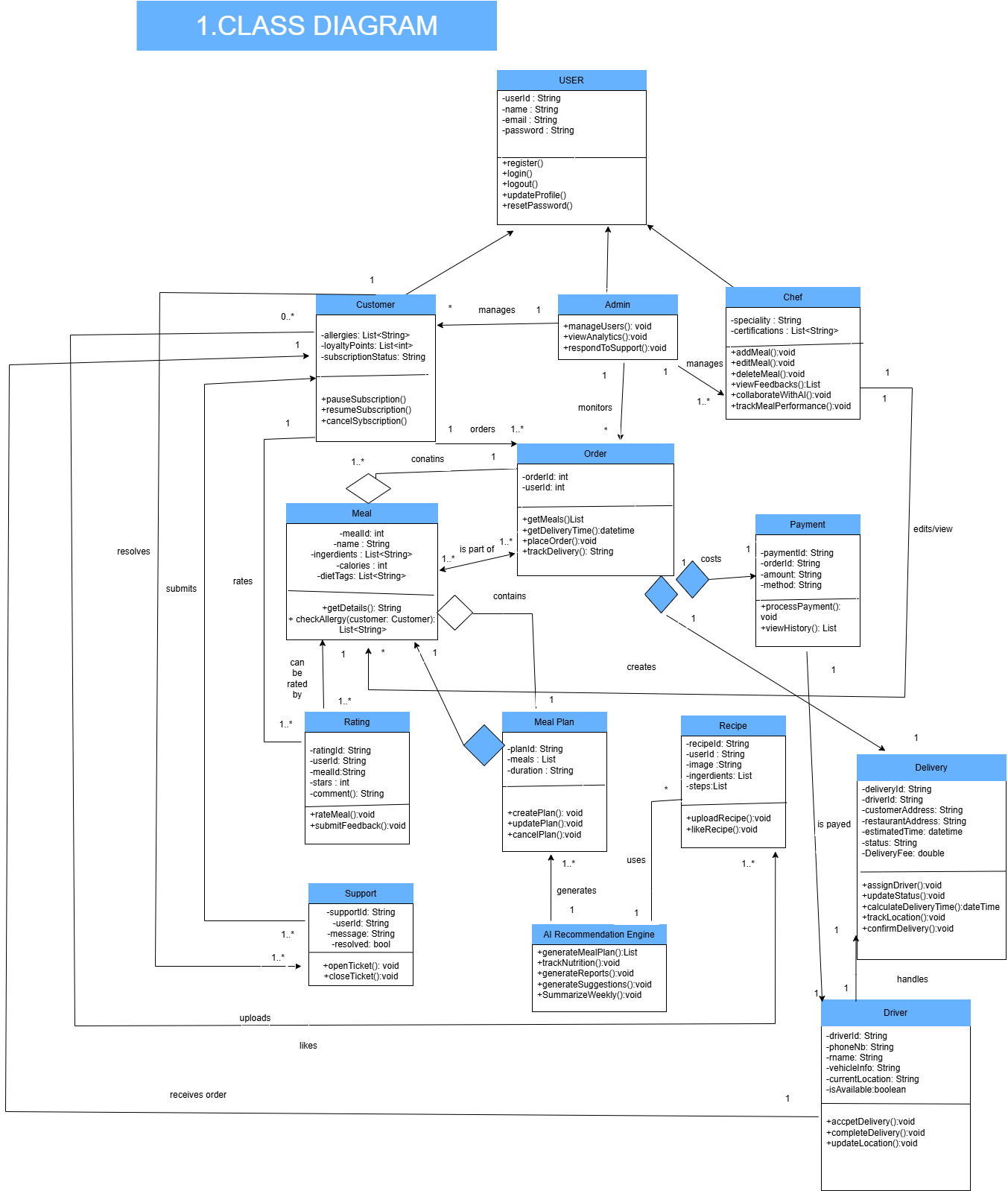
* Ensure the design is user-friendly and intuitive, reducing the time and effort users need to understand and navigate the platform.

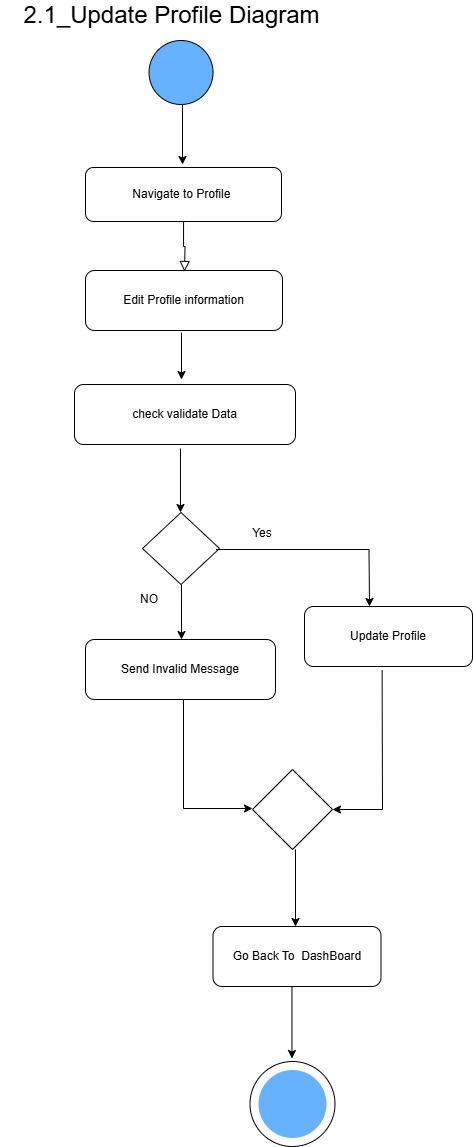
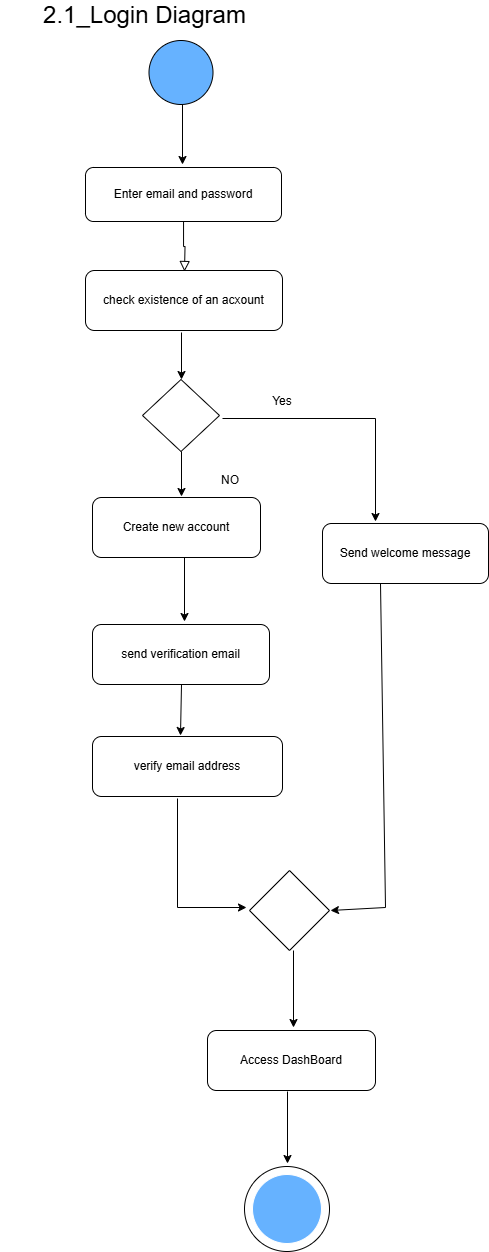
**4.8-Other Non-functional requirements:**

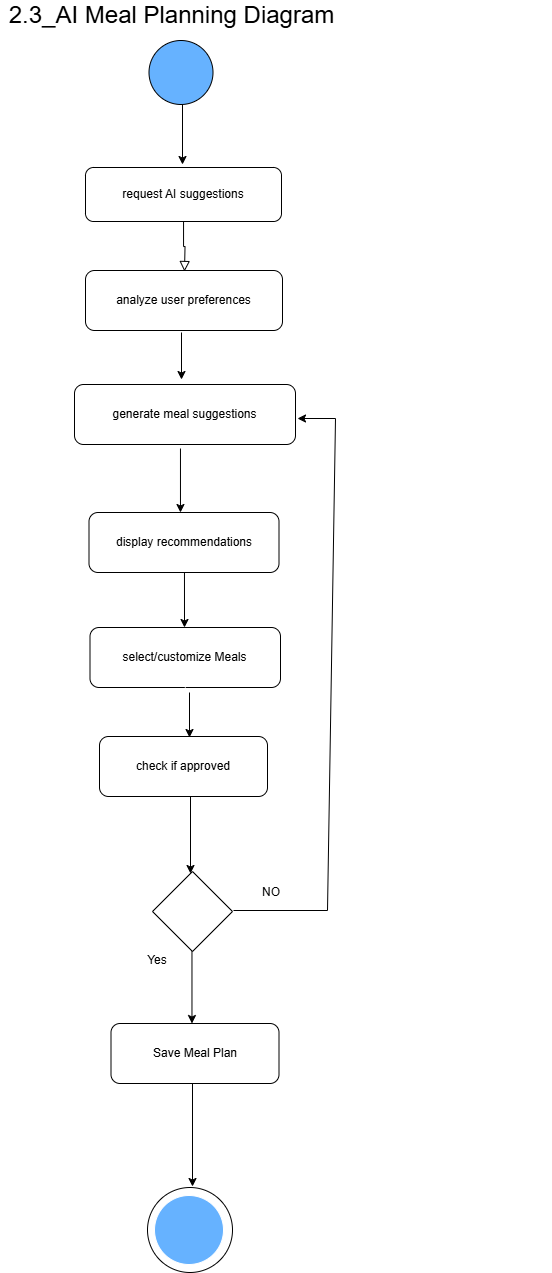
* support multiple languages, enhancing accessibility for diverse audiences.

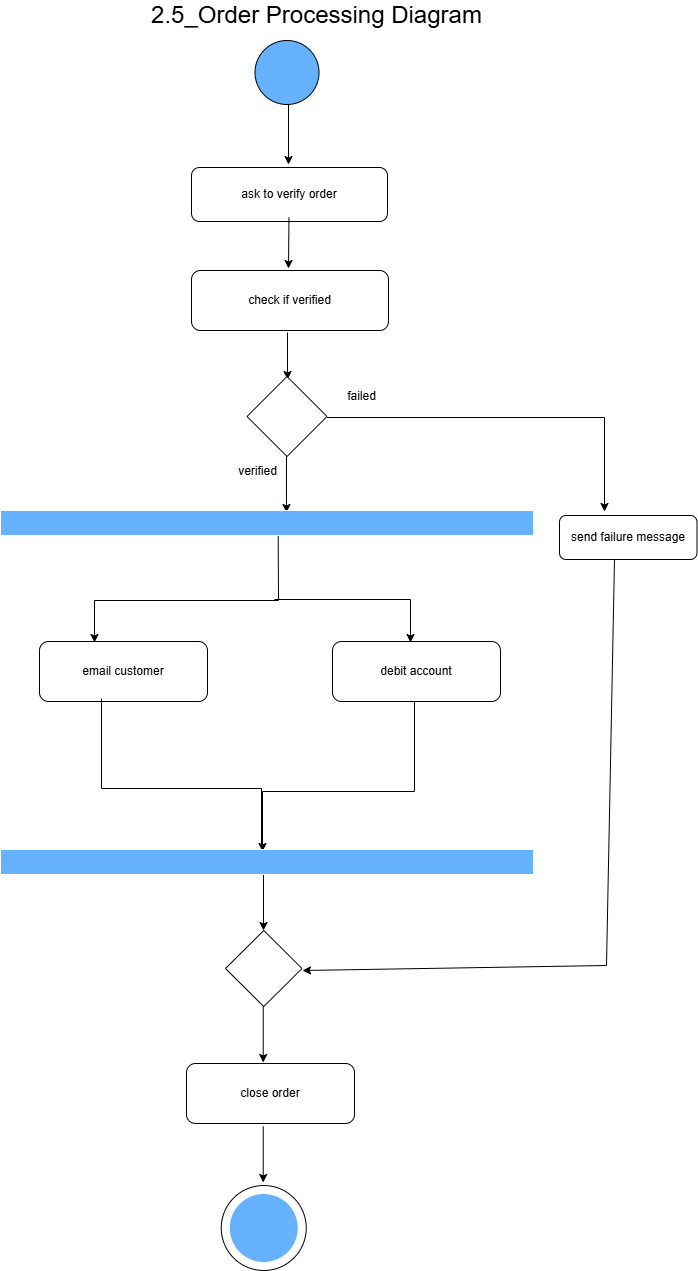
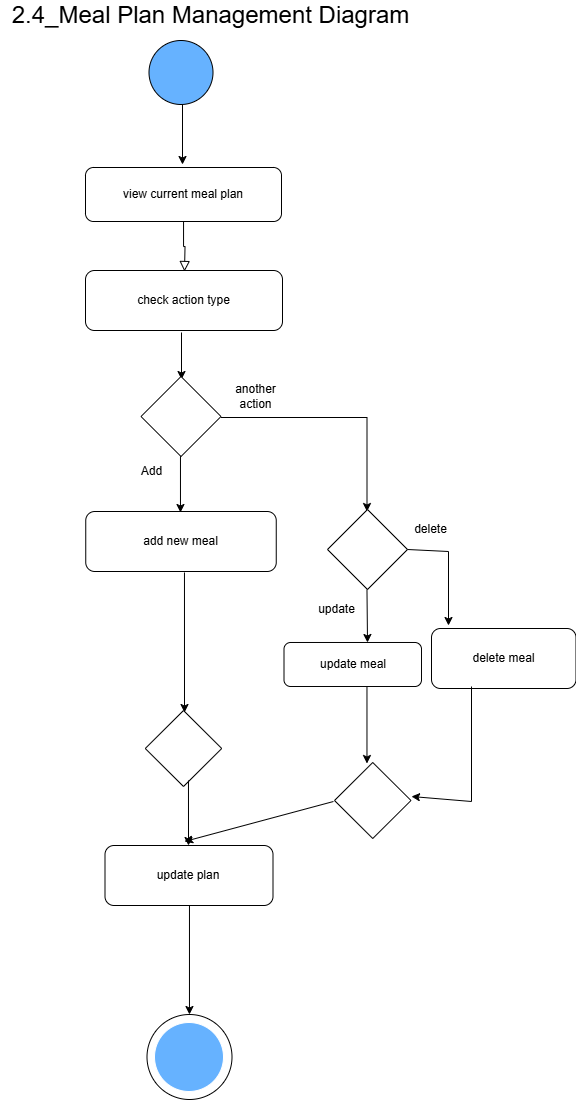
# Definition and Acronyms:

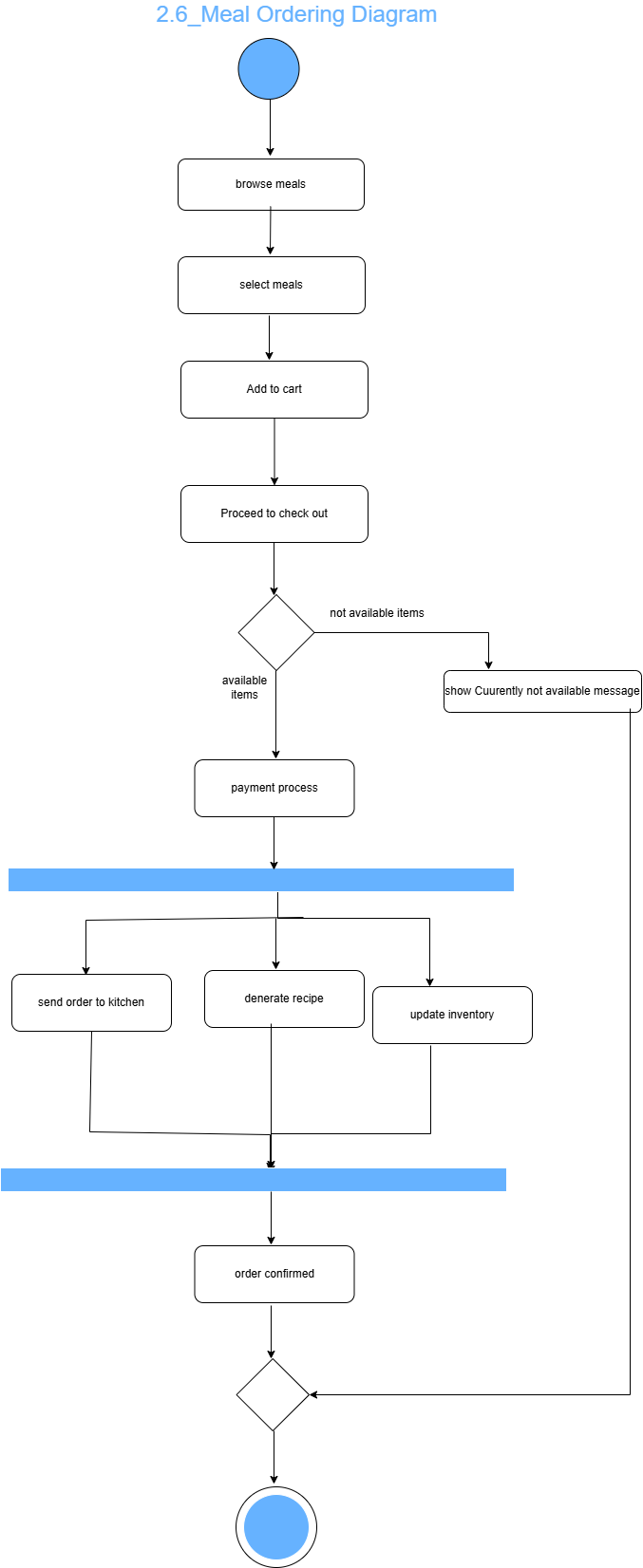
|  |  |
| --- | --- |
| **UI** | User Interface, refers to the visual and interactive aspects of the web or mobile platform. |
| **API** | Application Programming Interface, used to connect Omnifood’s backend with third-party services. |
| **HTTPS** | Hypertext Transfer Protocol Secure, ensures secure communication between users and the platform. |
| **WCAG** | Web Content Accessibility Guidelines, standards for making the platform accessible to everyone. |
| **AES** | Advanced Encryption Standard, used for secure data encryption. |
| **RSA** | Rivest–Shamir–Adleman, a cryptographic algorithm for secure key exchange. |
| **GDPR** | General Data Protection Regulation, European legislation for data protection and privacy. |
| **SEO** | Search Engine Optimization, improving website visibility in search engines. |
| **CDN** | Content Delivery Network, ensures faster delivery of web content to users. |

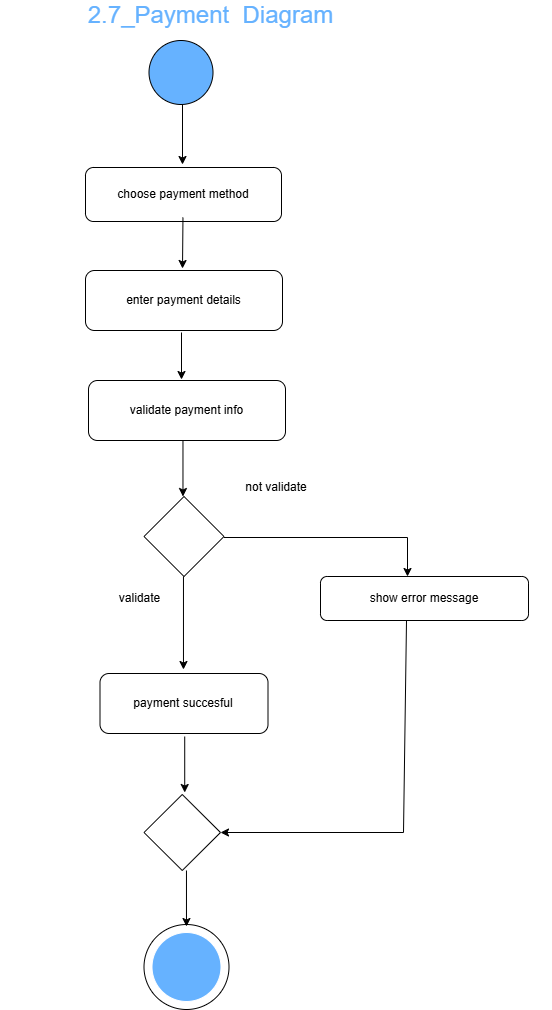
****

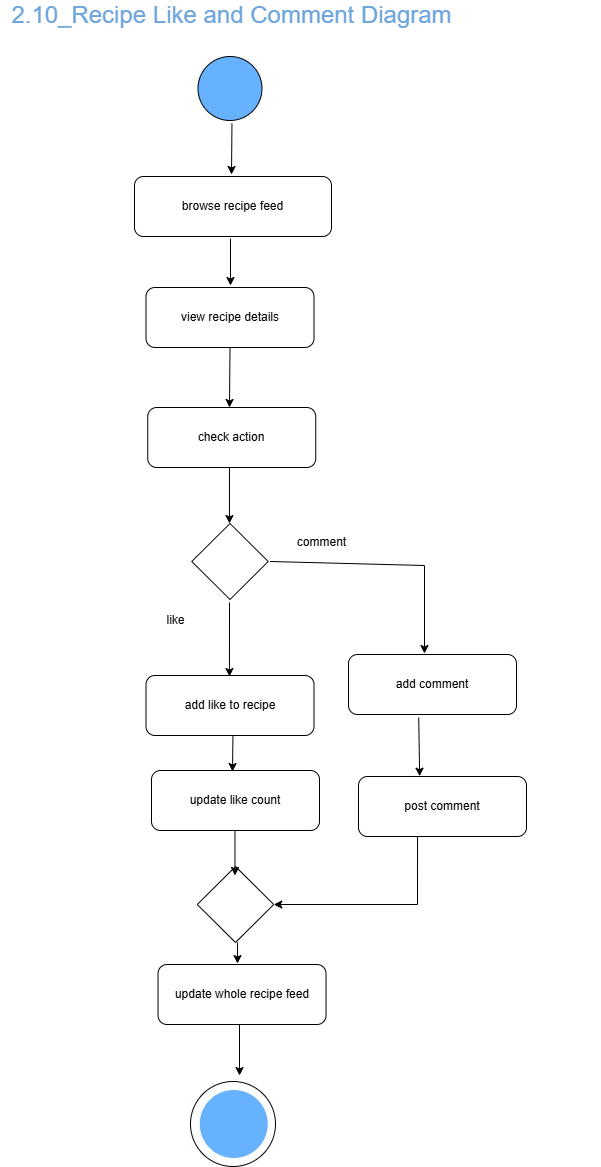
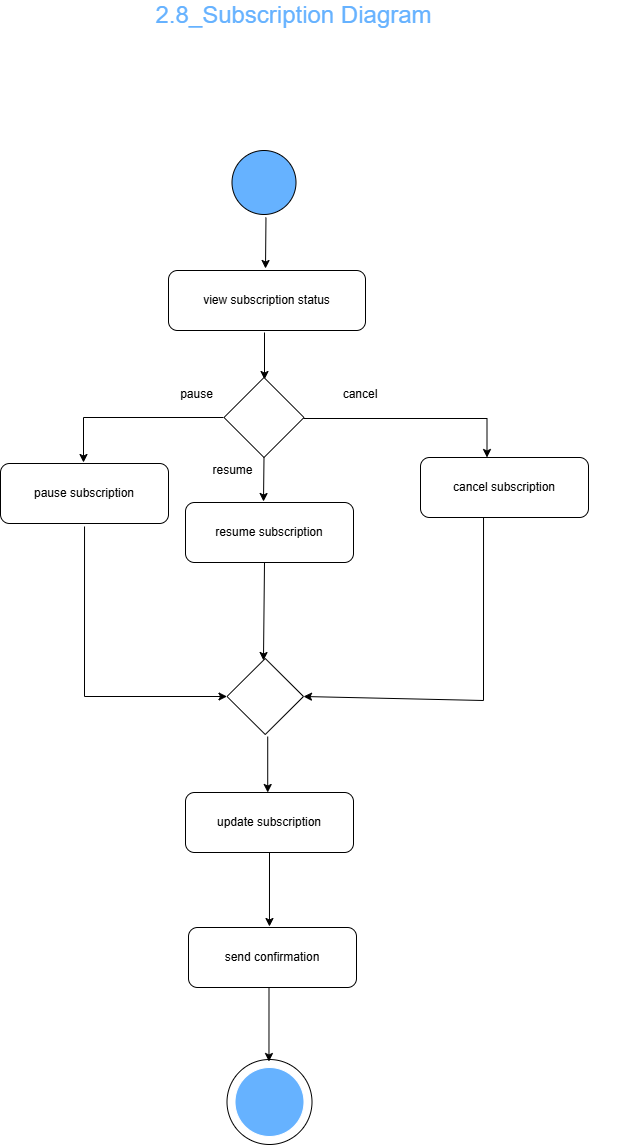
****

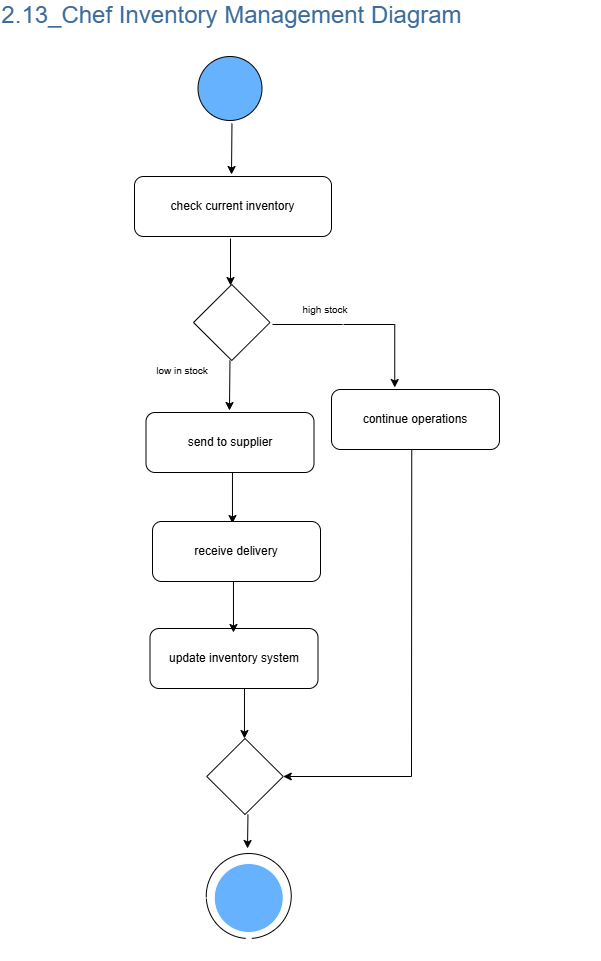
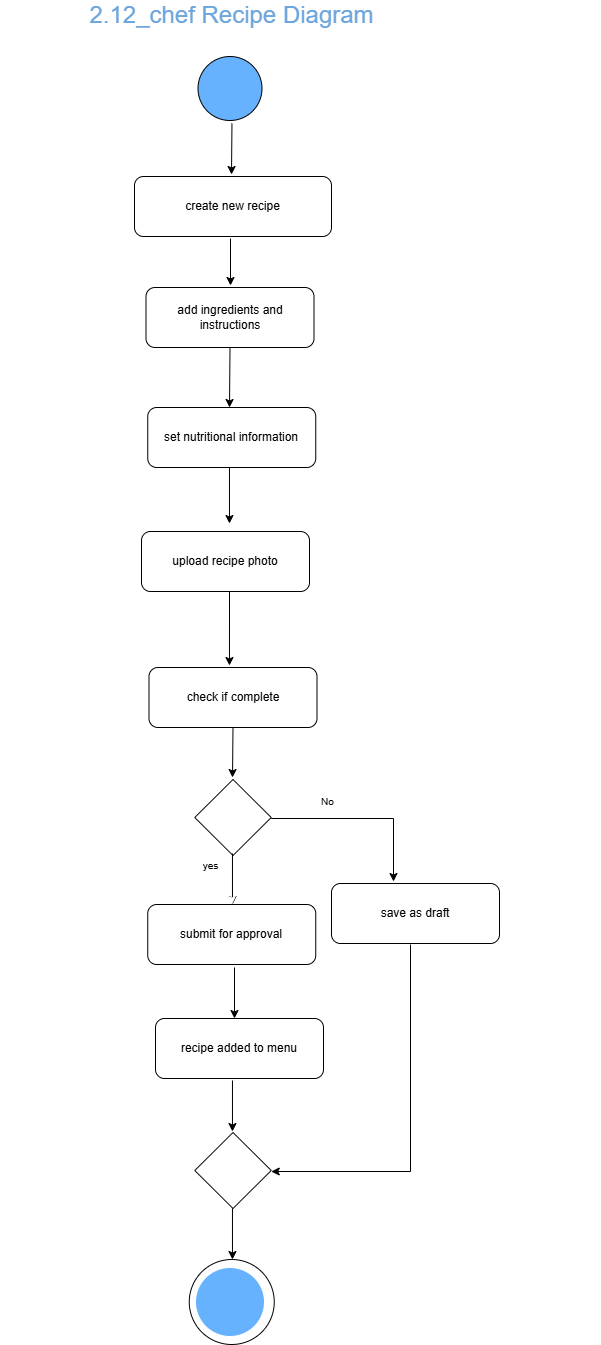
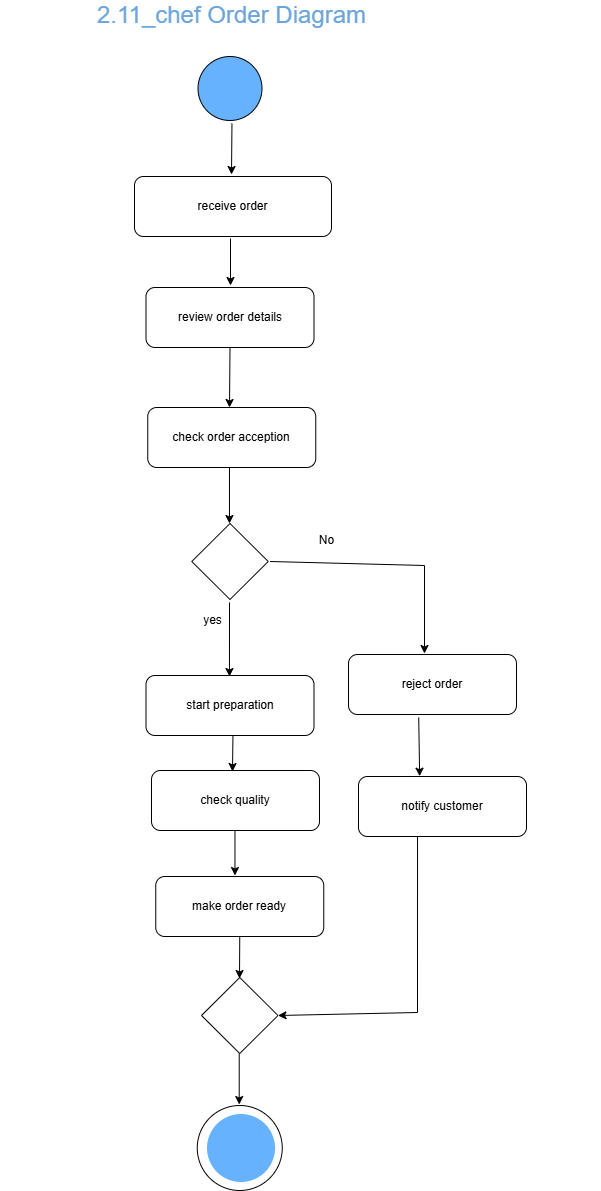
****

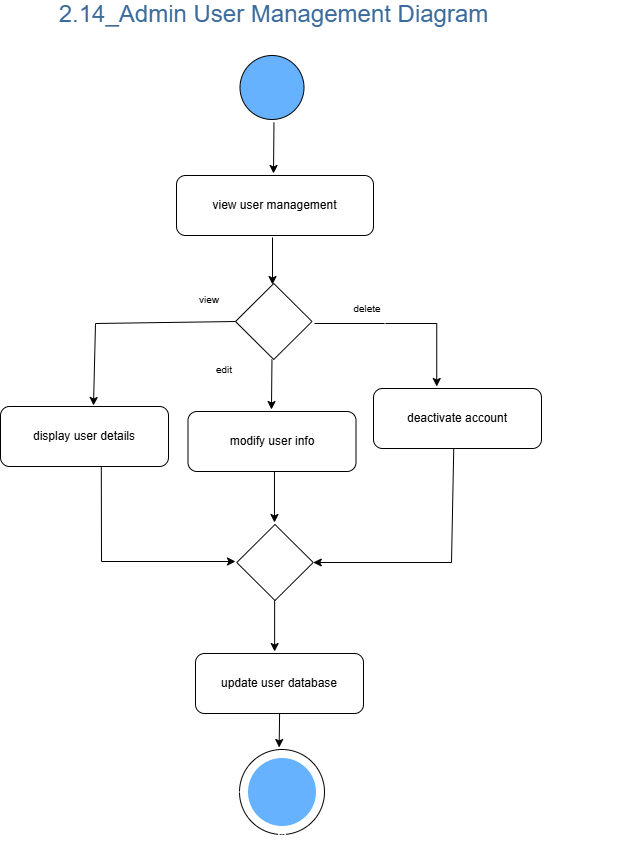
****

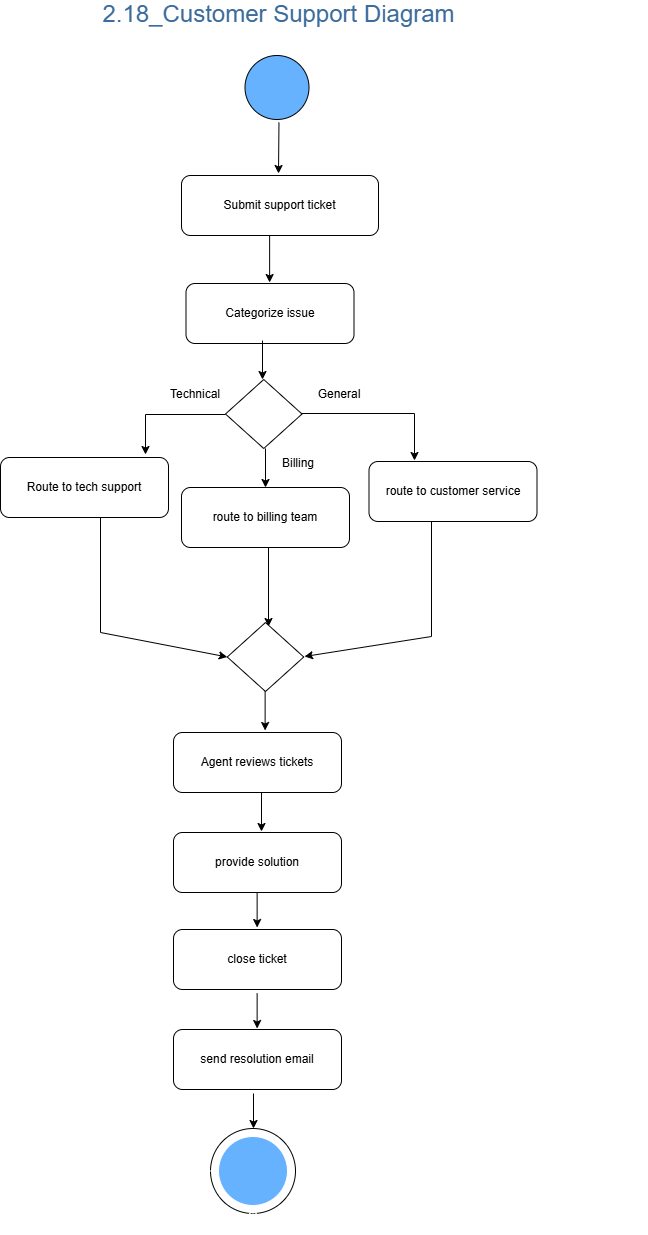
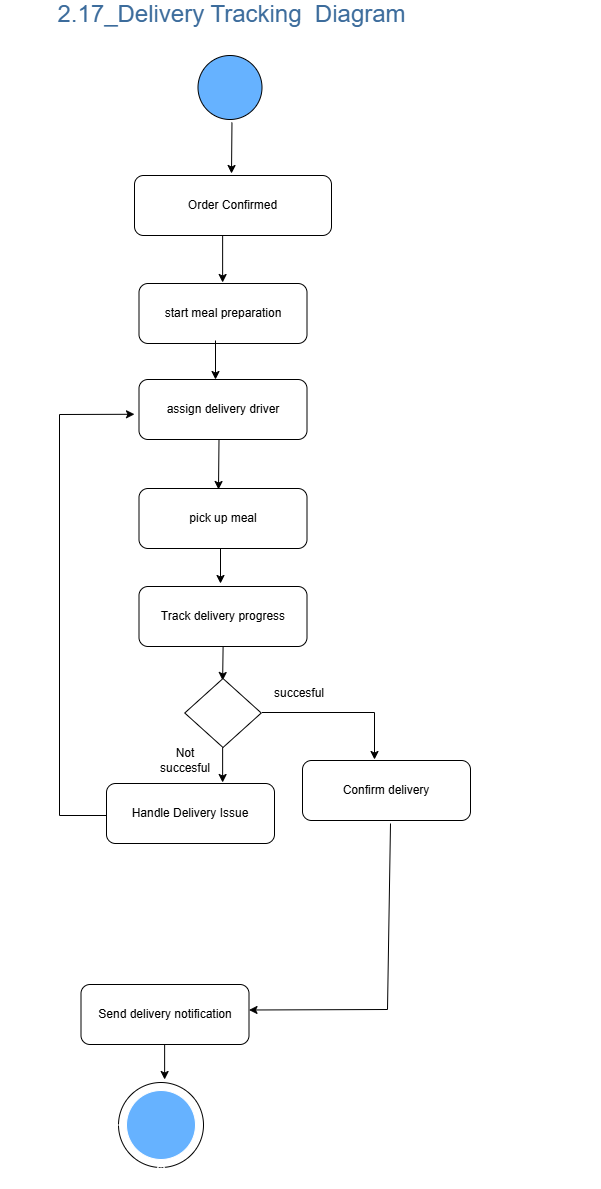
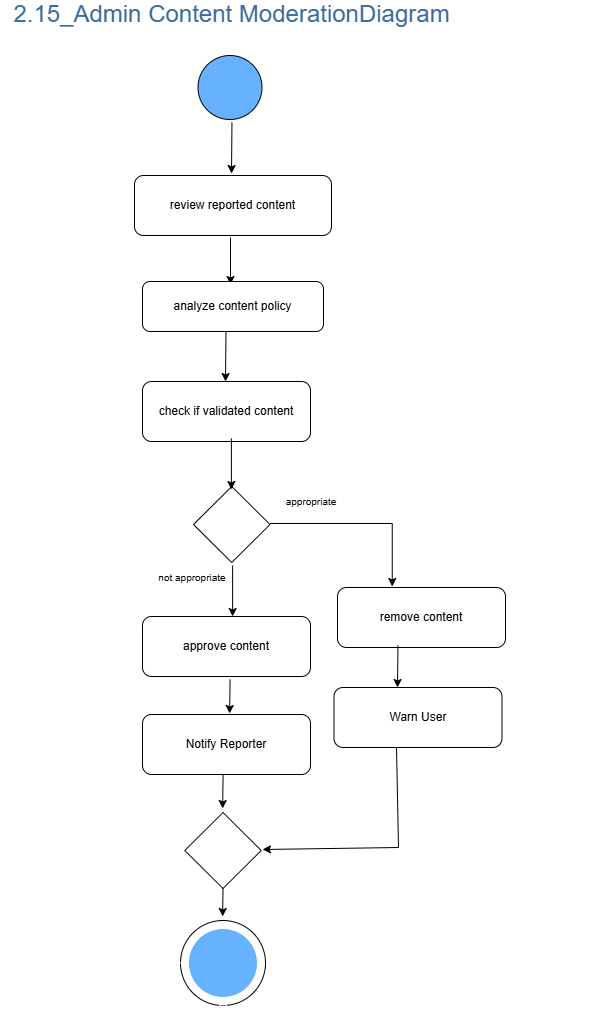
****

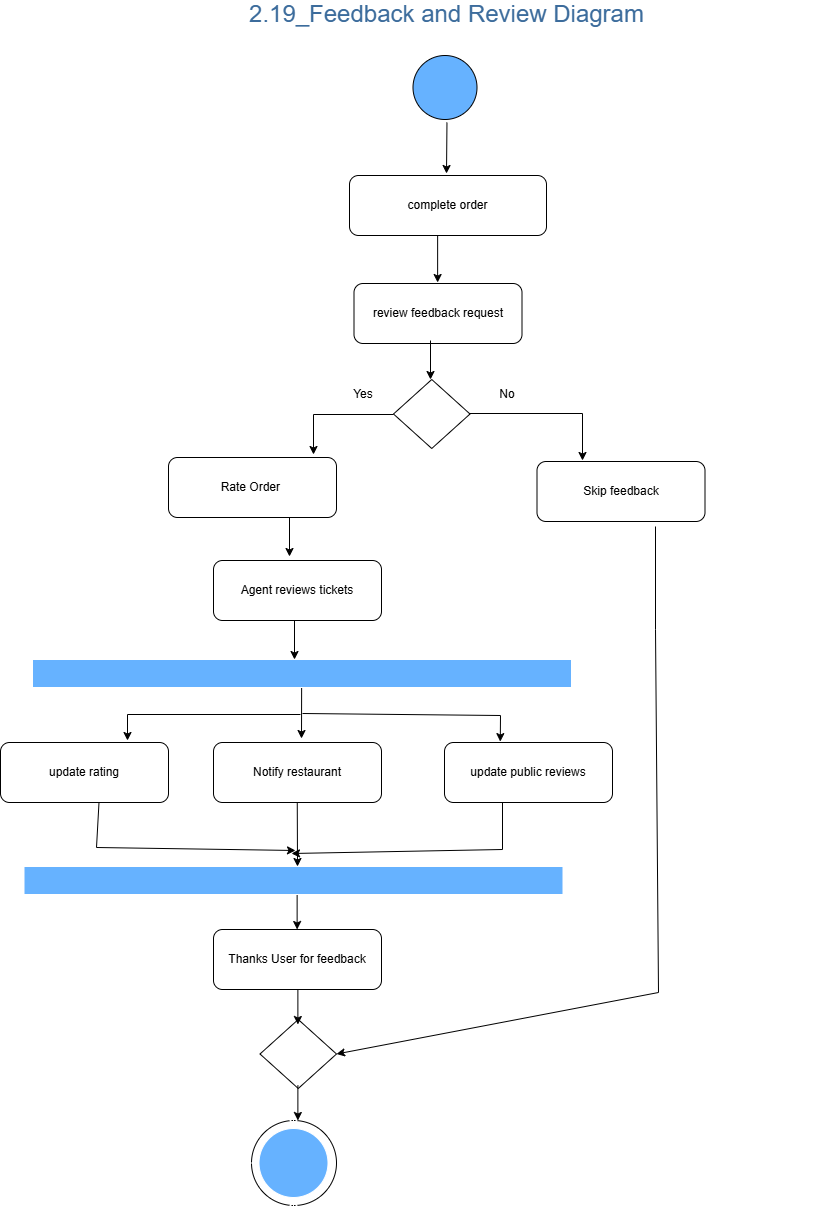
****

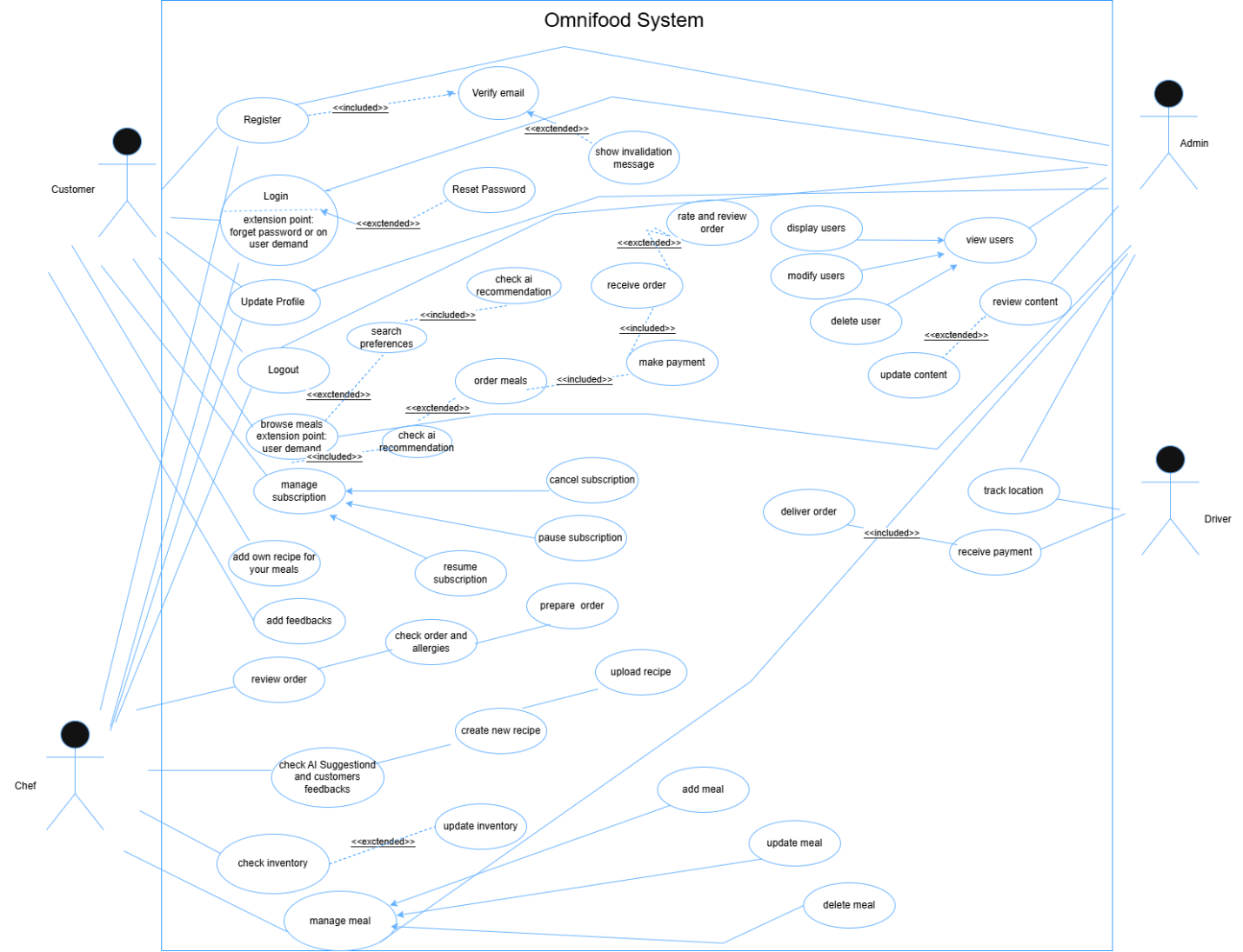


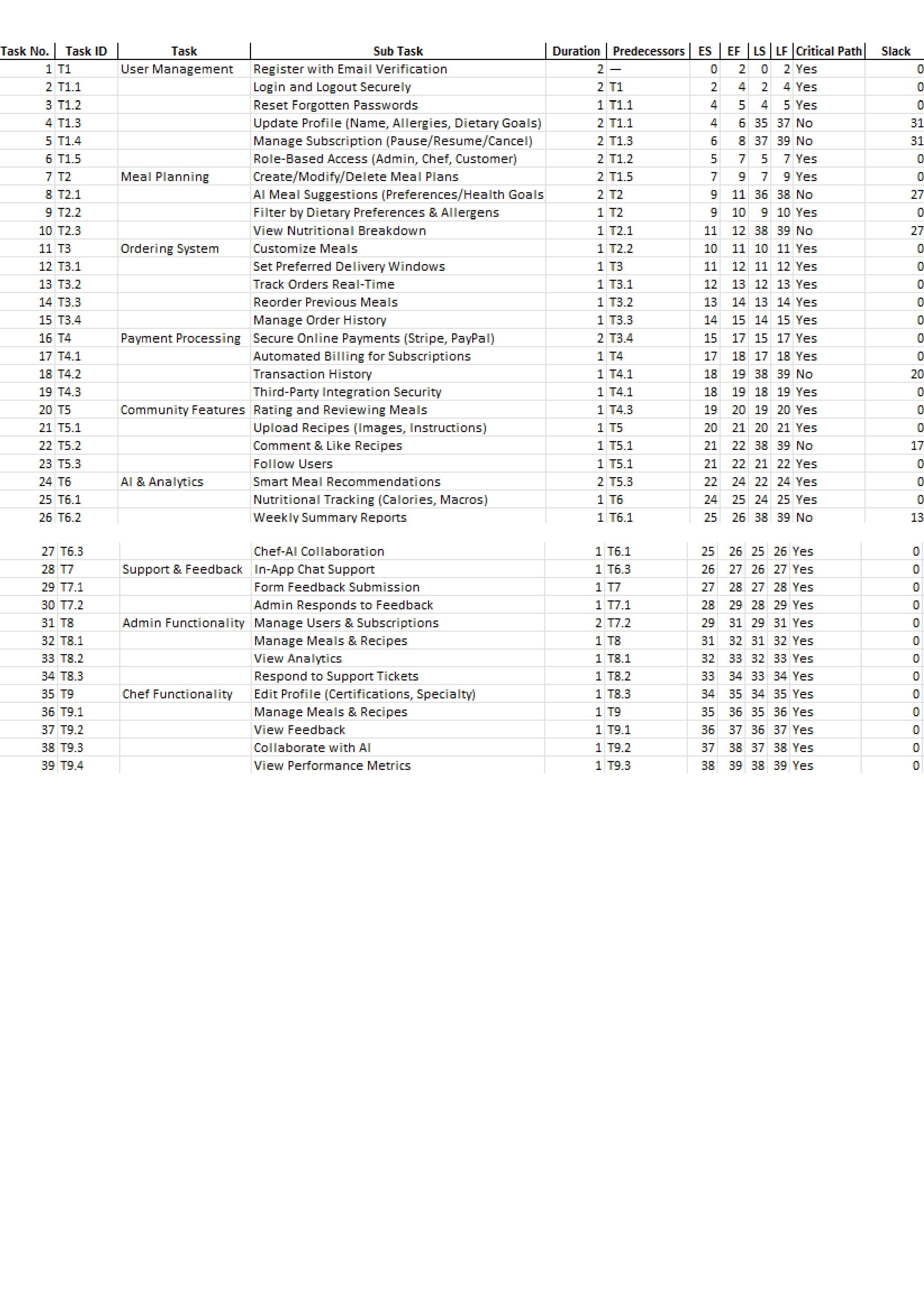
****

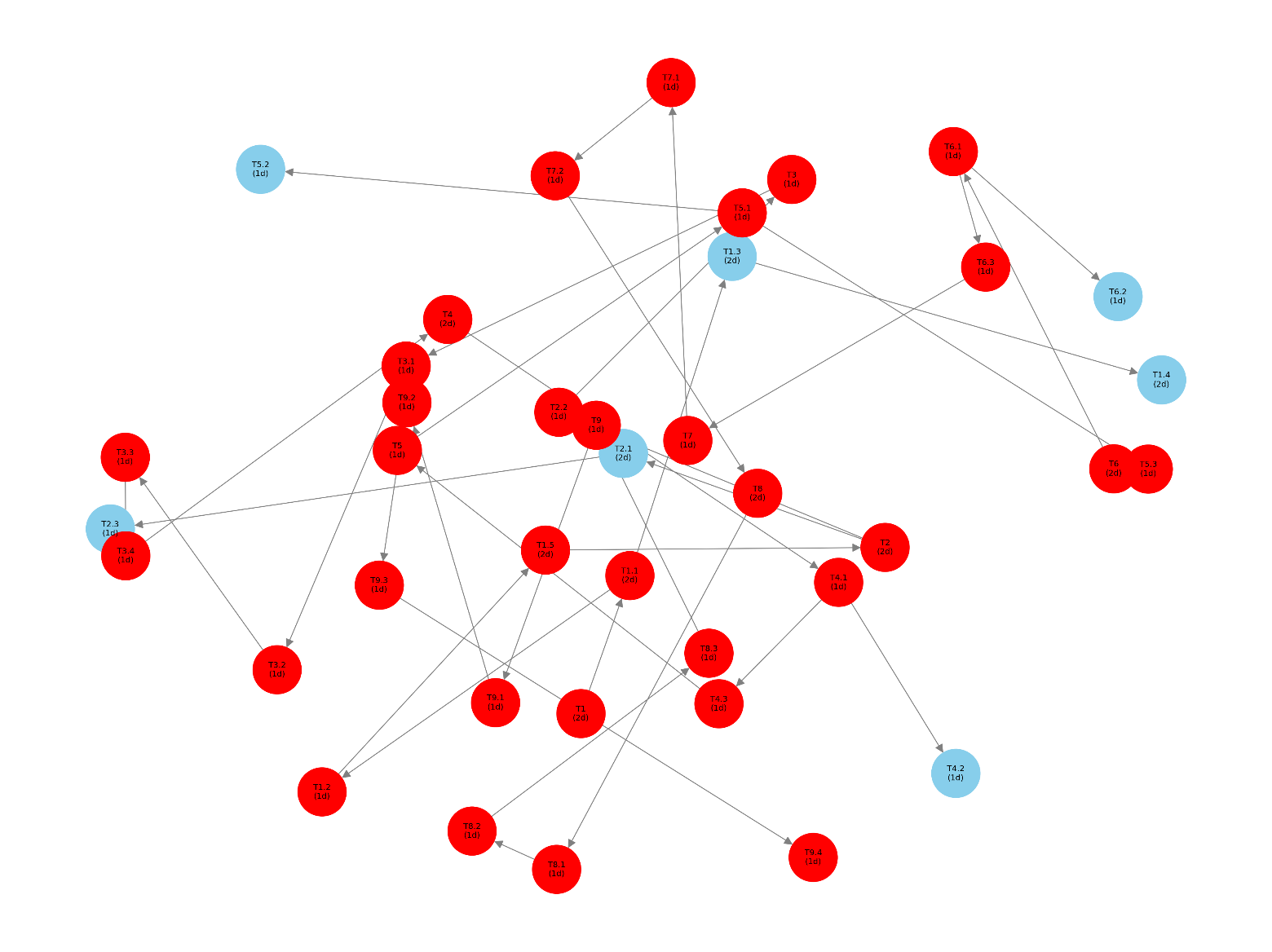
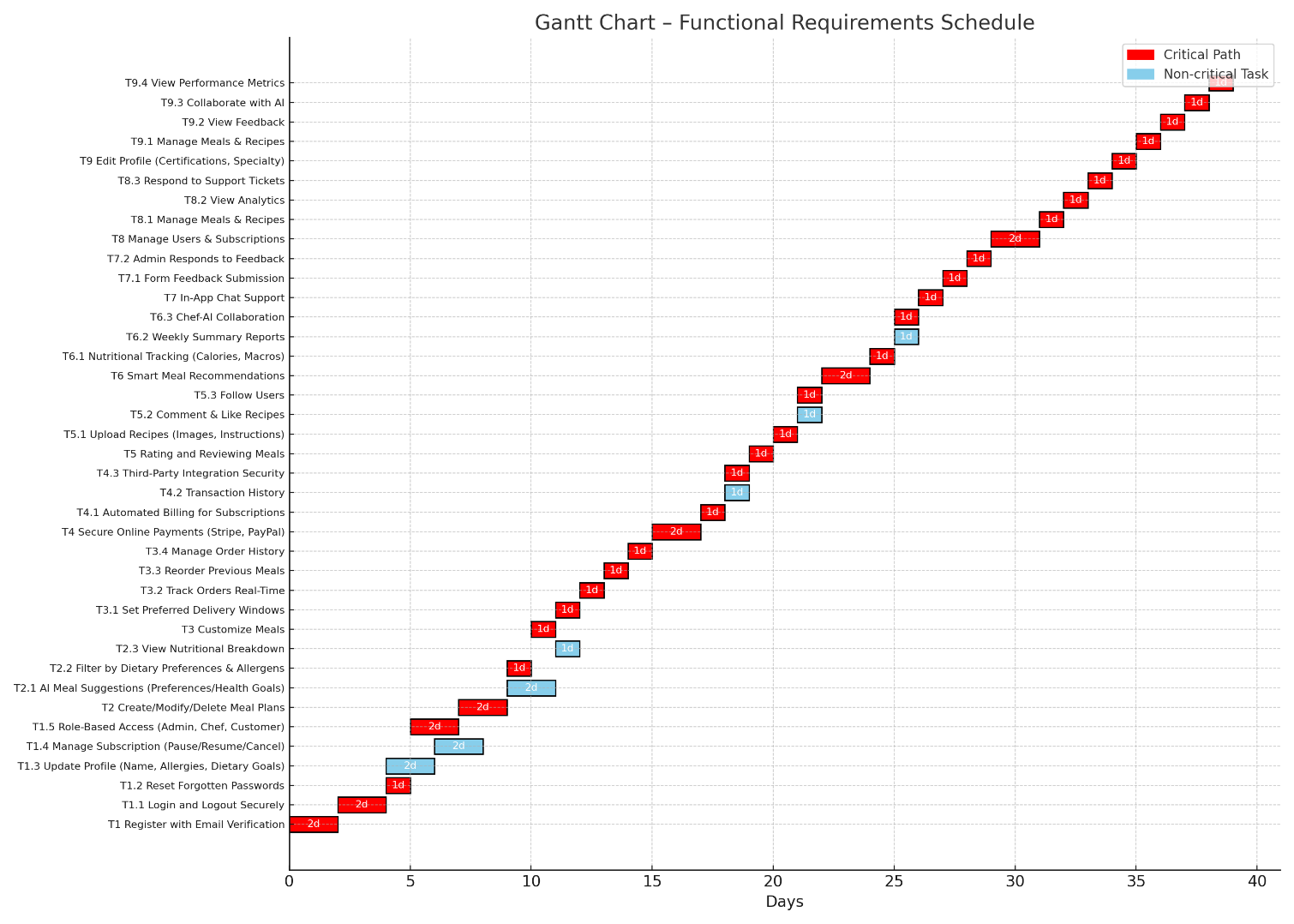
****

****

****



****

****