Software Requirements Specification

for

Annapurna : Home chef system

Version 1.0

Prepared by Project Team 2

Name	PRN
Sai Ram Yadla	240840320091
Bhairavi Nitin Chaudhari	240840320023
Yogesh Pandurang Pawar	240840320133
Shreyas Sunil Bhatkar	240840520075

Centre for Development of Advanced Computing

Raintree Marg, Near Bharati Vidyapeeth, Opp. Kharghar Railway Station, Sector 7,
CBD Belapur, Navi Mumbai - 400 614 - Maharashtra (India)
Phone: +91-22-27565303
Fax: +91-22-2756-0004

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of Annapurna is to create a platform that empowers local housewives by providing them with a source of income through their culinary skills, fostering women's empowerment and financial independence. It aims to deliver healthy, home-cooked meals to communities, including corporate employees and students, promoting nutritious eating habits. By incorporating zero-waste practices and subscription meal models, Annapurna seeks to support sustainability, reduce food wastage, and cater to customers' convenience. The project strives to bridge the gap between local cooks and nearby customers, creating a positive social and economic impact.

1.2 Intended Audience and Reading Suggestions

The intended audience for Annapurna includes local housewives looking to leverage their culinary skills for financial independence and empowerment. Corporate employees and students form another key audience segment, as they seek convenient, healthy, and affordable meal options in their vicinity. The platform also appeals to sustainability advocates who value zero-waste practices and eco-friendly initiatives. Additionally, investors and stakeholders interested in socially responsible projects, particularly those focused on women empowerment and sustainability, are vital to the project's growth.

Developers and tech enthusiasts, including software engineers, product managers, and UX/UI designers, will also benefit from the platform's technical insights. Finally, community leaders and NGOs advocating for women entrepreneurship and local business initiatives are significant stakeholders.

Readers should focus on understanding the social impact of Annapurna, particularly how the platform empowers local housewives by providing them with a sustainable income and fostering financial independence. It is essential to explore the health benefits of the platform, emphasizing how it promotes nutritious, home-cooked meals as a better alternative to fast food. The zero-waste meal models and eco-friendly packaging should be highlighted to showcase the platform's commitment to sustainability and reducing food wastage. Readers should also delve into the business model, including the subscription-based meal plans and competitive pricing strategies, to grasp how Annapurna caters to customer convenience and affordability.

For those with a technical interest, the platform's technological foundation, built with React, Spring Boot, and MySQL, along with its future roadmap, should be explored. Finally, success stories of housewives and customers will provide inspiration and a tangible demonstration of the platform's positive impact on individuals and communities.

1.3 Project Scope

The scope of the Food Ordering Service (AnnaPurna) project defines the boundaries of the project, outlining what the system will and will not do. The scope of the project includes the following:

Onboarding Local Housewives: The platform will enable local housewives to register as cooks, set up their profiles, and offer their home-cooked meals to customers in nearby areas.

Customer Segments: The primary target customers include corporate employees, students, and individuals looking for healthy and affordable meals.

Key Features:

- Zero-waste meal management to minimize food wastage.
- Subscription-based meal plans for regular customers.
- Common functionalities like order tracking, payment gateway integration, reviews and ratings, and customer support.

Geographical Reach: The platform will initially focus on local areas and expand to other regions in a phased manner.

Technology Stack: Built using React for the frontend, Spring Boot for the backend, and MySQL for database management.

Business and Social Impact: The project promotes women's empowerment by offering a source of income and reduces reliance on restaurant-based or processed meals by connecting customers with local cooks.

Future Enhancements: Potential additions include advanced analytics for cooks, AIbased meal recommendations, and partnerships with corporates and educational institutions.

1.4 References

- https://spring.io/
- https://www.javascript.com/
- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Spring Boot + React: JWT Authentication with Spring Security

2. Overall Description

2.1 Product Perspective

The Annapurna-Home Chef System is an innovative online platform that connects home chefs with customers who seek freshly prepared, home-cooked meals. The system provides a seamless experience for customers to browse, order, and subscribe to meal plans, while enabling chefs to manage menus and handle orders efficiently. The platform also integrates secure payment options and customer feedback mechanisms to enhance trust and satisfaction.

This system operates as a multi-role web application, ensuring smooth interactions between Admins, Chefs, and Customers.

2.2 Product Features

The following are the key features and requirements of the app:

- **User authentication and authorization:** This feature will allow users to log into the system using their credentials, and the system will determine their access level and permissions.
- **Role-Based Dashboards:** Separate interfaces for Admins, Chefs, and Customers.
- **Chef Management:** Admins can verify and approve chefs, ensuring quality and compliance.
- **Menu & Order Management:** Chefs can create, update, and manage their menus while processing customer orders.
- **Subscription Plans:** Customers can subscribe to meal plans (weekly/monthly) for convenient ordering.
- **Special Requests:** Users can place custom meal requests for dietary needs or special occasions.
- **Secure Payments:** Integration with a secure payment gateway for one-time purchases and subscription payments.
- **Customer Preferences & Dietary Alerts:** Customers can set preferences and receive alerts for allergens.
- **Zero-Waste Initiative:** Chefs can donate surplus meals or offer discounted canceled orders.

- **Mobile OTP & Email Notifications:** Ensures secure authentication and real-time order updates.

- **Review & Rating System:** Customers can rate and review meals to enhance platform credibility.

- **Reporting and analytics:** This feature will allow administrators to generate reports and track the progress of the assignments and the performance of the students.

- **User management:** This feature will allow administrators to manage users, including the ability to add, edit, and delete users, and assign roles and permissions.

2.3 User Classes and Characteristics

The System has three main user classes:

1. Customers

Characteristics: They are the primary users who browse meals, place orders, subscribe to meal plans, and provide feedback.

Requirements: Should be able to browse meals, place orders, manage subscriptions, make secure payments, track orders, submit reviews, and donate surplus meals.

2. Chefs

Characteristics: They are responsible for creating and updating menus, managing orders, and preparing meals.

Requirements: Must be able to create and update menus, manage orders, process payments, communicate with customers, track earnings, and handle surplus food donations.

3. Admins

Characteristics: They are responsible for managing the system, verifying chefs, and overseeing platform operations.

Requirements: Should verify chefs, oversee orders and payments, manage users, monitor platform activity, resolve disputes, and generate system performance reports.

2.4 Operating Environment

The operating environment for the project consists of the following hardware and software components:

Hardware:

- A machine with at least 8GB of RAM and a fast processor, such as Intel Core i5 or higher, to ensure smooth and efficient execution of the project.

Software:

- ReactJS for the frontend development.
- Spring Boot for the backend development
- MySQL for the database management.

- JWT Authentication for security.
- AWS for deployment.

Other Applications:

- Code editor (such as Visual Studio Code, Eclipse)
- Git version control software
- Command line interface (CLI) or terminal
- A browser for testing the application.
- Postman for testing APIs.
- MySQL Workbench or another database management tool • An AWS account for deployment.

2.5 Design and Implementation Constraints

- The platform must support high concurrency, as multiple users may browse, order, or manage menus simultaneously.
- Secure authentication must be implemented using JWT tokens and OTP verification.
- Payment transactions must comply with PCI DSS standards for financial security.
- The system should be optimized for both desktop and mobile interfaces.
- Data storage must follow GDPR compliance to protect user privacy.

2.6 User Documentation

User Guide: Covers steps for chefs to register, create menus, and manage orders.

Admin Guide: Explains how to verify chefs, oversee payments, and monitor platform analytics.

Customer Guide: Includes instructions on how to browse meals, place orders, subscribe, and manage preferences.

FAQ Section: Addresses common user queries related to ordering, payments, and refunds.

2.7 Assumptions and Dependencies

Assumptions:

- Users must have a stable internet connection to access the platform.
- The system assumes chefs provide fresh, high-quality meals and comply with hygiene standards.
- The payment gateway is assumed to be reliable and available 24/7.
- The platform relies on cloud hosting services for scalability and uptime.
- The mobile experience is assumed to work efficiently within web browsers rather than requiring a native mobile app.

Dependencies:

- A functional email service is required for notifications to be sent.
- Access to GitHub API to verify the URL of the project and retrieve the list of branches.
- An active internet connection is necessary to access the application and receive notifications.
- AWS deployment infrastructure.
- ReactJS, Spring Boot and MySQL components are required to run the application.

3. System Features

3.1 Role-Based Authentication and Authorization

3.1.1 Description and Priority

• Description: The system implements role-based authentication and authorization with three primary roles: Admin, Chef, and Customer. Admins manage platform settings, Chefs update menus and manage orders, and Customers browse, place orders, and leave reviews. JWT is used to ensure secure authentication for all users.

• Priority: High

3.1.2 Stimulus/Response Sequences

• The user logs in to the platform using their credentials. Based on their role, they are redirected to their specific dashboard. For example, Admins see platform settings, Chefs see their menus and orders, and Customers view meal options and subscriptions. JWT ensures secure session management.

3.1.3 Functional Requirements

• Role-based access control: Ensure users only access features appropriate to their role.

• JWT implementation: Use JSON Web Tokens for secure authentication and session handling.

• Dynamic dashboard redirection: Redirect users to role-specific dashboards after login.

3.2 Separate Dashboards for Each Role

3.2.1 Description and Priority

• Description: Each role (Admin, Chef, Customer) has a dedicated dashboard to access relevant functionalities. Admins can verify chefs, oversee payments, and analyze orders. Chefs can manage menus and view orders, while Customers can manage subscriptions and access support services.

• Priority: High

3.2.2 Stimulus/Response Sequences

• Upon login, users are directed to their specific dashboards. Admins see options for verification, payment, and analytics. Chefs can edit their menus and view incoming orders. Customers can browse available dishes and manage subscriptions.

3.2.3 Functional Requirements

• Dedicated dashboard for Admins: Include chef verification, payment oversight, and order analysis tools.

• Dedicated dashboard for Chefs: Provide menu management and order tracking functionalities.

• Dedicated dashboard for Customers: Allow subscription management and access to support.

3.3 Chef Story and Background

3.3.1 Description and Priority

• Description: Chefs can share personal stories and specialties to connect with customers. This feature builds trust and showcases the chef's unique culinary skills.

• Priority: Medium

3.3.2 Stimulus/Response Sequences

• The chef logs into their dashboard and clicks on "Add Story." They upload a story with images or videos highlighting their background and specialties. Customers can view these stories on the platform.

3.3.3 Functional Requirements

• Story upload: Allow chefs to share stories, including text, images, and videos.

• Customer view: Enable customers to view chef stories on the platform.

3.4 Subscription Plans

3.4.1 Description and Priority

• Description: Customers can opt for weekly or monthly subscription plans to receive meals regularly.

• Priority: High

3.4.2 Stimulus/Response Sequences

• The customer selects a subscription plan from the dashboard. The system calculates the cost and schedules deliveries based on the chosen plan.

3.4.3 Functional Requirements

• Plan selection: Allow customers to choose from available subscription plans.

• Delivery scheduling: Automate meal delivery based on the subscription plan.

• Payment processing: Handle payments for subscription plans securely.

3.5 Order Management

3.5.1 Description and Priority

• Description: Customers can pre-book orders for group events or gatherings.

• Priority: Medium

3.5.2 Stimulus/Response Sequences

• The customer selects the "Pre-Order" option and specifies the group size and event date. The system confirms the booking and updates the chef's dashboard.

3.5.3 Functional Requirements

• Pre-order booking: Enable customers to book meals in advance for groups. • Chef notification: Notify chefs of group bookings.

3.6 Universal Meal Preferences

3.6.1 Description and Priority

• Description: Customers can set meal preferences, including dietary restrictions, to ensure meals align with their values and needs.

• Priority: Medium

3.6.2 Stimulus/Response Sequences

• The customer sets preferences (e.g., vegetarian, gluten-free) in their profile. The system ensures only suitable meals are displayed.

3.6.3 Functional Requirements

• Preference setting: Allow customers to specify dietary restrictions.

• Meal filtering: Display meals that match customer preferences.

3.7 Special Occasion Requests

3.7.1 Description and Priority

• Description: Customers can request custom garnishes or meal presentation for special occasions.

• Priority: Low

3.7.2 Stimulus/Response Sequences

• The customer adds a special request during order placement. The system notifies the chef to accommodate the request.

3.7.3 Functional Requirements

• Request submission: Enable customers to specify special requests.

• Chef notification: Inform chefs of special requests in real time.

3.8 Dietary Alerts

3.8.1 Description and Priority

• Description: The system alerts chefs about customer allergies or dietary restrictions to ensure safer meal preparation.

• Priority: High

3.8.2 Stimulus/Response Sequences

• The customer sets allergies in their profile. When placing an order, the system alerts the chef about these restrictions. 3.11.3 Functional Requirements • Allergy tracking: Store and retrieve customer allergy information. • Chef alerts: Notify chefs of allergies during meal preparation.

3.9 Zero-Waste and Donation Initiatives

3.9.1 Description and Priority

• Description: Implement a zero-waste initiative by partnering with food banks or NGOs. Chefs can donate surplus meals or offer discounted rates for canceled meals. • Priority: Low

3.9.2 Stimulus/Response Sequences

• The chef marks surplus meals as "donate" or "discount." The system notifies the food bank or displays discounted rates to customers.

3.9.3 Functional Requirements

• Donation tracking: Log meals donated to food banks.

4. External Interface Requirements

4.1 User Interfaces

- The platform will have a modern, responsive UI designed using **React.js** and Bootstrap for seamless navigation across desktop and mobile devices.
- **Key UI Components**:
    - **Admin Dashboard**: Features include chef verification, payment monitoring, and analytics.
    - **Chef Dashboard**: Displays real-time order management and menu customization options.
    - **Customer Dashboard**: Allows meal browsing, order tracking, and subscription management.
        - User-friendly forms with validation for login, registration, order placement, and payment.

4.2 Hardware Interfaces

- **Server-Side Requirements**:
    - A machine with at least 8 GB RAM, Intel Core i5 or higher processor.
    - Hosted on a local server or dedicated hosting with MySQL backend.
- **Client-Side Requirements**:
    - Desktop or mobile devices capable of running modern web browsers (e.g., Chrome, Edge).
    - Input/output devices such as keyboard, mouse, and touchscreens for interaction.

4.3 Software Interfaces

- **Backend**: Java Spring Boot for RESTful API development.
- **Frontend**: React.js for dynamic, interactive UI.
- **Database**:
    - **MySQL** for structured data (users, orders, payments).
- **Authentication**: JWT for secure, role-based access.
- **Third-Party Integrations**:
    - Razorpay/Stripe for secure payments.

- OTP service for user verification.
- Email services (e.g., SendGrid) for notifications.

4.4 Communications Interfaces

- The platform uses **HTTP/HTTPS protocols** for secure client-server communication.
- **API Calls**: React frontend interacts with Spring Boot backend via RESTful APIs.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **Response Time**: API response time should not exceed 500ms under normal load.
- **Scalability**: The system should support up to 1,000 simultaneous users without degradation in performance.
- **Reliability**: Ensure 99.9% uptime for the application.

5.2 Safety Requirements

- Robust error-handling mechanisms to prevent data corruption or loss during order placement or payments.
- Graceful degradation of features in case of system failures, ensuring essential functions like browsing and order history remain accessible.

.

5.3 Security Requirements

- **Role-Based Access Control:** Limit access to resources and data based on user roles (Admin, Chef, Customer).
- **Authentication:** A secure authentication system that ensures only authorized users can access the system.
- **Authorization:** An authorization mechanism to determine what actions a user can perform within the system based on their role and permissions.
- **Encryption:** Data in transit and at rest should be encrypted to protect sensitive information from being intercepted or accessed by unauthorized parties.

5.4 Software Quality Attributes

The software quality attributes for the project include the following:

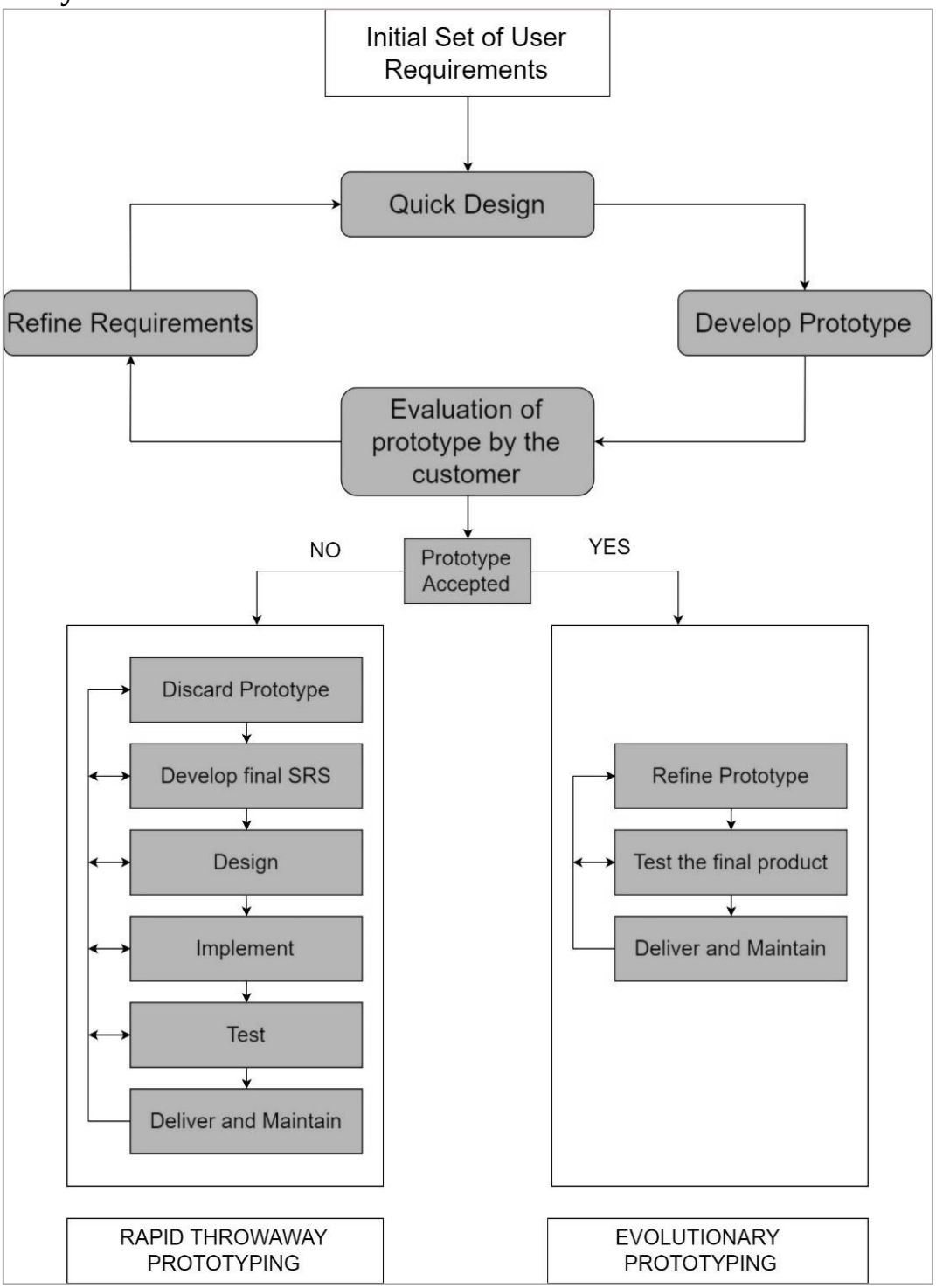
- **Usability**: Easy-to-navigate interface for all user roles, ensuring minimal learning curve.
- **Maintainability**: Modular code structure with comprehensive documentation for easy updates.
- **Scalability**: Designed to accommodate growing users and data with minimal changes to infrastructure.
- **Interoperability**: Seamless integration with third-party APIs for payments, notifications, and real-time data handling.
- **Reliability:** The system should be reliable and provide accurate results, ensuring that assignments are submitted and reviewed accurately and efficiently.
- **Performance:** The system should perform efficiently and respond quickly to user requests, allowing for fast and seamless assignment submissions and reviews.
- **Scalability**: The system should be scalable, allowing for an increasing number of users and assignments as the demand grows.

Appendix A: Glossary

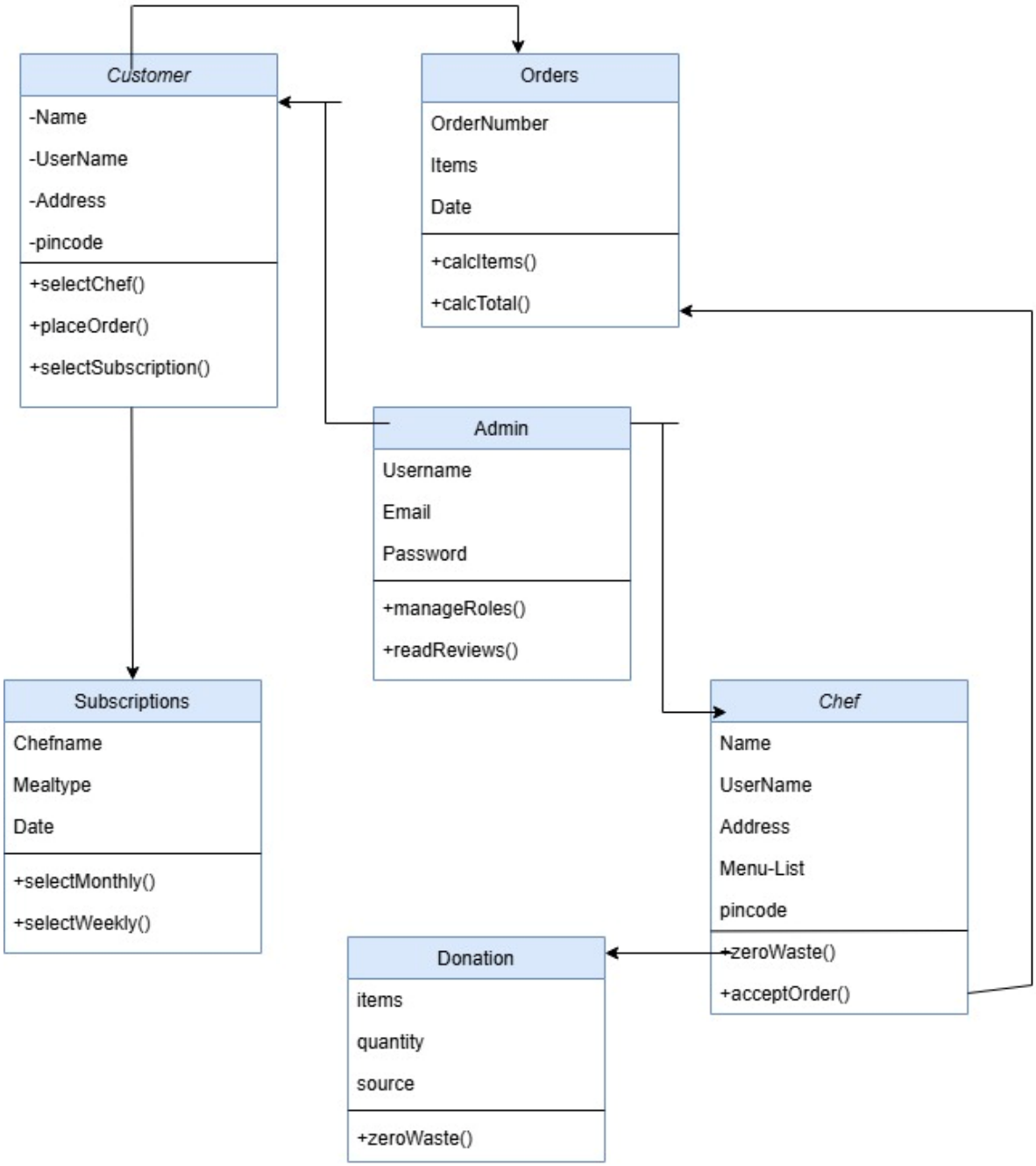
Sr. No.	Acronym	Full Form
1	GB	Gigabyte
2	HTML	Hypertext Markup Language
3	OS	Operating System
4	AWS	Amazon Web Service
5	JWT	Java Web Token
6	CLI	Command Line Argument
7	SQL	Structured Query Language
8	SRS	Software Requirement Specification
9	ER	Entity Relationship
10	HTTP / HTTPS	Hypertext Transfer Protocol / Secure
11	ID	Identification
12	URL	Uniform Resource Locator
13	API	Application Programming Interface
14	RAM	Random Access Memory
15	JS	JavaScript

Appendix B: Analysis Models

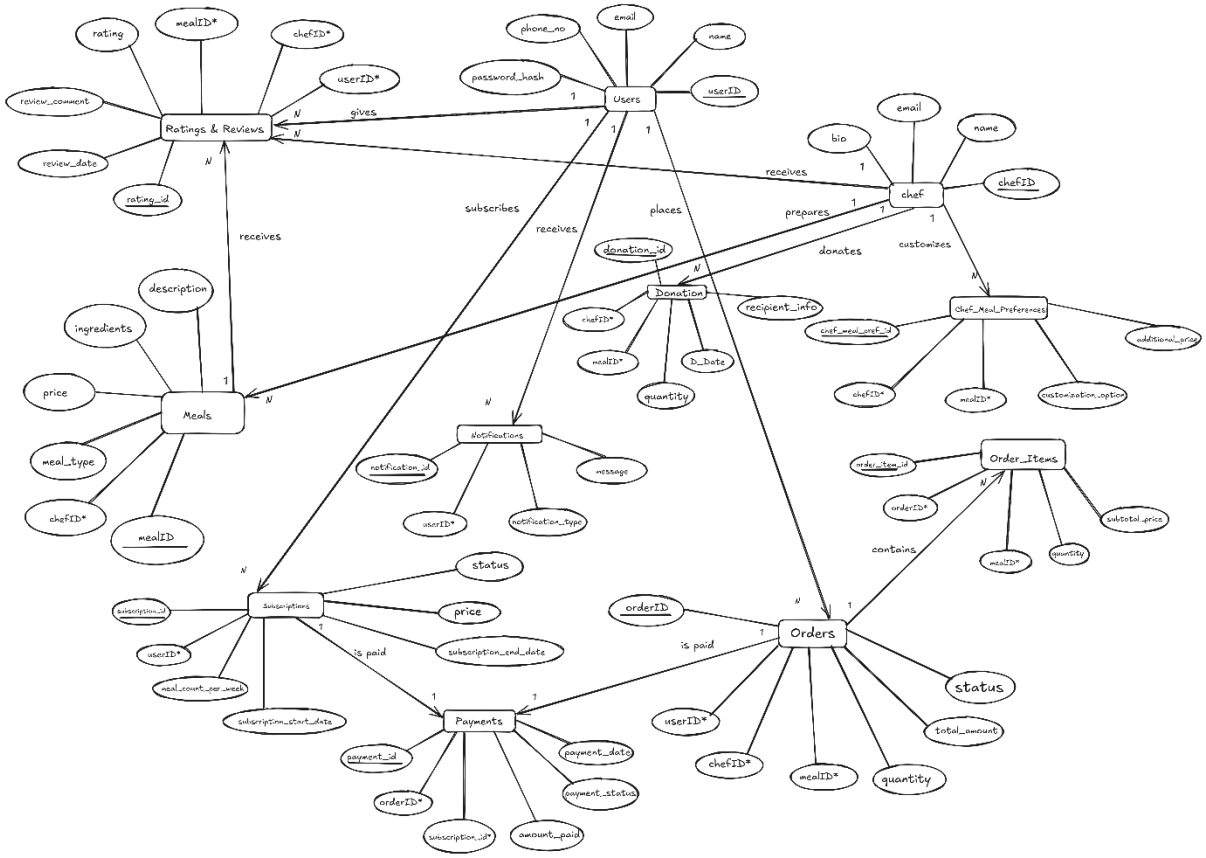
- **Software Development Approach in Our System**

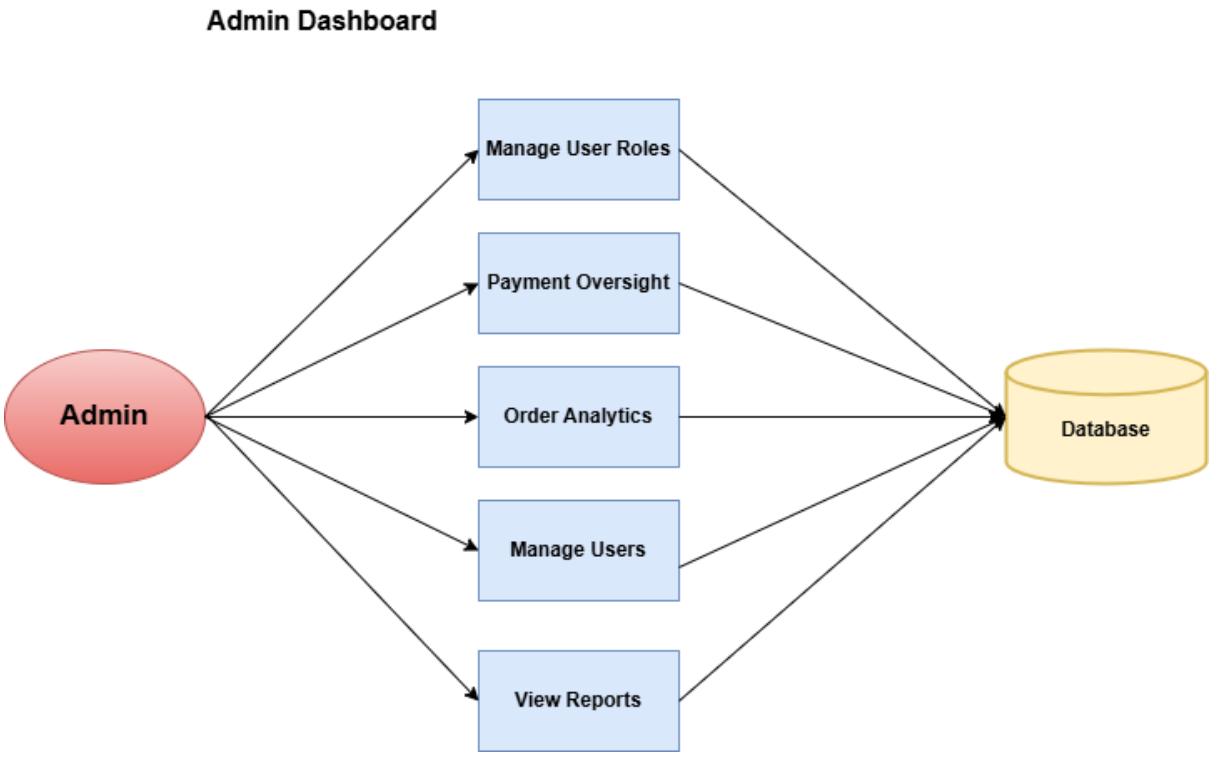


Class Diagram

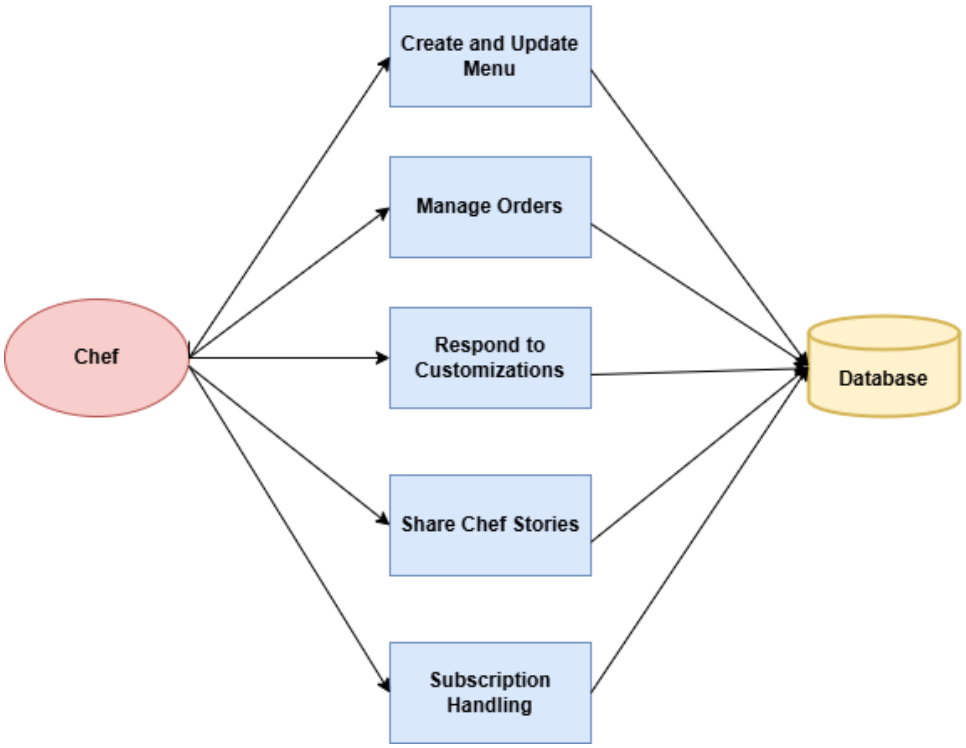


- **ER Diagram**

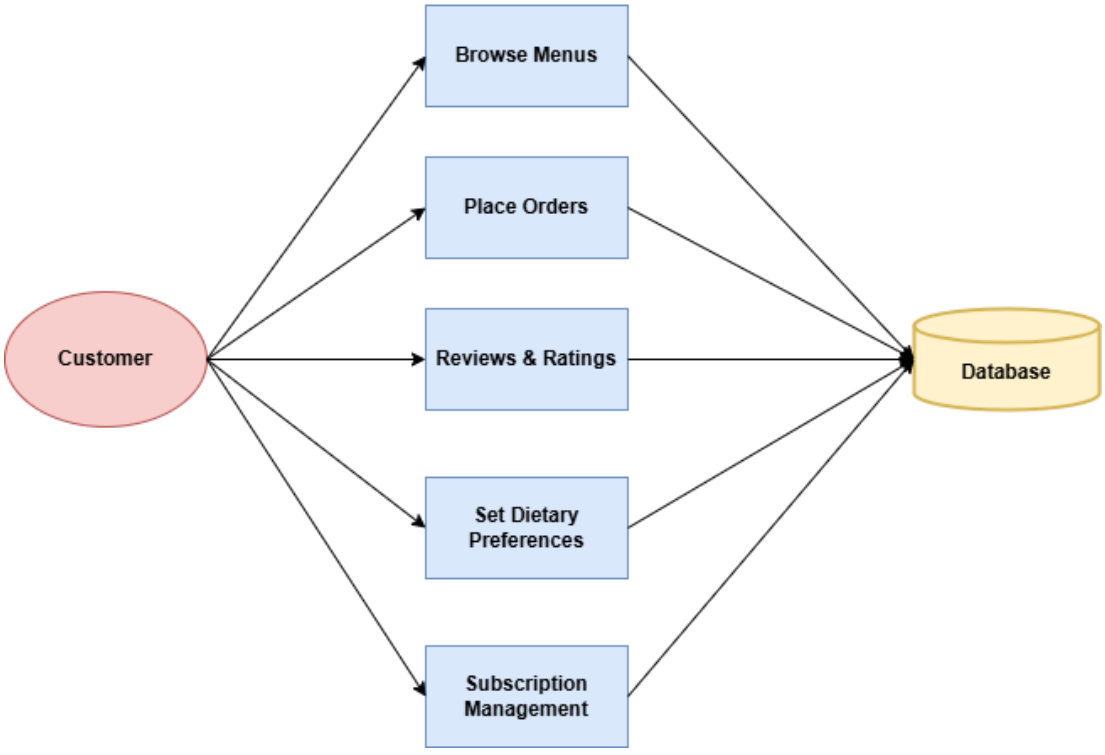

rating
mealID*
chefID*
userID*
review_comment
review_date
rating_id
Ratings & Reviews
gives
receives
phone_no
email
name
password_hash
userID
Users
bio
email
name
chef
chefID
receives
subscribes
receives
places
prepares
donates
customizes
donation_id
Donation
recipient_info
chefID*
mealID*
D_Date
quantity
Chef_Meal_Preferences
chef_meal_pref_id
additional_price
chefID*
mealID*
customization_option
description
ingredients
price
Meals
meal_type
chefID*
mealID
Notifications
notification_id
message
userID*
notification_type
Order_Items
order_item_id
orderID*
subtotal_price
mealID*
quantity
contains
status
subscription_id
Subscriptions
price
userID*
meal_count_per_week
subscription_start_date
is paid
subscription_end_date
orderID
Orders
is paid
status
total_amount
userID*
chefID*
mealID*
quantity
Payments
payment_id
payment_date
orderID*
payment_status
subscription_id*
amount_paid


Admin Dashboard
Admin
Manage User Roles
Payment Oversight
Order Analytics
Manage Users
View Reports
Database

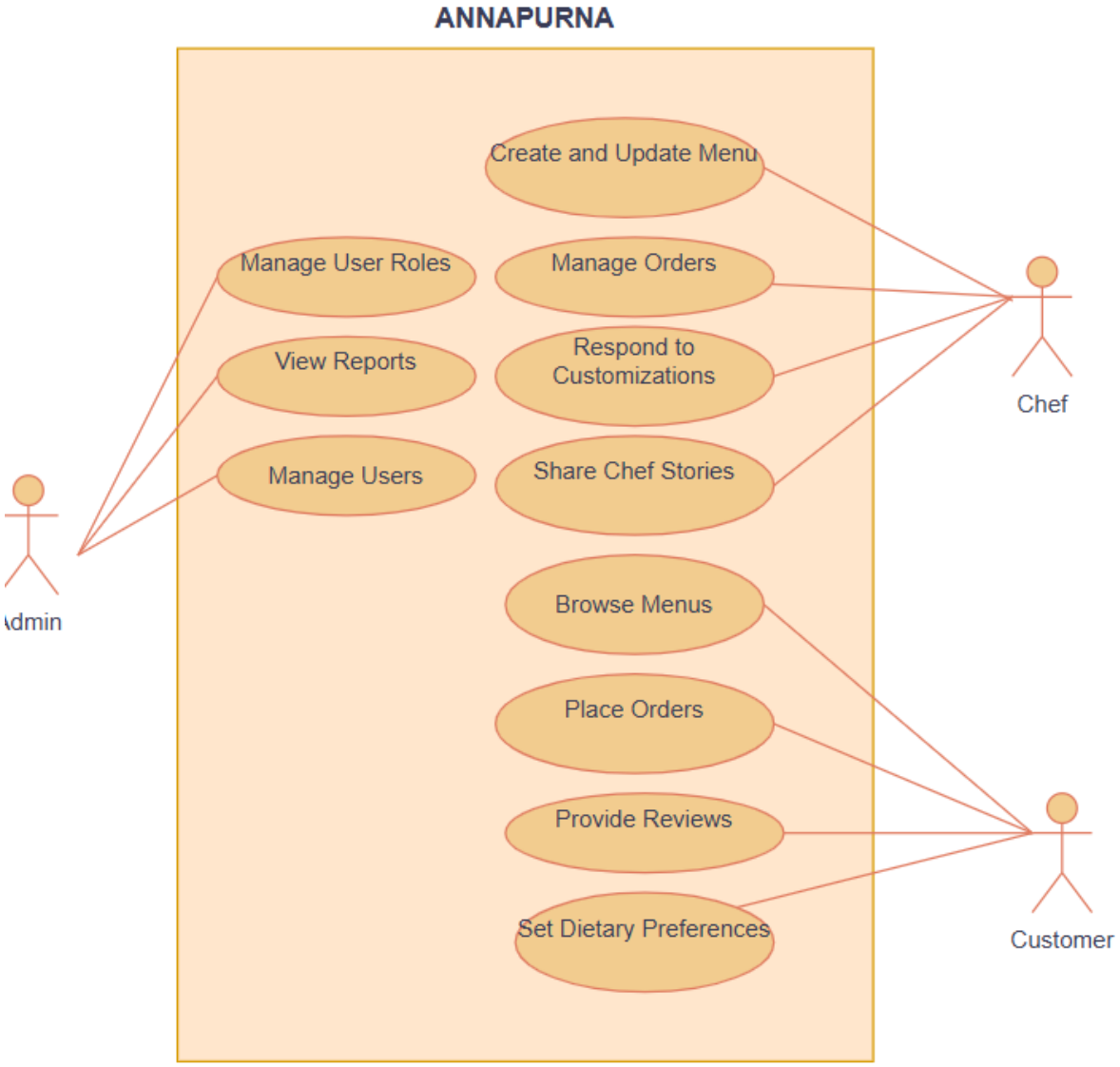
Chef Dashboard



Customer Dashboard



- **Use-Case Diagram**



- **Activity Diagram**