System Requirements

Wasfa!

VERSION: 1.1 REVISION DATE: 08-04-2025

Contents

Section 1 Document Purpose

The purpose of the System Requirements Document is to specify the overall system requirements that will govern the development and implementation of the system. The document will also establish initial security, training, capacity and system architecture requirements, as well as system acceptance criteria agreed upon be the project sponsor and key stakeholders.

Section 2 System Overview

The system is designed to provide users with a seamless experience for discovering, and preparing meals based on available ingredients. By allowing users to input their available ingredients, the system suggests a variety of recipes, offers ratings and reviews, and supports account management features, including subscriptions for unlimited access.

The system will also include an **admin dashboard** for managing dishes, ingredients, users, and subscriptions.

Section 3 General System Requirements

3.1 Major System Capabilities

Specify the major system capabilities in terms of availability, target deployment environment(s), device accessibility, and/or technical capability.
For example:

- *System must be available on the Internet*
- *System must be available 24 hours per day*
- *System must be accessible by mobile devices*
- *System must be able to accept electronic payments*

- The system must be available on the Internet and accessible 24/7 to all users.
- The system must support responsive web design, ensuring compatibility across desktops, tablets, and mobile devices.
- The system must provide an intuitive and user-friendly interface that reflects themes of cooking and culinary culture.
- The system must allow users to search for dishes based on ingredients entered through a dynamic and interactive input interface.
- The system must support account creation, login, and management features for users.
- The system must allow registered users to rate dishes (1–5 stars) and leave comments.
- The system must support electronic payment functionality through PayPal for subscription-based services.
- The system must provide a free trial period upon initial account creation, after which users must subscribe to continue accessing premium features.
- The system must include an admin dashboard for managing ingredients, dishes, subscription plans, and user accounts.

3.2 Major System Conditions

Specify major system assumptions and/or constraints (aka conditions). The conditions may limit the options available to the designer/developer. For example:

- System must use the FDOT Enterprise GIS Framework
- System must use FDOT Enterprise Document Management System
- System must interface with Bank of America credit card payment system

- The system must be designed to work across multiple devices (desktops, tablets, and mobile devices) without compromising user experience.
- The system must integrate with a secure and reliable electronic payment gateway, specifically PayPal, for handling subscription-based payments.
- The system must support multiple user roles, including regular and subscribed users (for browsing, searching, and interacting with dishes) and admin users (for managing ingredients, dishes, subscription plans, and user data).
- The system must adhere to data protection regulations and ensure user privacy, especially for sensitive personal and payment data.
- The system must implement a robust authentication mechanism to securely handle user registration, login, and account management processes.
- The system must ensure compatibility with modern web browsers (Chrome, Firefox, Safari, Edge) and compatibility with mobile web browsers.
- The system must be responsive, adjusting content layout dynamically for different screen sizes, ensuring accessibility on both mobile and desktop platforms.

3.3 System Interfaces

Describe the dependency and relationship requirements of the system to other enterprise/external systems. Include any interface to a future system or one under development. For clarity, a graphical representation of the interfaces should be used when appropriate.

The system will interact with several external systems and services to ensure full functionality, including payment processing, and user authentication. The following outlines the key system interfaces:

1. **Payment Gateway Interface:**
   - The system will interface with PayPal for processing subscription payments. This interface will handle transaction requests, securely process payments, and return transaction status (success/failure) to the system.
   - The integration with PayPal will be through their REST API to facilitate secure and real-time payment processing.
2. **User Authentication and Authorization:**
   - The system will use a third-party authentication service (e.g., OAuth, Firebase Authentication, or a similar service) and/or an internal authentication service (e.g., JWT) for user registration, login, and account management.
   - This interface will securely handle user credentials and ensure proper validation for all account-related operations.

3. **Admin Dashboard Interface:**
   - The system will include a web-based admin dashboard for managing dishes, ingredients, user accounts, and subscriptions.
   - The admin dashboard will be accessible only by authorized admin users and will interface directly with the backend database to manage content and user data.
4. **Future System Interfaces:**
   - The system is designed to accommodate future integrations, such as additional payment gateways, external recipe databases, or machine learning-based recommendation systems for personalized meal suggestions.
   - Interfaces to these future systems will be developed as necessary, with a focus on maintaining scalability and flexibility.
   - **Cloud Hosting Platform:**
     - The system will be hosted on a cloud-based platform (e.g., AWS, Google Cloud, or Azure) to provide scalability, reliability, and availability.
     - Cloud services will host both the web and mobile applications, including databases and storage for user data, dish details, and payment information.

3.4 System User Characteristics

Identify each type of user of the system by function, location, and type of device. Specify the number of users in each group and the nature of their use of the system.

The system is designed to accommodate various types of users, each with distinct functions and usage patterns. The primary user groups are outlined below:

End Users (Registered and Non-Registered Users):

- **Function:** End users are individuals seeking cooking inspiration, meal suggestions, and the ability to browse dishes based on ingredients they have. They can interact with the system by searching for dishes, rating them, and leaving comments.
- **Location:** Users can access the system from anywhere with an internet connection, primarily from their home or mobile locations.
- **Type of Device:** End users will access the system through web browsers on desktops, tablets, and mobile devices.
- **Number of Users:** The expected user base can range from hundreds to thousands of users, depending on marketing and growth. The system should be scalable to accommodate future growth.
- **Nature of Use:** Users can search for recipes by ingredients, browse meal suggestions, rate dishes, and leave comments. Users who are registered and subscribed will have unlimited dish suggestions and more detailed meal recommendations. Unsubscribed will have a 30-days free trial.

Admin Users:

- **Function:** Admin users are responsible for managing and overseeing the content and users within the system. This includes adding, updating, or deleting ingredients and dishes, managing subscriptions, and handling user queries or issues.
- **Location:** Admin users will access the system from any location with internet connectivity, typically from a computer or device with administrative privileges.

- **Type of Device:** Admins will interact with the system using web browsers from desktop or laptop devices. They will also use the admin dashboard for content management.
- **Number of Users:** The number of admin users will be limited to a small group, typically 3–5 users, depending on the system's scale and needs.
- **Nature of Use:** Admin users will use the admin dashboard to manage content (dishes, ingredients), view user activities, and process payments or manage subscriptions. They will have access to all areas of the system, including reports on user behavior and content analytics.

Section 4 Functional Requirements

Specify functional requirement for users of the system.
A functional requirement specifies what a system should do, detailing the services, tasks, or functions the system must perform.
Example: The system must allow users to log in with their email and password.

1. **User Registration and Authentication**
    - The system must allow users to create an account by providing their first name, last name, email address, and password.
    - The system must allow users to log in
        - Using their registered email address and password.
    - The system must provide a "Forgot Password" feature to allow users to reset their password through a registered email address.
2. **Subscription Management:**
    - The system must allow users to subscribe to a premium plan after the free trial period has expired.
    - The system must allow users to cancel or change their subscription plan at any time.
    - The system must allow users to view their subscription status and payment history in their account dashboard.
3. **Dish Search and Ingredient Input:**
    - The system must allow users to search for dishes by entering ingredients they have available.
    - The system must display a list of suggested dishes based on the ingredients entered by the user.
    - The system must allow users to add multiple ingredients for dish search, with real-time suggestions as they type.
    - The system must display dish names, ratings (out of 5 stars), and images in the search results.
4. **Dish Details and Ratings:**
    - The system must allow users to view detailed information about a selected dish, including ingredients, preparation steps, description, and user ratings.
    - The system must allow users to rate dishes on a scale of 1–5 stars and leave written comments.
    - The system must allow users to share dish details on social media platforms.
5. **User Profiles:**
    - The system must allow users to view and edit their personal information, including their name, email address, and password.
    - The system must allow users to view and manage their account settings, including notification preferences.

6. **Admin Dashboard:**
    - The system must allow admin users to manage dish data, including adding, updating, and deleting dishes.
    - The system must allow admin users to manage ingredient data, including adding, updating, and deleting ingredients.
    - The system must allow admin users to view and manage user accounts, including subscription status, user feedback, and payment history.
7. **Payment Processing:**
    - The system must integrate with PayPal to process subscription payments securely.
    - The system must allow users to make payments through PayPal for subscription purchases.
    - The system must provide users with a confirmation of successful payments and update their subscription status accordingly.
8. **Multi-Device Support:**
    - The system must be responsive and accessible on a variety of devices, including desktop computers, tablets, and mobile phones.
    - The system must provide a seamless experience across the website, ensuring users can access their accounts and features from any device.
9. **Notifications and Alerts:**
    - The system must send notifications to users regarding subscription expiry, new dish suggestions, and updates to account information.
    - The system must allow users to configure notification preferences, including email and push notifications.
10. **Admin Content Moderation:**
    - The system must allow admin users to review and remove user-generated content, such as comments and ratings, to ensure compliance with content policies.
11. **Admin Audit System:**
    - The system must audit and log every admin operation.
    - Admin operations include managing recipes, ingredients, users and subscriptions.

Section 5 Non-Functional Requirements

Specify non-functional requirements for the system.
Non-Functional Requirement: A non-functional requirement specifies how a system should perform, describing the quality attributes, system behaviors, or constraints the system must meet. Example: The system should load the user dashboard within 3 seconds.

1. **Performance:**
    - The system must load all pages (including the homepage, dish search, and user profiles) within **3 seconds** for a smooth user experience.
    - The system must handle up to **10,000 concurrent users** without performance degradation.
    - Search results for dishes based on ingredients must be returned within **2-4 seconds**.
2. **Availability:**
    - The system must be available **99.9% of the time**, excluding scheduled maintenance windows.

3. **Scalability:**
   - The system must be designed to handle increasing numbers of users and data without a significant drop in performance. This includes the ability to scale horizontally or vertically on the cloud infrastructure as a future system interface.
4. **Security:**
   - The system must implement and provide secure authentication for admin users and regular users through **hashed passwords** and **OAuth**.
5. **Usability:**
   - The system's interface should be intuitive and user-friendly, with no more than **three clicks** required for users to complete a basic task (e.g., searching for dishes or rating a recipe).

6. **Maintainability:**
   - The system should be modular and well-documented, allowing for easy updates and maintenance by developers.
   - The system must include logging mechanisms that capture errors, performance issues, and user interactions for troubleshooting and optimization.
   - System components (e.g., the admin dashboard, user authentication, payment gateway) must be loosely coupled, enabling independent updates and changes.
7. **Mobile Responsiveness:**
   - The system's mobile application and website must be fully responsive, providing an optimized user experience across all screen sizes, from small smartphones to large tablets.

Section 6 Initial System Architecture

Specify the data platform, hardware, software, programming languages, tools and operating system requirements for the application or project.

a. Identify any specialized hardware requirements that must be purchased or upgraded prior to development, or in support of the implementation, of the application or project.
b. Identify any specialized software requirements that must be purchased or upgraded prior to development, or in support of the implementation, of the application or project.
c. Identify any programming languages and tools selected for the development of the application or project.
d. Identify any network/operating system or combination of network/operating systems that will be used for the development of the application of project.

6.1 Data Platform

The system will use a **relational database management system (RDBMS)**, such as **MySQL** or **PostgreSQL**, to store and manage user data, dish information, ingredients, and subscription details.

6.2 Software Requirements

- **Web Framework:**
  - The system will use **Spring Boot** framework for the backend API.
  - The frontend will be built with **React.js** to provide a dynamic, interactive user experience across the web and mobile platforms.
- **Payment Integration:**
  - The system will integrate with the **PayPal API** for payment processing, requiring the **PayPal SDK** to be integrated into both web and mobile applications.
- **Authentication and Authorization:**
  - The system will use **OAuth 2.0** with **JWT** for user authentication and secure login processes.
- **Admin Dashboard:**
  - The admin panel will be built using **React.js** for the frontend and will communicate with the backend through **RESTful APIs**.

6.3 Programming Languages and Tools

- **Backend Development:**
  - **Spring Boot** framework will be used for the backend API.
  - **Java 8+** for structured and scalable code.
- **Frontend Development:**
  - **React.js** for building the user interface of the web application.
  - **HTML5, CSS3** for web page structure and styling.
- **Database Management:**
  - **MySQL** or **PostgreSQL** will be used for database management and interactions.
  - **Hibernate ORM** will be used for simplifying database queries in Java.
- **Version Control:**
  - **Git** will be used for version control, with **GitHub** as the repository platform for code collaboration.
- **Development Environment:**
  - The development environment will use **Visual Studio Code** or **JetBrains WebStorm** for efficient coding.
  - **Docker** will be used for containerization and easier deployment across different environments (local, staging, production).

6.4 Network/Operating System Requirements

- **Development OS:**
  - Developers can use either **Linux (Ubuntu/Arch Linux)** or **Windows 10** for their development environment.
- **Production OS:**
  - The production environment will run on **Linux-based cloud servers** (e.g., **Ubuntu** or **Amazon Linux**) for the backend.
  - The frontend will communicate with the backend through REST APIs over HTTPS, ensuring secure and optimized data transmission.

Section 7 Glossary

Define all terms and acronyms required to properly interpret the requirements contained within this document.

API (Application Programming Interface):
A set of rules and protocols that allow different software applications to communicate with each other, enabling the system to interact with external services (e.g., payment gateways, third-party services).

Authentication:
The process of verifying the identity of a user to ensure that only authorized individuals can access certain features of the system.

Authorization:
The process of granting or denying specific privileges to authenticated users, determining what actions or resources a user is allowed to access.

Backend:
The part of the application that handles data processing, business logic, and database interactions. It communicates with the frontend to manage the overall functionality of the system.

Frontend:
The user interface and experience part of the system that users interact with directly. It includes elements like web pages, mobile app interfaces, and interactive components.

OAuth 2.0:
An authorization framework that allows third-party applications to securely access user data without sharing user credentials. Commonly used for user login and authorization.

PayPal API:
An application programming interface provided by PayPal that allows systems to integrate payment processing capabilities, enabling users to pay for subscriptions or make purchases.

RESTful API (Representational State Transfer):
A type of API that adheres to the principles of REST architecture, using standard HTTP methods (GET, POST, PUT, DELETE) for communication between the frontend and backend.

Responsive Design:
An approach to web design that ensures a website or application adjusts to different screen sizes and resolutions, providing optimal experience on desktops, tablets, and mobile devices.

Hibernate ORM (Object-Relational Mapping):
An ORM library for Java that helps developers interact with relational databases like MySQL or PostgreSQL using Java, abstracting SQL queries into Java objects and methods.

UI/UX (User Interface/User Experience):
UI refers to the design and layout of a system's interface, while UX encompasses the overall experience a user has with the system, focusing on usability, accessibility, and user satisfaction.