**Software Requirements Specification**

**for**

**Random Meal Generator**

Prepared by

Dharshini K (23CSR046)  
Ashwinth D K(23CSR026)  
 Bharathvaj N G(23CSL253)

Kongu Engineering College  
  
27/02/2025

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Intended Audience 1

1.3 Project Scope 1

1.4 Definitions,Acronyms and Abbreviations 1

1.5 References 1

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Features 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

3. System Features 3

3.1 User Authentication 3

3.2 Ingredient Management 3

3.3 Meal Suggestion System 3

3.4 Advanced Search and Filtering 3

4. External Interface Requirements 4

4.1 User Interfaces 4

4.2 Hardware Interfaces 4

4.3 Software Interfaces 5

4.4 Communications Interfaces 5

5. Other Nonfunctional Requirements 5

5.1 Performance Requirements 5

5.2 Security Requirements 5

5.3 Usability Requirements 5

5.4 Scalability Requirements 5

6. Other Requirements 5

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Introduction**

**1.1 Purpose**

The purpose of this Software Requirements Specification (SRS) is to define the requirements for the Random Meal Generator application. This application will suggest meals based on user preferences and available ingredients. The platform will include features such as user authentication, ingredient management, meal suggestions, recipe details, meal history tracking, and customizable meal plans.

**1.2 Intended Audience**

The intended audience for this SRS includes frontend and backend developers working on the system's implementation, UX/UI designers responsible for the interface, and project stakeholders such as business owners, professors, or investors reviewing the project.

**1.3 Project Scope**

The application will serve as a meal suggestion tool that allows users to receive random meal recommendations based on dietary preferences and available ingredients. Users can manage ingredient lists, filter meal options, save meal plans, and access recipes. The system will ensure personalized recommendations and an intuitive user experience.

**1.4 Definitions, Acronyms, and Abbreviations**

* **User**: Any person using the application.
* **Meal Suggestion**: A meal randomly selected based on user preferences and available ingredients.
* **API**: Application Programming Interface.
* **Frontend**: The user interface built using React.js.
* **Backend**: The server logic built using Node.js and Express.
* **Database**: MongoDB for storing user data and meal information.
* **JWT**: JSON Web Token, used for authentication and security.

**1.5 References**

* MongoDB Documentation: https://www.mongodb.com/docs/
* React Documentation: https://reactjs.org/docs/
* Node.js Documentation: https://nodejs.org/en/docs/

**2. Overall Description**

**2.1 Product Perspective**

The system is a web-based and mobile-friendly application that allows users to input available ingredients and receive meal suggestions. It will consist of a frontend built using React, a backend using Node.js, and MongoDB as the database. The application may integrate with external APIs for recipe suggestions and nutritional analysis.

**2.2 Product Features**

1. **User Authentication** - Users can sign up, log in, and manage their profiles.
2. **Ingredient Management** - Users can add, remove, or update ingredients in their virtual pantry.
3. **Meal Suggestions** - The system will suggest meals based on available ingredients and preferences.
4. **Recipe Details** - Each meal suggestion will include a recipe, ingredients, and preparation steps.
5. **Meal History** - Users can view a history of past meal suggestions.
6. **Filters & Preferences** - Users can set dietary restrictions, preferences, and ingredient exclusions.
7. **Custom Meal Planning** - Users can create and save meal plans for the week.
8. **Nutritional Information** - The application will display the nutritional values of meals.
9. **Advanced Search and Filtering** - Users can search for meals using advanced filters, such as cooking time, ingredient count, difficulty level, and meal type (breakfast, lunch, dinner, snack).

**2.3 User Classes and Characteristics**

* **Regular Users**: Users who input ingredients, receive meal suggestions, and view recipes.
* **Admin**: Users responsible for managing the meal database, moderating content, and ensuring high-quality recommendations.

**2.4 Operating Environment**

* **Frontend**: Built using React.js.
* **Backend**: Built using Node.js and Express.
* **Database**: MongoDB.
* **Browser Compatibility**: Chrome, Firefox, Edge, and mobile browsers.
* **Mobile App Compatibility**: Android and iOS.

**2.5 Design and Implementation Constraints**

The system must ensure a secure and smooth user experience. It should handle concurrent users efficiently, protect user data, and integrate external recipe APIs if necessary. It must be scalable and allow for future expansion.

**3. System Features**

**3.1 User Authentication**

1. **Description**: Users can register, log in, and manage their profiles.
2. **Input**: Email, password.
3. **Output**: Authentication success/failure message.
4. **Preconditions**: Valid email and password.
5. **Postconditions**: User session is created.
6. **User Interaction**: Users fill out a form and submit login or registration details.

**3.2 Ingredient Management**

1. **Description**: Users can add, update, or delete ingredients from their pantry.
2. **Input**: Ingredient name, quantity.
3. **Output**: Updated pantry list.
4. **Preconditions**: User must be logged in.
5. **Postconditions**: The ingredient list is updated.
6. **User Interaction**: Users enter ingredients manually or select from a predefined list.

**3.3 Meal Suggestion System**

1. **Description**: The system randomly suggests a meal based on available ingredients.
2. **Input**: User's ingredient list, dietary preferences.
3. **Output**: A suggested meal with recipe details.
4. **Preconditions**: User has added ingredients.
5. **Postconditions**: A meal is displayed.
6. **User Interaction**: Users click "Generate Meal" to receive a suggestion.

**3.4 Advanced Search and Filtering**

1. **Description**: Users can search for meals based on specific criteria.
2. **Input**: Filters such as cooking time, ingredient count, meal type, difficulty level.
3. **Output**: List of filtered meals.
4. **Preconditions**: User must enter search criteria.
5. **Postconditions**: Filtered results are displayed.
6. **User Interaction**: Users input criteria and receive a list of meals matching their preferences.

**4. External Interface Requirements**

**4.1 User Interfaces**

The application should have an intuitive and responsive UI across web and mobile platforms:

* **Login & Registration Pages**: Allow users to sign up or log in.
* **Dashboard**: Displays the ingredient list, meal suggestions, and options for meal planning.
* **Meal Suggestion Page**: Shows randomly generated meal ideas.
* **Recipe Detail Page**: Displays cooking instructions and nutritional information.
* **Meal History & Favorites Page**: Lists past suggestions and saved meals.
* **Settings Page**: Allows users to manage preferences and account details.

**4.2 Hardware Interfaces**

The system will be compatible with the following hardware:

* **Web Browsers**: Chrome, Firefox, Safari, Edge.
* **Mobile Devices**: Android and iOS smartphones and tablets.
* **Barcode Scanner (Optional)**: If implemented, users can scan ingredients for quick entry.

**4.3 Software Interfaces**

The system will integrate with:

* **Recipe API** for fetching recipes.
* **Nutritional API** (e.g., USDA Nutrition Database) for calorie and nutrient breakdown.
* **Authentication API** (e.g., Firebase, OAuth) for secure user login.
* **Database**: MongoDB for storing user data and meal information.

**4.4 Communication Interfaces**

* **Frontend to Backend**: Uses RESTful APIs to send and receive data.
* **Backend to Database**: Uses MongoDB for CRUD (Create, Read, Update, Delete) operations.
* **External API Integration**: Connects to recipe and nutritional services for meal suggestions.

**5. Nonfunctional Requirements**

**5.1 Performance Requirements**

* The platform must support at least 500 concurrent users without slowing down or crashing, ensuring a smooth user experience during peak times.

**5.2 Security Requirements**

* User data should be encrypted to protect privacy.
* Secure authentication mechanisms, such as JWT tokens, should be implemented.
* Payments (if applicable) must be securely processed using external gateways like Stripe.

**5.3 Usability Requirements**

* The platform should be easy to use and accessible across all modern web browsers and mobile devices.
* The UI should be intuitive, with a seamless experience for all users.

**5.4 Scalability Requirements**

* The platform should be able to accommodate more users as demand grows without performance issues.
* The backend should support horizontal scaling.

**6. Other Requirements**

* The platform must be compatible with modern browsers and mobile devices.
* Secure session management is required to prevent unauthorized access.
* All critical actions must be logged for auditing purposes.