**Software Requirements Specification (SRS) for RationEase**

Table of Contents

1. Introduction

1.1 Purpose

1.2 Scope

2. Overall Description

2.1 Product Perspective

2.2 User Classes and Characteristics

2.3 Operating Environment

2.4 Design and Implementation Constraints

3. Functional Requirements

3.1 User Registration and Authentication

3.2 Ration Booking

3.3 Ration Collection

3.4 Check Ration Status

3.5 Update Family Members

4. Non-Functional Requirements

4.1 Performance

4.2 Security

4.3 Usability

4.4 Reliability

4.5 Scalability

**1. Introduction**

**1.1** **Purpose**

The purpose of this Software Requirements Specification (SRS) document is to provide a detailed and comprehensive overview of the Ration Management System. This document outlines the functionalities, features, and requirements of the system, serving as a reference for stakeholders, developers, and users.

**1.2 Scope**

The Ration Management System is aimed at transforming the conventional ration distribution process prevalent in small Indian towns. This system addresses the challenges faced by citizens in accessing government-allocated rations by providing an online platform for booking collection slots and managing the entire distribution process. The system facilitates efficient communication between users and vendors while enhancing transparency and convenience.

**2. Overall Description**

**2.1 Product Perspective**

The Ration Management System acts as a web-based intermediary platform connecting users and vendors. It manages user authentication, ration booking, collection tracking, and user information updates. The system interfaces with a relational database to store user details, ration information, and collection status.

**2.2 User Classes and Characteristics**

- User: Individuals eligible for government-allocated rations. Users register, book collection slots, update family details, and mark ration collection.

- Vendor: Individuals responsible for ration distribution. Vendors access collection data, manage slots, and view reports.

**2.3 Operating Environment**

The system operates as a web application accessible through modern browsers. Users and vendors require a stable internet connection to interact with the system.

**2.4 Design and Implementation Constraints**

- The frontend must be developed using React.js, ensuring a responsive and user-friendly interface.

- The backend will be implemented using Spring Boot, handling user interactions and database operations.

- The system will use a relational database (e.g., MySQL) for data storage, ensuring efficient data retrieval and management.

**3. Functional Requirements**

**3.1 User Registration and Authentication**

- Users must register by providing personal details, including name, contact information, and address.

- Usernames and passwords must be securely stored and managed.

- Users should be able to log in using their credentials.

**3.2 Ration Booking**

- Users must be able to view available time slots for ration collection.

- Users can book a suitable time slot for collection.

- Confirmation emails or notifications should be sent to users upon successful booking.

**3.3 Ration Collection**

- Users must mark their ration collection as complete after visiting the distribution center.

- Vendors should have access to a dashboard displaying user collection status.

**3.4 Check Ration Status**

- Users should be able to view the status of their ration collection for the current month.

- The system should display whether the ration has been collected or not.

**3.5 Update Family Members**

- Users can add new family members by providing their details.

- Existing family member information can be updated or deleted.

- Users should have the ability to manage their family details efficiently.

**4. Non-Functional Requirements**

**4.1 Performance**

- The system must handle multiple concurrent user interactions without experiencing significant performance degradation.

- Response times for user interactions, including booking and status checks, should not exceed 5 seconds.

**4.2 Security**

- User passwords must be securely hashed and stored in the database.

- Access to sensitive user information, such as contact details, should be restricted based on user roles.

**4.3 Usability**

- The user interface should be intuitive, requiring minimal training for users to navigate and perform actions.

- User interactions and actions should be guided by clear and concise instructions.

**4.4 Reliability**

- The system should be available 24/7, with scheduled maintenance windows communicated in advance.

- Data integrity must be maintained, and backup mechanisms should be in place to prevent data loss.

**4.5 Scalability**

- The system should be designed to accommodate a growing number of users and vendors without compromising performance.

- Scalability considerations should include load balancing and efficient database management.

**5. Glossary**

- User: An individual eligible for government-allocated rations.

- Vendor: An individual responsible for the distribution of rations.