Software Requirements Specification for *FruitsSpark*

Introduction

Purpose

The purpose of this document is to define the requirements for the Fruit Consumption Tracker web application. This application aims to help users monitor their daily fruit intake, track nutrient consumption, and provide insights into their eating habits.

Scope

The Fruit Consumption Tracker is a web-based application designed to be user-friendly and engaging. The primary users are individuals who want to maintain a healthy diet by tracking their fruit consumption. The application will allow users to log their daily fruit intake, monitor nutrient levels, receive reminders, and access insights based on their eating patterns.

Definitions, Acronyms, and Abbreviations

User: An individual using the Fruit Consumption Tracker to log their fruit intake.
Nutrients: Essential dietary components such as vitamins, minerals, and fiber.
UI: User Interface.
DB: Database

Overview

This document details the functional and non-functional requirements of the Fruit Consumption Tracker web application. It also provides use case scenarios and describes the system's external interfaces.

.Overall Description

. Product Perspective

The Fruit Consumption Tracker is a standalone application designed to be used on any modern web browser. It will interact with a central database to store and retrieve user data.

. Product Features

User Registration and Login: Users can create an account and log in to access personalized data.

Daily Logging: Users can log the fruits they consume daily, including quantities and types.
Nutrient Tracking: The application will calculate and display the nutritional content of the logged fruits.

- Reminders: Users will receive notifications to log their fruit intake.

- Insights: The application will generate insights based on the user's eating habits, such as nutrient deficiencies or trends.

- User Interface: A responsive, easy-to-navigate interface designed to engage users.

. User Classes and Characteristics

- General Users: Individuals interested in tracking their fruit consumption. These users may not have technical expertise.

- Admin Users: Individuals responsible for managing the application's content and user data.

. Operating Environment

- Client-Side: Any modern web browser (e.g., Chrome, Firefox, Safari) on desktops, tablets, or smartphones.
- Server-Side: The application will be hosted on a cloud server using technologies like Node.js and MongoDB.

. Design and Implementation Constraints

- Responsive Design: The application will be responsive and function well on various screen sizes.
- Data Security: User data will be securely stored and protected from unauthorized access.
- Performance: The application should load within seconds on standard internet connections.

. Assumptions and Dependencies

- Users have access to the internet.
- The application will be compatible with modern web browsers.

- Nutritional data for fruits will be sourced from a reliable database.

Specific Requirements

. Functional Requirements

User Registration and Login
- Users will be able to register with their email and password.
- Users will be able to log in with their credentials.
- The system will validate user credentials against the stored data.

Daily Logging

- Users will be able to select fruits from a predefined list.
- Users will be able to input the quantity of each fruit consumed.
- The system will store the user's daily log in the database.

Nutrient Tracking
- The application will calculate nutrient intake based on the user's daily log.
- The system will display nutrient information in an easy-to-read format.

Reminders
- Users will be able to set reminders for logging their fruit intake.
- The system will send reminders via email or push notifications.

Insights
- The application will analyze the user's data to provide insights on nutrient consumption trends.
- Insights will be displayed on the dashboard and updated in real-time.

Non-Functional Requirements

Performance

- The application should handle up to 0,000 concurrent users without performance degradation.
- The application will have a maximum response time of seconds for any user action.

Usability

- The application should have a consistent and intuitive UI across all platforms.
- The application should be accessible to users with disabilities, following WCAG . guidelines.

Security

- User data will be encrypted both in transit and at rest.
- The system will implement strong password policies and two-factor authentication (FA).

Reliability

- The application should have 99.9% uptime, with planned maintenance scheduled during off-peak hours.
- The system should automatically back up data every hours.

External Interface Requirements

. User Interfaces

- Login Page: A simple interface for users to enter their credentials.

- Dashboard: Displays a summary of the user's fruit consumption and insights.

- Logging Interface: Allows users to input their daily fruit intake.

- Settings: Provides options for users to manage their account and reminders.

Software Interfaces

- Database: The application will interface with a MongoDB database to store and retrieve user data.
- API: The application will use RESTful APIs to communicate between the client and server.

Communication Interfaces

- The system will use HTTPS for secure communication between the client and server.

- Sending email notifications through an SMTP server.

. Other Non-Functional Requirements

Performance Requirements

- The application will load within seconds on a standard G connection.

Safety Requirements

- The system will include error-handling mechanisms to prevent data loss during unexpected shutdowns.
- The system should log any failed operations for later review.

Security Requirements

- Implementing role-based access control (RBAC) to restrict access to sensitive data.
- Use encryption for all data storage and transmission.

Software Quality Attributes

- Maintainability: The codebase will be modular and well-documented to facilitate future updates.

- Scalability: The application will be designed to scale horizontally to accommodate growing user numbers.