**Software Requirements Specification (SRS)**

**AI Diet & Nutrition Planner**

**1. Introduction**

**1.1 Purpose**

The AI Diet & Nutrition Planner aims to provide personalized meal recommendations based on user dietary preferences, nutritional goals, and health requirements. The system leverages machine learning models to estimate caloric and macronutrient content for various food items.

**1.2 Scope**

This project consists of a backend API, a web application, and a mobile app. It allows users to:

* Get nutritional predictions for food items.
* Generate customized meal plans.
* Authenticate using sign-up/login features.
* Interact with an AI chatbot for dietary guidance.

**1.3 Definitions, Acronyms, and Abbreviations**

* **API**: Application Programming Interface
* **ML**: Machine Learning
* **FastAPI**: A Python framework for building APIs
* **Flask**: A micro web framework for Python
* **React**: A JavaScript library for frontend development
* **Flutter**: A UI toolkit for mobile applications

**2. System Overview**

The system consists of:

* **Frontend (Web & Mobile App)**: Developed using React and Flutter.
* **Backend (API Services)**: Developed using FastAPI and Flask.
* **Machine Learning Model**: Uses a trained Gradient Boosting model for food nutrient prediction.

**3. Functional Requirements**

* **User Authentication**: Users must be able to sign up, log in, and manage their accounts.
* **Food Prediction Service**: Users can input a food name and receive an estimated nutritional breakdown.
* **Meal Planning**: The system will generate meal plans based on user preferences.
* **AI Chatbot**: Users can interact with a chatbot to get diet recommendations.

**4. Non-Functional Requirements**

* **Performance**: The system should return predictions within 2 seconds.
* **Scalability**: It should support multiple concurrent users.
* **Security**: User data should be encrypted and securely stored.
* **Availability**: The system should have at least 99.5% uptime.

**5. System Architecture**

* **Frontend**: React (Web), Flutter (Mobile)
* **Backend**: FastAPI (Primary API), Flask (Nutrient Prediction API)
* **Database**: PostgreSQL (User data), CSV (Food nutrition data)
* **ML Model**: Gradient Boosting with MultiOutput Regression

**6. Data Flow and Processing**

1. User inputs a food item via web or mobile app.
2. Request is sent to the Flask API.
3. The model predicts nutrients based on stored data.
4. The response is returned and displayed to the user.

**7. Constraints and Assumptions**

* The system assumes a stable internet connection.
* The food dataset should be periodically updated.
* The ML model should be retrained for better accuracy.