	ADA University School of Information Technologies and Engineering **CSCI 3509: Intro to Software Engineering** Fall 2023

Homework 2: **Software Requirements Specification**
Due by 23 November 2023

<Project Name>

Project team: **<Team Name>**
Instructor: Kamila Ismayilova

Submitted in partial fulfillment of the requirements of the CSCI 3509: Intro to Software Engineering course project

Version date	Version information
<Date>	Initial draft
<Date>	**<Version description>**

Team member	Contribution to the homework
<Student Name 1>	**<Description of the work contributed>**
<Student Name 2>	
<Student Name 3>	
<Student Name 4>	

Table of Contents

<Automatically generate here>

1. Introduction

This is a Software Requirements Specification for project **<Project Name>** proposed for partial fulfillment of the requirements of the *Introduction to Software Engineering* course in the School of Information Technologies and Engineering at ADA University, Baku, Azerbaijan.

<Write here a very brief description of your project. It is intended as an introduction and should be informal and concise. You can copy it from the Project Proposal document and revise it (if needed).>

The project consists of a web app for generating personalized, evidence-based diet plans. Users enter their basic profile (age, sex, height, weight), their goals (loss/gain/ maintenance), activity level and diet preferences. The calorie and macro targets are calculated, relevant nutrition facts and meal ideas are retrieved from a curated knowledge base (KB) using retrieval augmented generation (RAG), and all are promoted to an LLM for producing a meal plan. The plan is provided in structured JSON format so that UI can render and display it back to the user clearly.

Scope and Objectives

<Please copy the Project Proposal document's respective paragraph. You may want to revise the Project Proposal document scope and objectives here (if needed).>

The scope of EatWise is to provide an accessible, web-based **personal diet planning assistant** that leverages modern AI technologies while staying within realistic resource and budget constraints. From the project proposal the main objectives are:

- Implement a **Minimum Viable Product (MVP)** that can generate diet plans within approximately **1.5 months**.
- Maintain LLM usage costs below **100 AZN** for the project.
- Provide a technically sound demonstration of:
  - Backend development with Java + Spring Boot.
  - Frontend with React.
  - RAG pipeline with Python + FastAPI, Chroma, and OpenAI APIs.
- Support a **plan history** feature so logged-in users can revisit previously generated plans.

Definitions

<Insert here any technical word for which the meaning may not be known. Do not assume that the readers have specialized knowledge in the application.>

Term	Definition
RAG	**Retrieval-Augmented Generation**
KB	**Knowledge Base**
LLM	**Large Language Model**
API	**Application Programming Interface**

2. Overall Description

<DELETE THIS EXPLANATION: This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in section 3, and makes them easier to understand. In a sense, this section tells the requirements in plain English for the consumption of the customer. Section 3 will contain a specification written for the developers.>

Project Features

This is the *main* content of section 2. This describes the functionality of the project in the language of the customer. What specifically does the software have to do? Drawings are good but remember this is a description of what the software does, NOT how you are going to build it.>

1. **User registration and Login:**
    - **Users create an account with e-mail/username and password**
    - **Users log in to access personalized features and plan history**

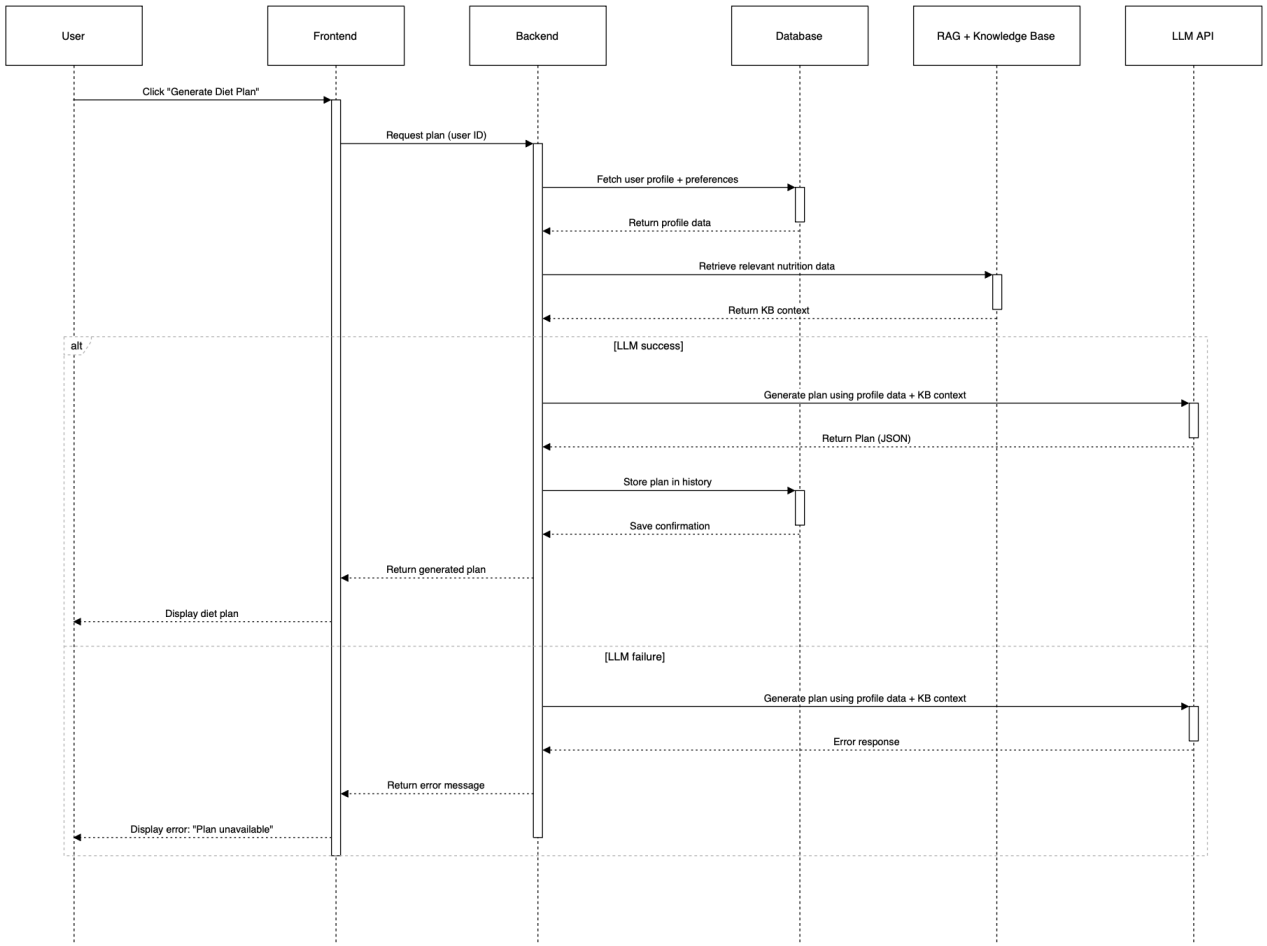
2. **Profile Set-up**
    - **User enters profile data: age, sex, height, weight**
    - **Users choose their activity level**

3. **AI-Powered Meal Planning (RAG + LLM)**
    - The system retrieves candidate meals from the knowledge base using semantic search (via Chroma) and filters by the user's diet and exclusions.
    - The system asks the LLM to generate a multi-day meal plan (3–30 days) that:
        - Uses the candidate meals.
        - Respects daily calorie/macro targets.
        - Accommodates the user's dietary preferences and excluded ingredients.
    - The LLM returns a structured plan in machine-readable (JSON) form.

4. **Plan Visualization**
    - The frontend displays each day's meals grouped by type (breakfast, lunch, dinner, snacks).
    - For each day, the user can see total calories and macros.
    - A short summary explains how the plan supports the user's chosen goal.

5. **Plan History and Retrieval:**
    - Each plan generated by a logged-in user is saved and associated with their account.
    - Users can access their plan history
    - Users can open any saved plan and view it in the same format as a newly generated one

Sequence Diagram (Main Feature)



Target User Characteristics

<Describe those general characteristics of the intended users of the software including educational level, experience, demographics (maybe) and technical expertise.

What is it about your potential user base that will impact the sofware design? Their experience and comfort with technology will drive UI/UX. Other characteristics might influence sofware design.>

The intended primary users are general adult population, who:

- Being interested in improving diet quality or achieving a specific weight-related goal.
- May or may not have prior experience tracking calories/macros.
- Having basic computer and smartphone literacy and are comfortable using web apps.

Technical skills:

- Being able to use browsers and web forms.
- Not required to understand LLMs, RAG, or nutritional science in detail.

Constraints

<Provide a general description of any other items that will limit the developer's options. This section captures non-functional requirements in the customer's language. A more formal presentation of these will occur in section 3.>

Time constraint:

- **MVP must be implemented in roughly 1.5 months**

Non-functional requirements:

- Plans must be generated within acceptable time

Data constraints:

- The KB should be clean and well curated
- No paid proprietary datasets

Assumptions and Dependencies

<This section is *catch-all* for everything else that might influence the software design and that did not fit in any of the categories above. You can copy it from the Project Proposal document and revise it.>

Assumptions:

- User Data Accuracy: We assume that the users will provide correct personal information, dietary preferences, restrictions and health goals for the LLM to create a relevant diet plan. The relevance of the plan directly depends on the accuracy of the user input.
- LLM Capability and Safety: We assume that the third-party LLM is capable of generating safe and helpful diet plans based on the knowledge base we will be using. We assume that the LLM's output will not be treated as medical advice.
- API Availability and Cost: We assume that the third-party LLM API will have high uptime and its costs will not exceed our budget.

Dependencies

- Availability and reliability of:
- LLM APIs.
- Chroma and its storage backend.
- PostgreSQL database.

Network access between:

- Backend and AI microservice.
- AI microservice and OpenAI servers.

3. Specific Requirements

3.1 Functional Requirements

3.1.1 User Registration & Authentication

FR-1. The system shall allow a user to register by providing an email, password, and required personal data fields.
FR-2. The system shall securely store user credentials using industry-standard hashing.
FR-3. The system shall allow a registered user to log in using valid credentials.
FR-4. The system shall reject login attempts when credentials are invalid.
FR-5. The system shall allow users to log out from any active session.

3.1.2 Diet Plan Generation (LLM + RAG)

FR-6. The system shall retrieve relevant nutrition data from the Knowledge Base (KB) using RAG.
FR-7. The system shall provide the user-specific targets and KB data to the LLM to generate a structured diet plan.
FR-8. The LLM output shall be returned strictly in structured JSON format.
FR-9. The system shall display the generated meal plan on the frontend.
FR-10. The system should allow users to request long-form or short-form plan formats.

3.1.3 Diet Plan History

FR-11. The system shall store each generated plan in a user-specific history.
FR-12**.** The system shall allow users to view previously generated diet plans.
FR-13. The system shall allow users to delete entries from their history.

3.1.4 Frontend User Interface

FR-12. The UI shall present all user input fields in a structured and accessible form.
FR-13. The UI shall display diet plans in a clearly readable layout.
FR-14**.** The UI shall notify users when required fields are missing or invalid.

3.2 Non-Functional Requirements

3.2.1 Usability

NFR-1. The UI shall be intuitive so that users can complete all actions without external instructions.
NFR-2. All required fields shall display real-time validation messages.

3.2.2 Performance

NFR-3. The system shall respond to API requests within 500 ms under typical load, excluding external LLM latency

Note: Requests that require LLM response are not taken into account in NFR-3 as it depends on external system and RAG layer on top of it.

3.2.3 Security

NFR-4. All communication between frontend, backend, and microservices shall use HTTPS.
NFR-5. Passwords shall never be stored in plaintext.
NFR-6. API keys for the LLM and vector store shall be stored in environment variables.

4. Analysis Models (Use Cases)

<DELETE THIS EXPLANATION: In the following sub-sections list all analysis models and related information use for deriving requirements.>

External Actor Descriptions

<Describe here your external actors. You may distinguish human actors and computer actors.>
User (human actor): The individual who is looking to improve their diet or achieve their weight goals. They interact with the frontend interface to provide personal data, define goals and request diet plans.

Third-party LLM API (computer actor): External LLM service which receives structured prompt containing user data and KB context. Processes this information to generate diet plans in JSON format and returns it to the system.

Use Case Descriptions

<DELETE THIS EXPLANATION: Provide a numbered list of user stories that are the features of the software to be implemented. Each user story is an operation that the user can perform in the software. For each user story, provide a detailed description so you know what to build and so you can write a test case to demonstrate that your software provides that feature. Your system should have at least 7 stories or features. See example below:>

No	User story name	Description
1	User Registration	Users sign up to create an account by providing their email, password and full name. This enables the possibility of storing the generated diet plans per user account.
2	Profile Setup	Users form their profile by providing personal information, dietary preferences, and health goals. By doing so, the accuracy and suitability of the generated diet plans for the user is ensured.
3	User Login	Users log in to utilize the web app's services by entering the email and password they had used during the registration.
4	User Logout	Users log out to terminate their active session.
5	Manage User Profile	Users update their profile by changing personal information to reflect their physical changes. Ensuring that future plans remain accurate as the user progresses.

6	**Generate Diet Plan**	**Users request a diet plan by specifying their goal and dietary preferences. The systems uses nutritional data and AI to generate a personalized diet plan.**
7	**View Diet Plan History**	**Users access their history of generated plans to review past recommendations.**
8	**Delete Diet Plan**	**Users remove any unwanted diet plan from their history.**

Use Case Diagram

<Draw the UML use case context diagram for the game. Have the use cases shown in your diagram correspond to the user stories described in the previous section and vice versa. Make sure to connect all appropriate actors to corresponding use cases.>

Use Case 1

<Use the following template for each of the use cases. Make sure to address alternate scenarios as these often go unnoticed and become problematic. Use the template and example provided as a guide.

Use Case Number:	**UC-XX (Replace XX with a number)**
Use Case Name:	**Enter the name of Use Case**
Actor(s):	**List all actors that participate in this Use Case**
Description:	**Describe the purpose of the Use Case and give a 1-2 line description. This could be the same as the description provided for the user story.**
Trigger:	**Enter the trigger and its type i.e. External or Temporal**
Pre-condition(s):	**Enter the condition that must be true when the main flow is started.**
Scenario Flow:	**Main (success) Flow: Steps should be numbered. Indicate information for each step.**
Alternate Flows:	**Alternate Flows: Include the post condition for each alternate flow if different from the main flow. Indicate information for each step. Indicate information for each step.**
Post Condition:	**Enter the condition that must be true when the main flow is completed.**

Use Case Number:	UC-01
Use Case Name:	User Registration
Actor(s):	User (new user wanting to create an account)
Description:	User should provide personal information, dietary preferences, and health goals to the system.
Trigger:	External - User opening registration page to create an account
Pre-condition(s):	- System has been setup and configured. - System is running and accessible via web browser - User has accessed the website with URL
Scenario Flow:	Main (success) Flow: 1. User selects option register/sign up 2. System displays registration form 3. System requests login credentials (email, password) 4. User provides email and password 5. User confirms password 6. System validates email format and password strength - If invalid, system displays error message. Return to step 4. 7. System checks if email already exists in database 8. System verifies if password confirmation matches 9. System creates user account 10. System stores account information in database 11. System displays confirmation message of successful registration 12. System automatically logs in the user 13. System redirects to profile setup page (UC-02)

Alternate Flows:	Alternate Flows: Alternate Flow #1: After Step 4, user wants to cancel registration 1. User presses "Go Back" to cancel registration 2. System requests confirmation 3. User confirms 4. System returns to main page Alternate Flow #2: After step 7, email already exists in system 1. System detects existing email 2. System displays message "Email already registered" 3. System offers option to login or recover password 4. Redirect to login page or return to step 4 Alternative Flow #3: After step 8, password confirmation does not match 1. System detects that passwords do not match 2. System displays error message "Passwords do not match" 3. System clears password fields 4. Return to step 4
Post Condition:	User account is created. User is logged in and ready to complete profile setup.

Use Case Number:	UC-02
Use Case Name:	Profile Setup
Actor(s):	Registered User
Description:	User should provide personal information, dietary preferences, and health goals to the system to be able to generate a diet plan
Trigger:	External - User has finished account registration or user wants to generate a diet plan
Pre-condition(s):	- User has completed account registration (UC-01) - User is logged in - User's diet-related profile is not completed
Scenario Flow:	Main (success) Flow: 1. System opens personal information form 2. System then requests personal information (age, weight, height etc.) 3. User enters personal information 4. System requests activity level information 5. User selects activity level 6. System requests dietary preferences 7. User provides dietary preferences (f.e. vegan, vegetarian etc.) 8. System requests health goals 9. User provides health goals (loose/gain weight, muscle gain etc.) 14. User confirms profile information 15. System verifies if all required information is provided and valid - If information is not provided, system displays error message. Return to step 2. 16. System saves profile information to database 17. System marks profile as complete 18. System displays success message "Profile setup complete" 19. System redirects user to dashboard
Alternate Flows:	Alternate Flows: Alternate Flow #1: After step 15, system detects invalid or incomplete input 1. System detects invalid or incomplete input 2. System highlights problematic fields with red 3. System provides a guide for valid input range 4. Return to step 2 Alternative Flow #2: User wants to skip profile setup 1. User selects "Complete" before completing the profile setup 2. System displays warning "Profile setup is required to generate a diet plan"

	3. System asks confirmation to skip 4. If user confirms to skip - System saves any entered information so far - System redirects to dashboard with limited features - System disables “Generate diet plan” function - System displays banner message which tells to complete profile setup 5. If user cancels, return to current step Alternative Flow #3: User closes browser/tab during profile setup 1. System detects interrupted session 2. System automatically saves all entered information to temporary storage 3. When user returns and logs in UC-03 - System detects incomplete profile setup - System displays a message “Would you like to complete your profile setup?” - If user selects yes, system loads saved data and resumes from last step. - If user selects no, redirect to Alternative Flow #2
Post Condition:	User profile is complete with all personal information entered. System now can generate diet plans. Profile data is stored in database.

Use Case Number:	UC-03
Use Case Name:	User Login
Actor(s):	Registered User
Description:	User should provide email and password they had used during the registration phase (UC-01)
Trigger:	External – Registered User logging into their account
Pre-condition(s):	- System is running and accessible via web browser - User has accessed the website with URL - User has completed account registration (UC-01) - User is logged out
Scenario Flow:	Main (success) Flow: 1. User selects the option to login 2. System displays login form 3. System requests login credentials (email, password) 4. User provides email and password 5. System verifies whether the provided email and password matches 6. System displays the main page
Alternate Flows:	Alternate Flows: Alternative Flow #1: After step 5, either email or password is incorrect 1. System detects invalid email or password 2. System displays error message “Invalid email or password” 3. Return to step 3
Post Condition:	User is logged in and sees the main page

Use Case Number:	UC-04
Use Case Name:	User Logout
Actor(s):	Registered User
Description:	User terminates their active session
Trigger:	External – User logging out of their account
Pre-condition(s):	- System is running and accessible via web browser - User has accessed the website with URL

	- User is logged in
Scenario Flow:	Main (success) Flow: 1. User selects the option to logout 2. System ends the user's active session 3. System redirects the user to the login page
Alternate Flows:	Alternate Flows:
Post Condition:	User is logged out and sees the login page

Use Case Number:	UC-05
Use Case Name:	Generate personalized diet plan
Actor(s):	Registered User, LLM (large language model), RAG System
Description:	System should generate personalized, evidence based diet plan based on user's data using RAG and LLM processing.
Trigger:	External – User requests to generate a diet plan
Pre-condition(s):	- User is logged in (UC-03) - User has completed profile setup (UC-02) - LLM API is available and accessible - RAG system is operational
Scenario Flow:	Main (success) Flow: 1. User navigates to diet plan generation page 2. User selects option to generate plan 3. System calculates calorie and macro target based on user data 4. RAG system takes relevant nutritional facts and meal ideas from knowledge base 5. System creates prompt with user data and retrieved information 6. System sends request to LLM API (OpenAI GPT 4o mini) 7. LLM processes information and generates a plan in JSON format 8. System recieves and validates the plan 9. System saves the plan in the database 10. System displays meal plan to user
Alternate Flows:	Alternate Flows: Alternate Flow #1: After step 6, LLM API call fails or returns invalid response 1. System detects error (API call fail, invalid JSON, or incomplete data) 2. System logs error info 3. System displays message "Unable to generate plan. Please try again later" Alternative Flow #2: After step 6, RAG system fails to get the information 1. System detects the error 2. System logs error info 3. System displays message "Unable to generate plan. Please try again later"
Post Condition:	User gets personalized diet plan. Plan is saved in database and added to user's history

Use Case 2

...

Use Case 5

...

5. References

<Insert here any document referred to in the document. An example might be articles or Web sites that you consulted during the literature search.>