SMART NUTRITION AND HEALTH ADVISOR

by
Esra KAYA

Engineering Project Report

Yeditepe University
Faculty of Engineering
Department of Computer Engineering
2025

SMART NUTRITION AND HEALTH ADVISOR

APPROVED BY:

Prof. Dr. Şebnem Baydere (Supervisor)	. . . . .
Assist. Prof. Dr. Esin Onbaşıoğlu	. . . . .
Instructor Betül Arslan	. . . . .

DATE OF APPROVAL: .../.../2025

ACKNOWLEDGEMENTS

First of all I would like to thank my advisor Prof. Dr. Şebnem Baydere for her guidance and support throughout my project.

Also I would like to thank my parents for their support and encouragement throughout my education up to the present.

ABSTRACT

SMART NUTRITION AND HEALTH ADVISOR

The increase in weight gain and unhealthy eating habits have become a major public health issue in modern society. This project offers an intelligent nutrition consulting system built to address the challenges by providing nutritional advice and health monitoring capabilities. The envisioned AI-powered platform integrates various features such as diet recommendations, exercise plans, recipe suggestions, calorie and nutritional information of foods with reliable information from USDA API, calorie intake tracking, water intake tracking and body mass index measurement. The system is built on an advanced FastAPI-based backend system and a complex database schema to manage user accounts, nutrition data and interaction history. The AI agent uses natural language processing to provide context-specific responses to user queries related to nutrition, sports and health.Behind it all lies a powerful artificial intelligence agent built on the Langchain framework and utilizing the Retrieval Augmented Generation (RAG) concept. Custom-collected nutrition and health documents are embedded using OpenAI's text-embedding-3-small model and stored in a FAISS vector database so that the retriever can fetch the most related answers to each user's query. Then, these retrieved documents are processed and responded to by OpenAI's GPT-4o large language model so that the system can give accurate and context-aware answers on nutrition, fitness, and health. Primary features include live macro and micronutrient tracking, personalized daily calorie and water intake goal calculations based on user-inputted attributes (age, gender, weight, height, activity level, and health goals) and an in-depth session-based chat function for continuous user interaction. The mobile app implements secure user login, full nutrition tracking with automatic macro calculation and smart goal setting algorithms that adapt to the user's profile. The water tracking system provides personalized hydration goals with the ability to track glass by glass and the nutrition side allows for in-depth food intake tracking with retrospective history. It also provides users with a space to create their shopping lists. This smart nutrition guide is a comprehensive solution for users who want to improve their eating habits and well-being by using AI-based recommendations, simplifying the process of healthy lifestyle choices for users with different demographics and health goals.

ÖZET

AKILLI BESLENME VE SAĞLIK DANIŞMANI

Kilo alımındaki artış ve sağlıksız beslenme alışkanlıkları toplumda önemli bir halk sağlığı sorunu haline gelmiştir. Bu proje, karşılaşılan zorlukları ele almak için oluşturulmuş akıllı bir beslenme danışmanlık sistemi sunmaktadır. Öngörülen yapay zeka destekli platform, diyet önerileri, egzersiz planları, tarif önerileri, yiyeceklerin kalori ve besin bilgileri, USDA API'sinden gelen güvenilir bilgiler, kalori alımı takibi, su alımı takibi ve vücut kitle indeksi ölçümü gibi çeşitli özellikleri bir araya getirir. Sistem, gelişmiş bir FastAPI tabanlı arka uç sistemi ve kullanıcı hesaplarını, beslenme verilerini ve etkileşim geçmişini yönetmek için karmaşık bir veritabanı şeması üzerine kurulmuştur. Yapay zeka ajanı, beslenme, spor ve sağlıkla ilgili kullanıcı sorgularına bağlama özgü yanıtlar sağlamak için doğal dil işlemeyi kullanır. Tüm bunların arkasında, Langchain çerçevesi üzerine kurulmuş ve Retrieval Augmented Generation (RAG) konseptini kullanan güçlü bir yapay zeka ajanı yer almaktadır. Özel olarak toplanan beslenme ve sağlık belgeleri, OpenAI'nin text-embedding-3-small modeli kullanılarak gömülür ve bir FAISS vektör veritabanında saklanır, böylece alıcı her kullanıcının sorgusuna en alakalı yanıtları alabilir. Daha sonra, bu alınan belgeler işlenir ve OpenAI'nin GPT-4o büyük dil modeli tarafından yanıtlanır, böylece sistem beslenme, spor ve sağlık konusunda doğru ve bağlama uygun yanıtlar verir. Uygulamanın birincil özellikleri arasında canlı makro ve mikro besin takibi, kullanıcı tarafından girilen niteliklere (yaş, cinsiyet, kilo, boy, aktivite seviyesi ve sağlık hedefleri) dayalı kişiselleştirilmiş günlük kalori ve su alımı hedefi hesaplamaları ve sürekli kullanıcı etkileşimi için derinlemesine oturum tabanlı sohbet işlevi bulunur. Mobil uygulama güvenli kullanıcı girişi, otomatik makro hesaplamalı takibi ve kullanıcının profiline uyum sağlayan akıllı hedef belirleme algoritmaları uygular. Su izleme sistemi, kişiselleştirilmiş hidrasyon hedefleri sağlar ve beslenme tarafı, geriye dönük geçmişle derinlemesine yiyecek alımı takibine olanak tanır. Ayrıca kullanıcılara alışveriş listelerini oluşturmaları için bir alan sağlar. Bu akıllı beslenme rehberi, AI tabanlı öneriler kullanarak yeme alışkanlıklarını ve refahlarını iyileştirmek isteyen kullanıcılar için kapsamlı bir çözümdür ve farklı demografik özelliklere ve sağlık hedeflerine sahip kullanıcılar için sağlıklı yaşam tarzı seçimleri sürecini basitleştirir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In society, people have sought various solutions to manage their dietary habits and health goals. The search for optimum healthy nutrition has become increasingly important. According to the Global Nutrition Report 2023, inadequate and unhealthy nutrition in all its forms, global hunger and declining food security continue to be among the world's greatest health challenges affecting almost every country, population group and age group. The rise of digital technologies and artificial intelligence offers new opportunities to address these challenges through innovative, easily accessible and customized dietary guidance systems.

1.1. Intelligent System Trends in Nutrition Technology Research

It is important to analyze the current trends in health applications to understand the development of nutrition technology. Recent studies show a significant increase in the adoption and use of AI-powered healthcare applications.The synthesis of chatbot technology into healthcare has gained traction in the recent days. AI systems are deemed effective in providing suggestions to the end-users and have garnered a generally positive attitude. Nutrition mobile applications are increasingly popular and offer features ranging from the simplest calorie counters to elaborate meal planners and nutrient analyzers. However, most of the existing solutions target only one concept. Quite often, these applications are way far from what a chatbot is supposed to offer. They do not possess the intelligence and personalization capabilities that enable them to provide real dietary guidance to their users. This unmet need in the market has brought to light an AI-based nutrition assistant that, upon understanding user preferences and queries, offers contextual suggestions in response to those queries.

1.2. Nutrition Data Analysis and Personalization

While most nutrition apps usually focus on basic tracking capabilities, fewer consider intelligent data analysis and personalized recommendations. When comprehensive nutrition databases like USDA FoodData Central are integrated with state-of-the-art AI technologies, nutrition analysis can be made even more accurate and precise. Our proposed system thus aims at filling this gap in combining large-scale nutrition data with intelligent conversational capability and personalized health tracking. This is an interpretation process of data and getting actionable insights rather than a matter of data collection. Users desire systems that track nutritional intake and intelligent feedback from the diet whereby patterns of nutrition or general health are identified and suggest meals to the diet's recommendations in accordance

with individual health goals and preferences.

1.3. Motivation

While planning my graduation project, I focused on nutrition and diet, which are among the biggest health problems today and thought about what kind of innovations I could add. In my observations, I realized that many people around me had a lack of knowledge about healthy nutrition, could not get the right diet advice, had difficulty in following their nutrition in today's applications, and had a lack of motivation. When I examined the existing applications, I saw that most of them were limited to providing static information. Of course, it is very important to be able to adapt to the rapidly developing technologies and attract the attention of users. If we look at the impact of today's large language models in society, we see that the majority attracts their attention, uses them, and gets ideas on every subject. As a result of these observations, I thought that an artificial intelligence-supported application that users can benefit from could be helpful and could be adapted to today. I found the use of LLM-Agent appropriate in terms of which artificial intelligence technologies can be used so that users can reach the right and fast answer on nutrition and diet issues. In this way, I would be able to benefit from the capabilities of the pretrained language model and convey the most accurate information to the user thanks to the tools of the AI Agent that I designed specifically for the project data, and if I present all of these in a mobile application, this useful project will reach the users very quickly. In this case, the motivation to develop a smarter, more communicative and user-friendly solution using modern artificial intelligence technologies arose.

In addition to all these, in the project that I will design around the User-Agent-LLM communication, I will integrate features that calculate the user's daily water and calorie needs based on the information they will enter in their profile and that will allow them to create a shopping list. These integrations are not independent of the project purpose, but aim to improve the user's experience.

1.3.1. Accessibility and User Engagement

One of the primary motivations is to solve low user engagement and lack of motivation in health and nutrition apps. Current market research shows that eighty percent of health and fitness apps are abandoned within six months of download. The main reasons for this are that users constantly receive information through static information, do not receive timely feedback, and lose motivation and get bored after a while. Considering these challenges, my

main goal is to develop an application that provides users with an AI-powered conversational interface to enable users to interact with nutrition and health and maintain their motivation. This project will allow natural language interaction where users can ask various questions about diet, nutrition, recipes and sports interactively instead of static information and get the most appropriate answers. In addition, the fact that they can get answers to their questions within the app, i.e. easily accessible, will be a positive feature for users. This approach encourages continuity and consistency in the application by turning it into an engaging conversation rather than a confusion or difficulty in nutrition. In this way, it was aimed to prevent applications from being abandoned in a short time. This application will be created to adapt to each user's communication style, preferences and desires and to provide both reliable information and appropriate responses. In this way, a real project that users will love and benefit from will be implemented.

1.3.2. Comprehensive Health Tracking

In the mobile application, apart from the artificial intelligence question-answer system, daily macro and micronutrient targets will be calculated based on the information to be kept in the database according to the age, height, weight, activity level and goal of each user, daily water intake will be calculated according to their weight and progress visualization will be provided, making it easier for users to follow the goals they need to reach. If users reach their goals, a motivating approach will be provided by the application or a warning will be given by the application if they exceed the required amount. Thus, it is aimed for users to easily adapt to the application and monitor their own development.

1.4. Terms

The terms used in this project are :

- **Framework:** It is the basic structure of underlying software that provides a foundation for developing applications.
- **Artificial Intelligence :** Artificial Intelligence (AI) is the technology and science of making machines think, learn and make decisions like humans.
- **Natural Language Processing:** Natural Language Processing (NLP) is a branch of AI that focuses on the interaction between computers and humans through natural language, enabling machines to understand, interpret and generate human language.

- **Large Language Models:** LLM (Large Language Model) is a type of artificial intelligence model trained on vast amounts of text data to understand and generate human-like language.
- **AI Agent:** An AI agent is a system that uses an AI model to reason, plan, and take actions through tools to achieve a specific user-defined goal.
- **RAG:** RAG (Retrieval-Augmented Generation) is an AI technique that combines information retrieval with text generation, allowing language models to access external knowledge bases to provide more accurate and up-to-date responses.
- **LangChain:** LangChain is a framework for developing applications powered by language models, providing tools for chaining together different AI components and managing complex AI workflows.
- **Vector Database:** Vector Database is a specialized database designed to store, index, and query high-dimensional vector data, optimized for similarity search and machine learning applications.
- **Embedding:** Embedding is a technique in machine learning where high-dimensional data is mapped to lower-dimensional vector spaces while preserving semantic relationships.
- **OpenAI LLM and Embedding:** OpenAI's LLM generates human-like text, while its embeddings represent text as vectors for tasks like semantic search and similarity.
- **FAISS:** FAISS is a library for efficient similarity search and clustering of dense vectors, optimized to handle large-scale vector databases and semantic search operations.
- **FastAPI:** FastAPI is a modern, high-performance web framework for building APIs with Python based on standard Python type hints, designed for rapid development and automatic interactive documentation generation.
- **PostgreSQL:** PostgreSQL is an advanced open-source relational database management system (RDBMS) that supports both SQL and JSON querying.
- **Chatbot:** Chatbot is a computer program designed to simulate conversation with human users through text interactions, utilizing AI and NLP technologies to understand and respond to user queries.
- **Flutter:** Flutter is Google's open-source UI software development kit (SDK) for creating natively compiled applications for mobile, web and desktop from a single codebase using the Dart programming language.

- **Dart:** Dart is a programming language developed by Google that is optimized for building user interfaces. It is primarily used with the Flutter framework for cross-platform app development.
- **REST API:** REST API (Representational State Transfer) is an architectural style for designing networked applications that uses standard HTTP methods (GET, POST, PUT, DELETE) for communication between client and server.
- **FoodData Central API :** The FoodData Central API allows developers to access detailed nutrient and food data from the USDA's integrated database, supporting efficient search and retrieval of information about foods available in the United States.
- **Edamam API :** The Edamam Meal Planner API offers the ability to generate customized meal plans tailored to individual dietary preferences, restrictions, and nutritional goals.

1.5. Scope and Limitations

It is very important to determine the scope and limitations in the project. Scope indicates which topics a project covers and which goals are intended to be achieved, while limitations indicate restrictions that may affect the results of the project. I have divided this section into 2 parts. I will discuss scope and limitations separately.

1.5.1. Scope

The scopes of the project are :

- Development of a comprehensive AI Agent system using the LangChain framework, integrating multiple custom tools including a BMI calculator, macro calculator, calorie calculator, water intake calculator, and USDA food database search capability.
- Implementation of RAG (Retrieval-Augmented Generation) technology to allow the AI agent to access and retrieve information using the GPT-4o model and FAISS vector database.
- Development of a cross-platform mobile application using the Flutter framework that provides nutrition tracking and AI-powered chatbot service that works on both iOS and Android platforms.

- Implementation of a backend system using FastAPI with PostgreSQL database integration for user management, nutrition tracking and conversation history storage.
- A comprehensive nutrition tracking system that provides personalized daily calorie calculations, macro and micronutrient tracking, water consumption tracking, and visualization capabilities to see user progress.
- Multi-session chat management system that allows users to open new chats,access past chat data and delete chats.
- Integration with external APIs including USDA FoodData Central API and Edamam Meal Planner API.

1.5.2. Limitations

The limitations of the project are :

- Due to budget and API cost constraints, our project relies on external paid services such as OpenAI API.This situation may limit extensive testing and long-term deployment without sufficient funding allocation.
- The chatbot system depends on OpenAI's API availability to provide answers to users. There are also external services such as FoodData Central API and Edamam API. Therefore, any service interruption or change in these external dependencies may affect system functionality and user experience.
- Due to the nature of AI language models, our chatbot may not always provide very accurate answers. Therefore, the system should not be considered a replacement for professional medical or nutritional advice.
- In order to provide personalized nutrition recommendations and track user progress, our system requires the collection of personal health information such as weight, height, age, dietary preferences, and eating habits. This may raise privacy concerns for some users.

1.6. Problem Definition

The problem we are trying to address is the lack of a unified smart solution where users can access comprehensive nutrition guidance, personalized diet recommendations, real-time

health monitoring and conversational AI technology. Current nutrition apps on the market have features that prevent them from providing effective, engaging, and sustainable health management solutions for users. Current solutions fail to meet the basic needs of modern health-conscious users who seek intelligent, personalized, and culturally relevant nutrition guidance that adapts to their individual preferences and goals. In addition, most current apps lack the integration of advanced AI technologies such as RAG systems that can provide evidence-based answers from reliable nutrition databases and research sources.

1.7. Requirements

First, we need to clearly define our requirements to develop a mobile application that meets our goals. Before defining our requirements, it is essential to define the types of user in our application to better understand the requirements. Our system primarily serves health-conscious individuals looking for smart nutrition guidance and tracking capabilities. I will define the requirements in 4 subcategories. These are functional, non-functional, software and hardware requirements. Each requirement will be defined, explained and assigned a priority level (low, medium or high) according to their importance to the application.

1.7.1. Users of Mobile Application

Our app users basically consist of one main group: Health Conscious Users. These are people who want to maintain or improve their health with better nutrition tracking, personalized diet guidance, and smart AI-powered recommendations. This user type can be thought as our app's core user base.This demographic pursues goals such as weight management, physical performance enhancement, and overall lifestyle improvement.

1.7.2. Functional Requirements

Functional requirements of the project are:

- **User Registration and Profile Management**
  - **Requirements:** Users shall be able to register and create profiles in the application.
  - **Description:** During registration ,users should provide the following informations: name, email, password, age, gender, height, weight, activity level, and health goals. All of this information is required for the personalized tracking

system. This information should be securely stored in the database.

  - **Priority:** High

- **User Authentication**

  - **Requirements:** Users shall be able to securely authenticate themselves to access the application.
  - **Description:**Users should be able to log in to the application via using their email and password . The authentication system must implement secure password hashing and session management. Users should also be able to change their passwords in the application.
  - **Priority:** High

- **AI-Powered Conversational Interface**

  - **Requirements:** Users shall be able to interact with chatbot for nutrition guidance and advice.
  - **Description:** The system should provide a conversational AI interface powered by LLM model via the LangChain framework. Users should be able to ask questions in natural Turkish or English about nutrition, diet, recipes, health goals and receive contextual answers to their questions. The chatbot should use tools and RAG technology to access nutritional information from local databases.
  - **Priority:** High

- **Multi-Session Chat Management**

  - **Requirements:** Users shall be able to create,access and manage multiple conversation sessions with the AI assistant.
  - **Description:** Users should be able to create separate chat sessions at any time. The system should maintain conversation history for each session, allow users to view previous conversations and deletion capabilities.
  - **Priority:** High

- **Nutrition History and Analytics**

- **Requirements:** Users shall be able to review their historical data and track their progress over time.
- **Description:** During registration ,users should provide the following informations: name, email, password, age, gender, height, weight, activity level, and health goals. All of this information is required for the personalized tracking system. This information should be securely stored in the database.
- **Priority:** Low

- **Calorie and Macro Goals**
  - **Requirements:** Users shall be able to see their daily calorie and macro needs.
  - **Description:** The system should be able to provide daily calorie and macro needs based on the users profile informations like age, gender,activity level and their goal.In addition , users should be able to enter their foods macros and the system should calculate macros.
  - **Priority:** High

- **Calorie Goal Alert System**
  - **Requirements:** Users shall receive pop-up alert if they exceed daily calorie intake.
  - **Description:**Users should see their progress in the application and if they exceed their daily calorie intake , system should give warnings. Through this warning , users should be directed to the chatbot to talk.
  - **Priority:** Medium

- **Water Intake Tracking and Notification**
  - **Requirements:** Users shall be able to see their daily amount of water based on their weight.
  - **Description:** In the system , users should see their water goal and add glass of water to track the progress . Additionally if they want to drink more water than daily water intake , users should be able to change their goal.If they reach and exceed their daily water goal , the system should display motivation message to motivate users.

- **Priority:** High

- **Shopping List Monitoring**
  - **Requirements:** Users shall be able to create their shopping list.
  - **Description:** In the system , users should be able to create their shopping list based on their needs .Also , users should add new items to the list and delete the items.
  - **Priority:** Medium

- **Profile Information Management**
  - **Requirements:** Users shall be able to modify their personal information.
  - **Description:** During registration , users should enter their personal informations .This informations should be modifiable by the user at any time and updated in the database to provide accurate tracking system.
  - **Priority:** Medium

- **User Progress Visualization**
  - **Requirements:** Users shall be able to see their daily progress .
  - **Description:** The system should be able to provide dashboard visualization for users to see their progress. This progress includes water intake ,calorie intake , shopping list completion and additional motivational messages.
  - **Priority:** Medium

1.7.3. Non-Functional Requirements

Non-Functional requirements of the project are :

- **Application Performance and Response Time**
  - **Requirements:** The application shall provide responsive performance within 10ms for basic operations and under 3 seconds for complex AI interactions.

  - **Description:** The mobile application responds to user interactions efficiently with basic endpoints achieving sub-10ms response times. The chatbot system completes AI operations within reasonable timeframes, while the backend maintains stable performance handling hundreds of concurrent requests.
  - **Priority:** High

- **Application Security and Data Protection**
  - **Requirements:** All personal information and health data shall be stored securely with over 95% system reliability for authentication processes.
  - **Description:** The application should implement secure password hashing and robust database storage. Authentication systems demonstrate high reliability rates, while sensitive health information remains protected through comprehensive security measures preventing unauthorized access.
  - **Priority:** High

- **System Reliability and Availability**
  - **Requirements:** The application shall maintain high system reliability with over 85% operational success under stress conditions and robust error handling.
  - **Description:** The system should handle network connectivity issues, API errors and unexpected failures without crashing. Component testing should demonstrate strong reliability across core modules, while performance under load should show consistent operational stability with intelligent error recovery mechanisms ensuring stable user experience.
  - **Priority:** High

- **User Interface Design and Usability**
  - **Requirements:** The application shall provide user-friendly interface.
  - **Description:** The user interface should follow modern design principles with easy navigation, consistent style and accessibility features. The interface should be simple, easy and understandable, and the application should have a help button where users can get help.
  - **Priority:** Medium

- **Cross Platform Compatibility**
  - **Requirements:** The application shall provide consistent functionality across iOS and Android platforms.
  - **Description:** The mobile application should maintain the same level of user experience on both iOS and Android devices, with a platform-appropriate user interface and the same application features.
  - **Priority:** Medium

- **Code Maintainability and Modularity**
  - **Requirements:** The application code shall be written in a maintainable and modular structure.
  - **Description:** The code of application should maintain code organization and modularity. It should be built in a modular structure to facilitate future updates and feature additions.
  - **Priority:** Medium

1.7.4. Software Requirements

Software requirements of the project are :

- **Artificial Intelligence Integration**
  - **Requirements:** The app shall use AI technologies for intelligent chat and recommendation capabilities.
  - **Description:** The system should use a framework for AI agent and tool development, be able to generate answers using a large language model, use a vector database for semantic search, and provide RAG-based knowledge retrieval from this vector base.
  - **Priority:** High

- **Database Management System**
  - **Requirements:** The application shall use a suitable relational database management system to store user data.

- **Description:** The system should use a reliable relational database for efficient database operations. The database should support ACID transactions and provide data integrity for user information, nutrition logs, chat history, and application settings.
- **Priority:** High

- **Cross Platform Mobile Development**
  - **Requirements:** The application shall be developed using a cross platform framework.
  - **Description:** The frontend should be developed using a framework that will enable both iOS and Android platforms from a single codebase.
  - **Priority:** High

- **Backend Development**
  - **Requirements:** The backend shall provide a modern, scalable API architecture for client-server communication.
  - **Description:**The backend of the application should be developed using a web framework. It should provide support for RESTful API endpoints, data validation mechanisms, and cross-origin resource sharing for secure client-server communication. The architecture should support concurrent requests and provide efficient response processing
  - **Priority:** High

- **External API Integration**
  - **Requirements:** The app shall integrate with external services for better functionality.
  - **Description:** The system should integrate with external databases for reliable food information while interacting with the chatbot .It should maintain secure API management. The integration should address API rate limiting and provide fallback mechanisms.
  - **Priority:** Medium

1.7.5. Hardware Requirements

Hardware requirements of the project are :

- **Network Connectivity**
  - **Requirements:** The application shall require stable internet connectivity.
  - **Description:** The system requires reliable internet connectivity for AI chatbot functionality, external API calls, and real-time data synchronization.
  - **Priority:** High

- **Backend Server**
  - **Requirements:** The backend should run on a system that can support many users at the same time and handle AI tasks.
  - **Description:** The server infrastructure should offer enough CPU and memory resources to run the backend application, manage database operations, and support AI features with the ability to scale when needed.
  - **Priority:** High

- **Development and Testing Infrastructure**
  - **Requirements:** Proper devices shall be used for developing and testing of application.
  - **Description:** The development process requires mobile devices suitable for testing across platforms, local development environments for back-end service development and testing, and sufficient computational resources for AI integration testing and performance evaluation.
  - **Priority:** Medium

2. BACKGROUND

Awareness about nutrition in society has increased the need and demand for smart health applications with the advancement of technologies. In this section, we cover articles, projects and applications related to artificial intelligence-supported nutrition guidance, chatbots and nutrition applications.

2.1. Previous works

Since health, nutrition and tracking applications are a wide area, there are various previous works in this area. I have grouped previous works under 3 headings according to the technologies I plan to use in the application. The first part of these headings will be about previous works on AI Chatbot. The second part will be about the use of the RAG system in the health field and finally the third part will be about static nutrition tracking application areas.

2.1.1. AI-Powered Nutrition Chatbots and Conversational Health Systems

Balloccu and Reiter [1] are investigating an approach to presenting nutritional information via natural language generation (NLG) chatbots, compared to traditional diet apps. The authors developed an AI conversational system that answers nutritional questions using natural language text. Their methodology is to create a chatbot that can present nutritional information and compare it with standard diet tracking apps. The study shows that chatbots actually improve user understanding and are more usable. This study is an important study on the impact of AI technology for presenting nutritional information. It also directly supports the approach adopted in the project. However, since the study focuses on information tasks rather than coaching and personalized guidance, a more comprehensive and personalized chatbot system is needed.

Yang et al. [2] aimed to provide a personalized nutrition chatbot that combines causal inference with large language models. Therefore, ChatDiet, a nutrition chatbot framework, was developed. It is in line with the goal of our project. It provides explainable and personalized recommendations. However, its evaluation is limited to case studies. This limitation also indicates the need for larger user tests.

As another example, Prasetyo et al. [3] focused on developing a mobile chatbot to provide timely dietary guidance. For this reason, they introduced FoodBot. They aimed to increase the number of users by reducing manual food recording and increasing conversational interactions with users. However, its long-term effect has not been tested.

2.1.2. Retrieval-Augmented Generation Systems in Health and Nutrition

Recent studies demonstrate the power of the RAG system in the field of nutrition and health. Araki et al. [4] achieved 99.25% accuracy using the RAG-powered LLama 2 model to extract malnutrition information from electronic health records. They also found that this reduced language model hallucinations.

Another example is a study done in Procedia Computer Science [5]. This study managed to answer questions in the field of nutrigenetics by using external knowledge bases with RAG instead of fine-tuning.

Another example, Hou and Zhang [6] introduced iDISK2.0-RAG, a system that combines Retrieval-Augmented Generation with biomedical knowledge base to answer questions about dietary supplements. Their approach achieved over 95 percent accuracy and effectively reduced hallucinations, demonstrating the value of integrating large language models with structured nutrition information sources. All these sources prove that RAG technology can be integrated into our application.

2.1.3. Accuracy and Validation Studies of Existing Nutrition Tracking Applications

Chen et al. [7] evaluated the accuracy of the MyFitnessPal app's nutrient calculations by comparing user-entered data with the Nubel database in Belgium. They found strong correlations in calorie and nutrient calculations. They noted underestimations of protein (-7.8%) and carbohydrates (-6.4%).

Rodriguez et al. [8] compared five nutrition apps with traditional 3-day food diaries. They found that intakes such as energy and fat were generally underestimated, while protein was overestimated. This study suggests that the apps are generally functional, but that there can be occasional inaccurate assessments when tracking nutrients. These findings indicate the need for professional oversight of current nutrition apps. These findings suggest that current nutrition apps have limitations in accuracy for precise nutrient tracking.

3. ANALYSIS & DESIGN

In this chapter, I am going to cover the architectural design of the application. I am going to present this in an abstract way where any application team, regardless of their chosen platform or technology stack, can follow the design principles I apply here and achieve a similar or equivalent user experience to that provided in the application.

3.1. Use Cases of the Application

The use case diagram is one of the most important first steps when designing an application. The application provides personalized nutrition and tracking services. This system is shaped around 3 main actors that communicate with each other. These actors are the user, AI Agent and External API. As shown in Figure 3.1, the use case diagram, the user acts as the main actor that initiates all system interactions. AI Agent provides intelligent assistance and suggestions with tools suitable for users' queries, and External API's pull nutrition information and meal information from external sources.The user has various use cases such as chatting with the Agent, nutrition tracking, water tracking, shopping list management, viewing their profile, changing their information, profile information. Some of these use cases are expanded with extend relationships and create a modular system design. For example, the nutrition tracking use case includes adding food entries, seeing daily food intake, deleting food entries, and seeing how much they should take daily. The AI Agent actor in the system has various tools. This actor performs calculations such as macro, water, and calorie intake, obtains data from the vector database for RAG implementation, and uses information from the user's profile to provide specific answers to users. In order for the AI Agent to create a specific answer for the user, it must be able to see the user's profile, which creates an include relationship. In addition, the External API, which it is a last actor, also has an important place in the diagram.This actor has 2 features. One is to retrieve data from the Food USDA API and the other is to retrieve data from the Edamam API for the meal plan. These two features are mandatory for the AI Agent to provide nutritional information to users and create a meal plan, in other words, there is an include relationship between them. In addition, instead of entering a manual food entry, the user can contact the AI Agent and request that it be entered automatically. There is an extend relationship between the user's food entry addition use case and the AI Agent's nutrition details. In this expanding relationship, there is an include relationship between the AI Agent and the External API. Indirectly, the user and the External API interact. As a result, the relationships in the use case diagram enable secure access to the application, and aim to provide an enhanced user experience with include and extend relationships.

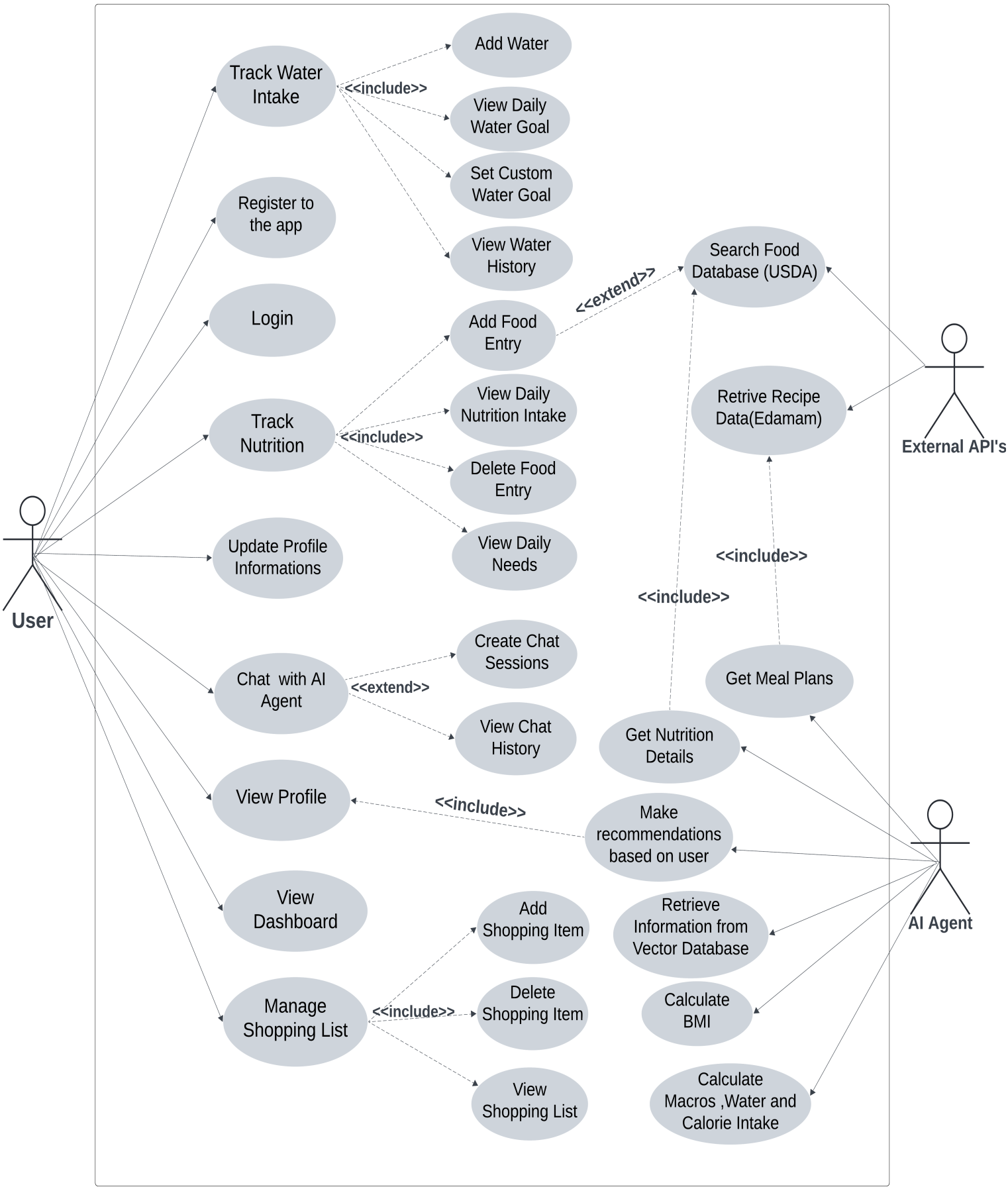


Figure 3.1: Use Case Diagram

3.2. Database Design and Entity Models

ER Model visually expresses the entities in the system, the relationships between these entities and their attributes. This provides an understandable foundation before creating the database. As we can see in the use case diagram in Figure 3.1, users will be able to consult AI and follow up on their daily needs such as food and water. For this reason, the user's application registration data, nutrition data, water data and chat data with the AI Agent are very important. In addition, it is a desired feature that users can see their chat history and create new sessions in the application, as I stated in the requirements. This data is critical for the functions in the application to work properly. For this reason, I designed this system to have a total of 8 main entities, as seen in the ER Model design in Figure 3.2.

If we examine this model, the user entity acts as a center that connects all functional modules in the system. Thus, a user-centered design approach is followed. This design philosophy ensures data consistency, provides access control to the application, and provides a logical basis for implementing personalized features and AI-focused suggestions. The design implements one-to-one relationships for user-specific configuration data and ensures that each user maintains a unique profile. This design allows all users to accumulate historical records in the required functional areas. In addition, temporal features are integrated throughout the model so that they can perform time-based analysis. This structure provides the necessary entities for user management, nutrition tracking, water tracking, and AI communication in the application. One-to-many and one-to-one relationships show the relationship between entities. When we look at the ER model, we can see that there are one-to-one relationships between the user entity and the user preferences entity and one-to-many relationships between the user and other entities. One-to-many relationships allow multiple records belonging to the user to be kept.In addition, the primary key is used in the model to define unique records for each table. This is done to prevent data from being repeated. In addition, foreign key structures are used to establish relationships between tables. For example, while the user_id is the primary key in the user entity, the same user_id is found as a foreign key in other entities. This approach provides a connection between the user and the user's data. Therefore, this database design provides a clear foundation for implementing the system, regardless of the database platform, programming language, or deployment environment that development teams choose.

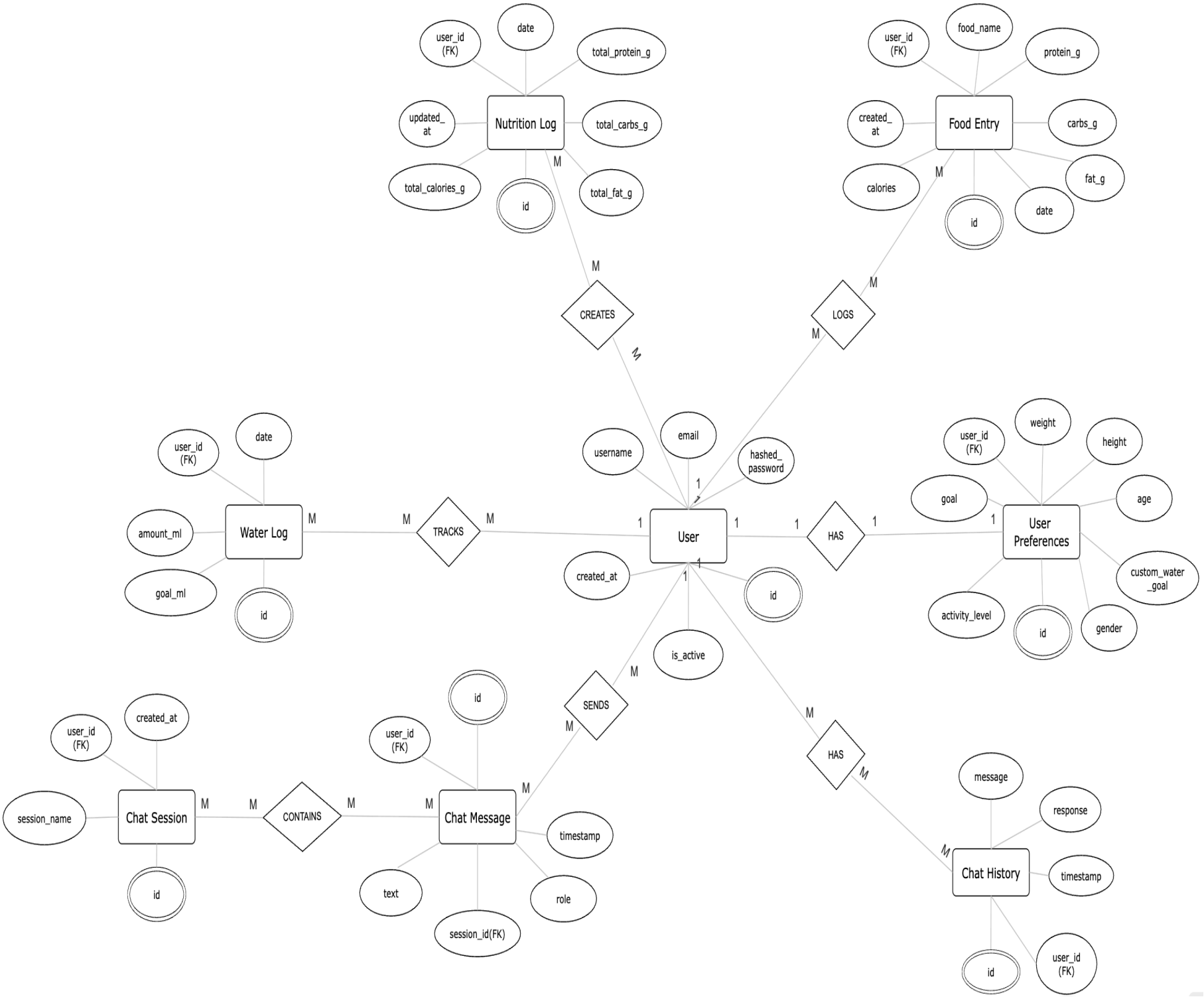


Figure 3.2: Entity Relationship (ER) Model

3.3. Object-Oriented Class Design

Figure 3.3 presents the class diagram of the structures required for the authentication and navigation structure of the mobile application. When we look at this class diagram, we can see that the MyApp class manages the general properties and initialization of the application. This class interacts with the Apptheme class to obtain the theme information of the application. It also provides navigation routes to the splash screen, login screen, registration screen and home screen. The SplashScreen class aims to show the application name to the user for a short time. The LoginScreen class is planned to log the user into the application, and the RegisterScreen class is planned to register the user in the application. This hierarchy shows that all authentication screens are inherited from a common base class. Thus, it is planned to obtain consistent behavior between the screens.

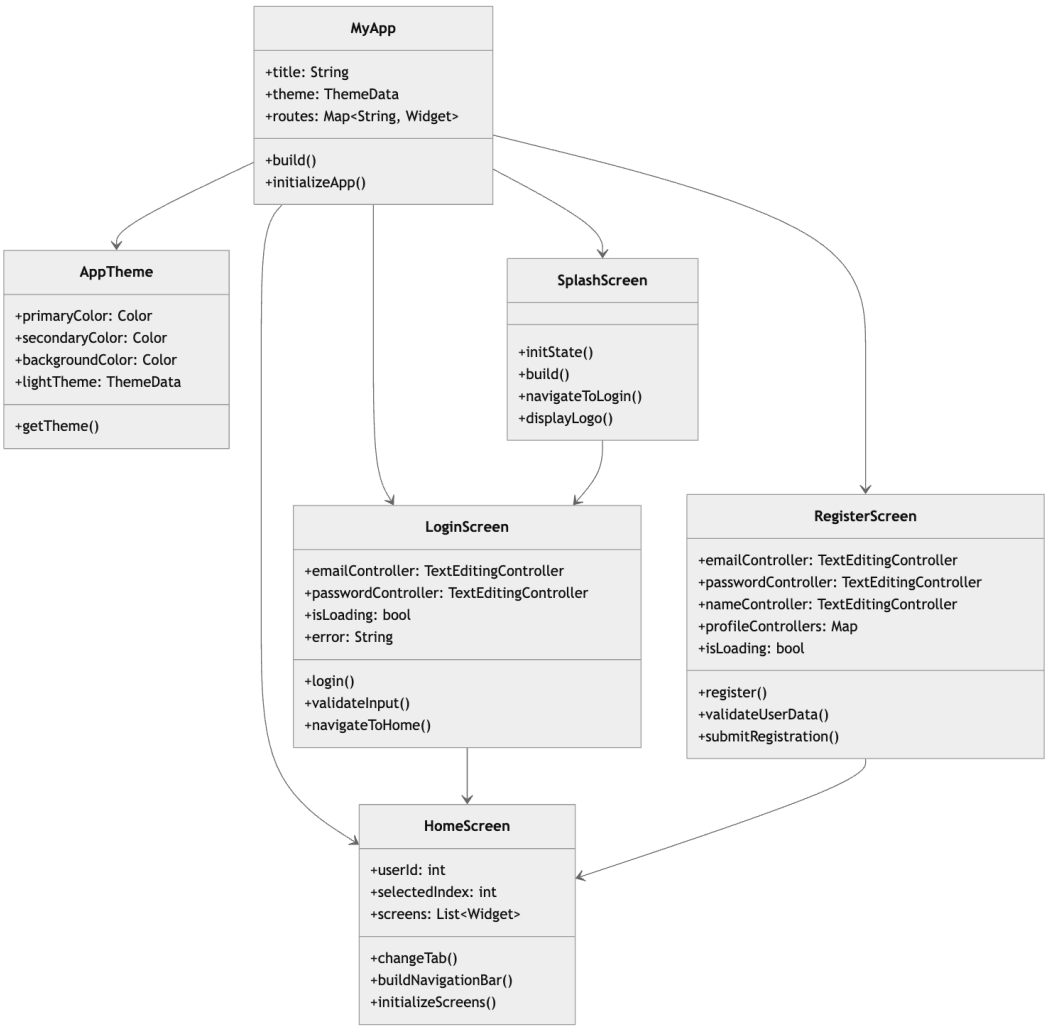


Figure 3.3: Authentication and Navigation Class Diagram

Now I am going to talk about the class design of the screen classes of the application. I divided this class diagram into two as Figures 3.4 and 3.5. When we examine the figures, we see that the functional screens in the application inherit from the AppScreen base class. There are several screen needs for application functionalities. I designed this class diagram for thinking these needs. This hierarchical structure is made to facilitate consistent data retrieval and user interface in functional areas. When we look at Figure 3.4, we can see that HomeScreen, DashboardScreen, ChatScreen and ChatSessionScreen in the application inherit from AppScreen. This structure allows the code to be modular, maintainable and readable.

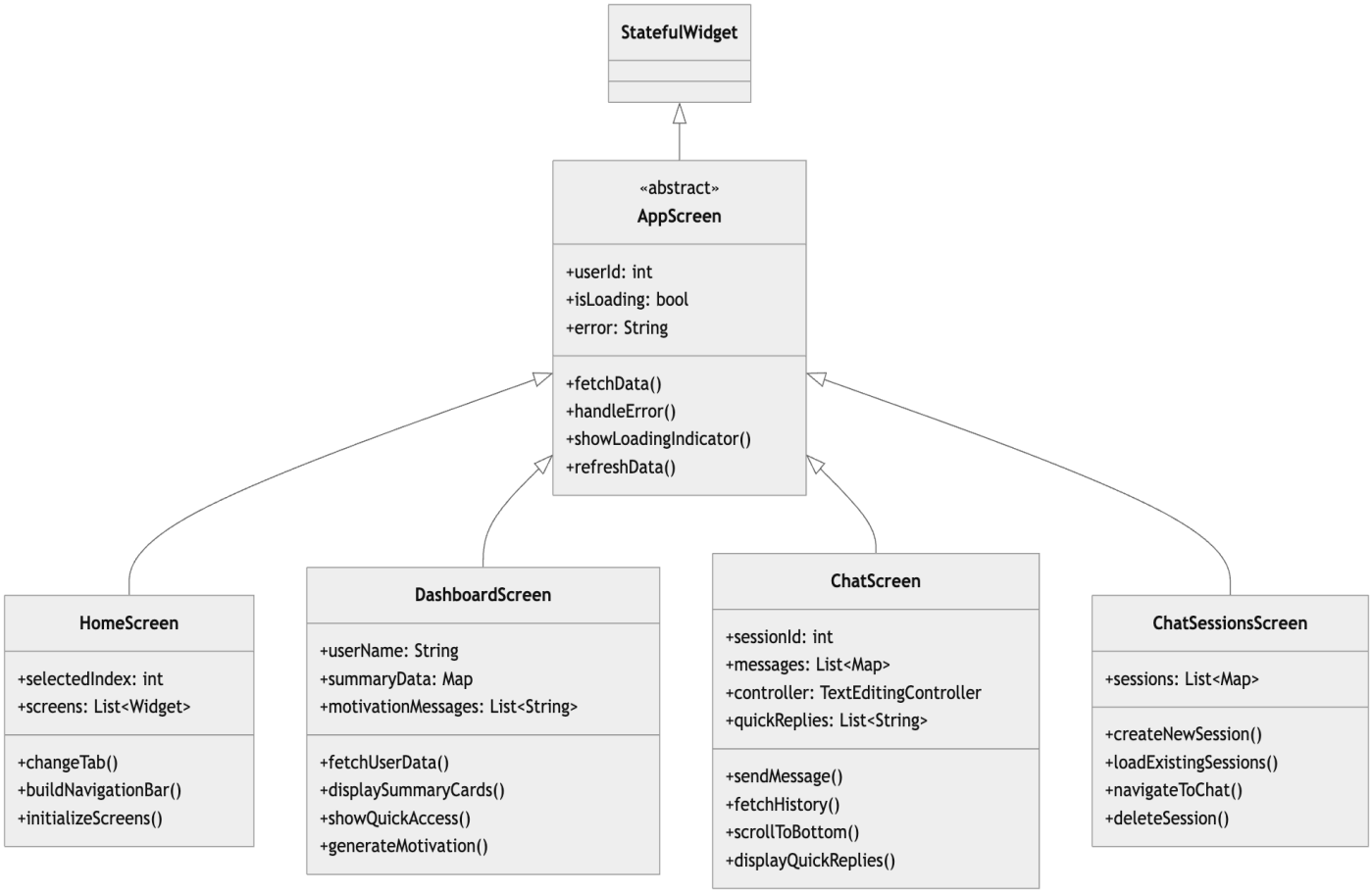


Figure 3.4: UI Class Diagram-1

Looking at Figure 3.5, which is a continuation of the class diagram in Figure 3.4, we can see again that the screen classes inherit from the AppScreen class. Additionally, the WaterScreen and CaloriesScreen classes are designed to inherit from the NutritionTrackingSystem class. This will allow the user to keep track their water and calories.This hierarchical structure ensures consistent data fetching, error handling, and user interface patterns across all functional areas while allowing domain-specific customizations for nutrition-related features.

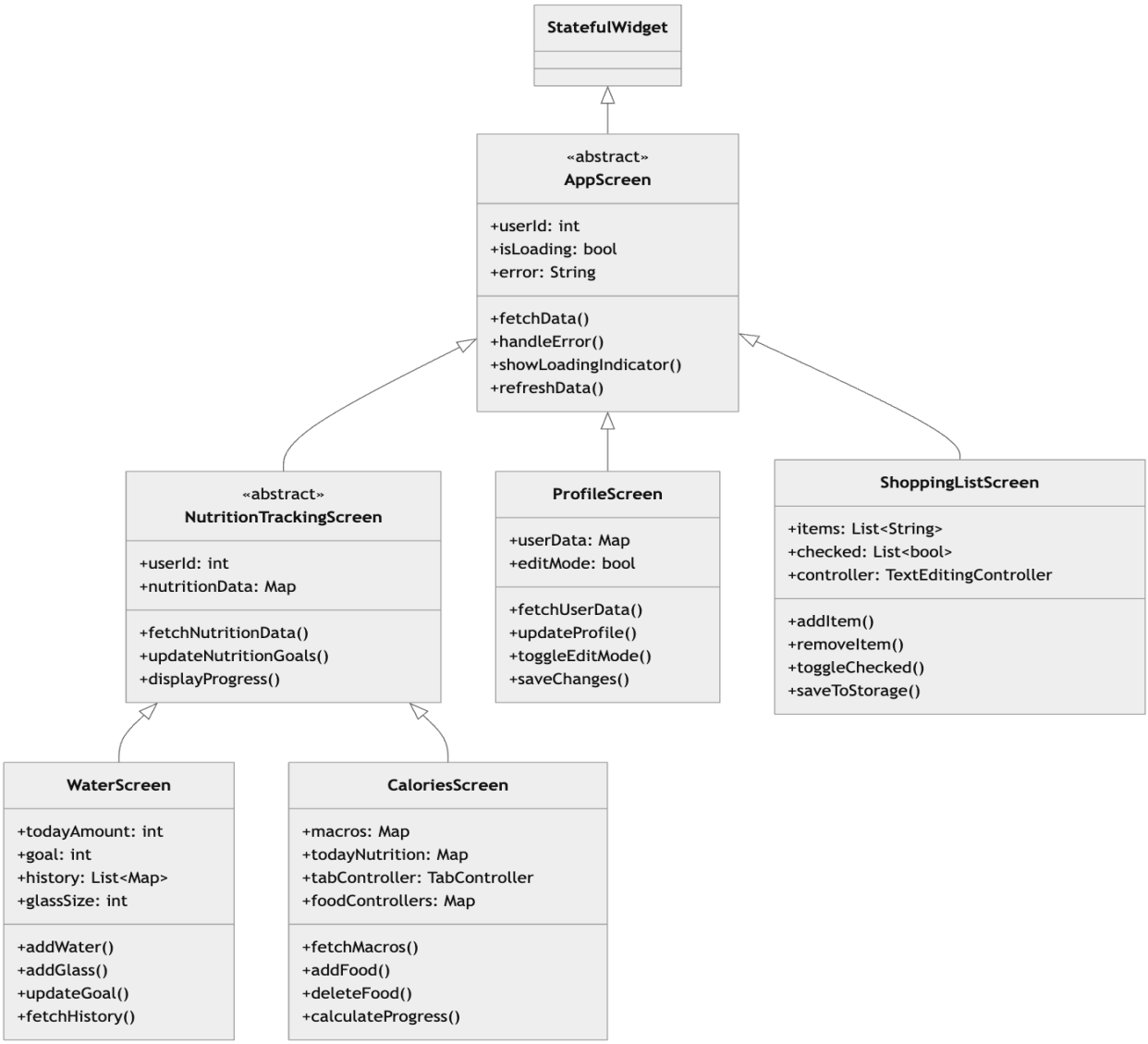


Figure 3.5: UI Class Diagram-2

One of the most important parts of the project is the AI Agent and its tools. Therefore, creating a class diagram for the AI Agent and its tools is very important for the design phase. I divided the class diagram into two parts, Figure 3.6 and Figure 3.7. My aim in this diagram is to comprehensively address the object-oriented architecture of the AI-based assistant. The DietAgent class, which is at the center of the system, is the main orchestrator that will manage all the tools and direct the queries coming from the user to the appropriate tool. This class is aimed to work with a series of BaseTool objects. The abstract BaseTool class in Figures 3.6 and 3.7 forms the common basis of all the tools. In other words, it defines the name, description and working functions for each tool.

I divided the AI Agent's tools into 4 main parts, thinking that it would be easier to separate them into different functional groups in terms of understandability and integration. When we look at the classes derived from BaseTool in Figure 3.6, we can see CalculatorTool and DataAccessTool. CalculatorTool is the basis of calculation-based tools. BMICalculator, CalorieCalculator, MacroCalculator and WaterCalculator classes are derived from this class. Each of them is designed to contain specialized functions. In other words, this class is planned to be used by users in calculation-related queries. DataAccessTool class is planned to be able to get information from external data sources. In this area, USDAFoodTool and EdamamMealPlannerTool classes are inherited from DataAccessTool class and are specialized. If we examine the other half of the diagram, Figure 3.7, we can see the KnowledgeTool class and UserInteractionTool class inherited from BaseTool in this part. I created KnowledgeTool for information queries. We see that DietRAGTool and WebSearch tool are derived from this class. I designed DietRAGTool especially so that it can retrieve the data that I will embed into the vector database and I aimed for Agent to search the internet with WebSearch tool. The reason I created the UserInteractionTool class is so that the AI Agent can do the daily tracking work and profile management of the user. I divided this class into 4 different classes. With FoodTrackingTool, it will allow the user to access and update the food tracking. The WaterTrackingTool class will be able to access the daily water tracking of the user and make additions. ProfileManagementTool is for the user to see and update their information. The SmartFoodEntryTool class is designed to pull and fill in the information of the food the user eats from the FoodUSDA API. These classes inherit from the UserInteractionTool class. Thus, instead of the user manually changing or adding information in the application, I aimed for the AI Agent to do it for the user.

I designed this class architecture in accordance with software engineering principles. Each tool is specialized in its own field, easily expandable and structured to be maintained. I ensured the modularity of the system. Thus, it has been made easier to add new tools and update existing ones.

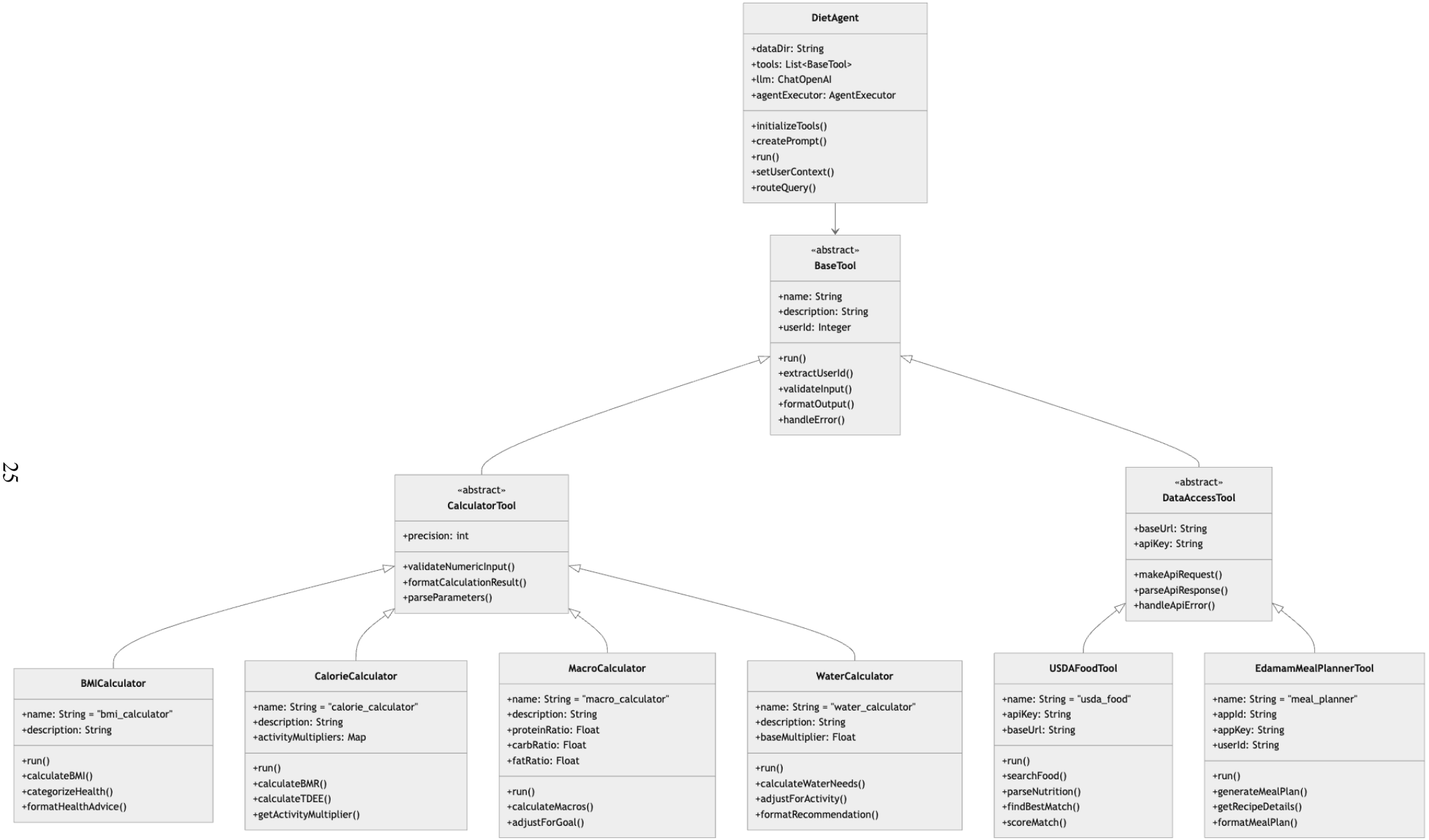


Figure 3.6: Agent and Tool Class Diagram-1

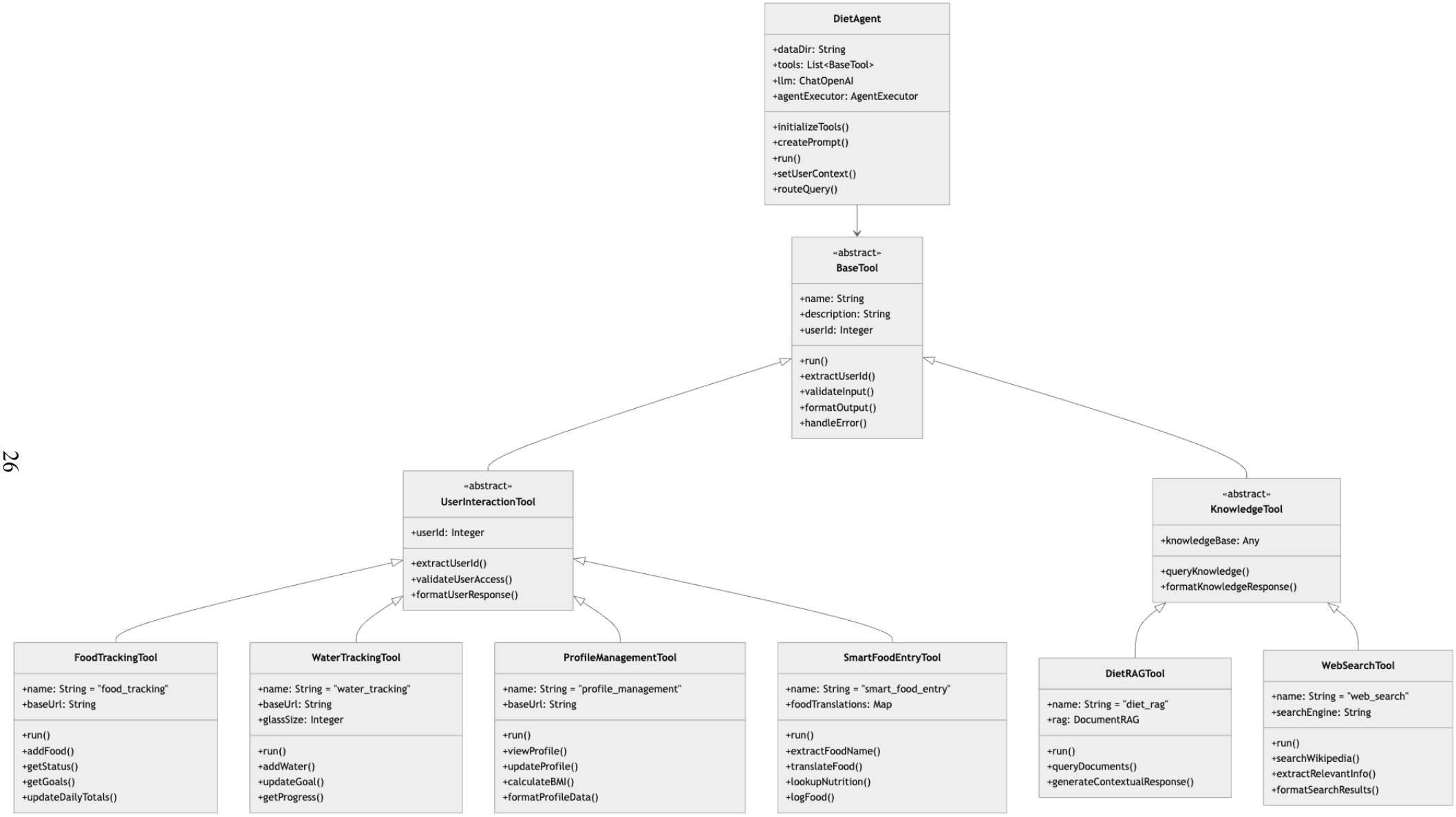


Figure 3.7: Agent and Tool Class Diagram-2

It is necessary to show the object-oriented structure of the API layer located on the back-end side of the application and the relationships between the main components. Figure 3.8 shows the API Class Diagram. When we examine this diagram, we see the FastAPI class, which manages all API endpoints and services, at the center of the system. This main application includes AI Agent, database session and the startup() function required for starting the application. I created 5 main API groups derived from FastAPI. I planned to create AuthAPI for user authentication and profile management processes, WaterAPI for water consumption-related processes, BusinessLogic for the application's business logic functions, NutritionAPI for nutrition-related processes and ChatAPI for chat-related processes. While each API group focuses on its own areas of responsibility, all APIs will also use DatabaseService. In addition, ChatAPI needs to connect to OpenAIClient for the language model, USDAClient for retrieving nutritional values and EdamamClient for nutritional planning, i.e. external APIs. This class architecture facilitates system maintenance and integration of new services.

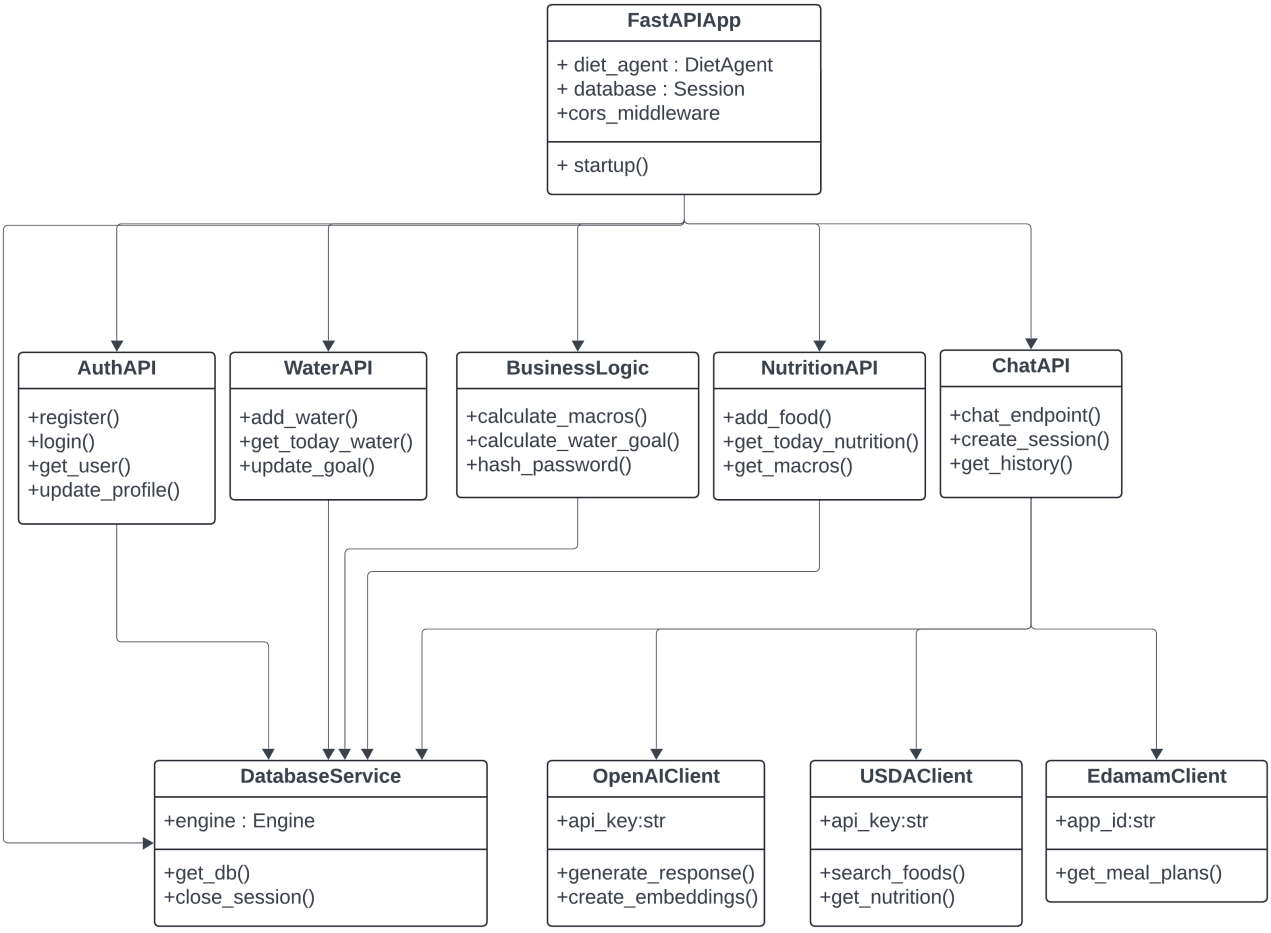


Figure 3.8: API Class Diagram

3.4. System Interaction Design

The diagram in Figure 3.9 shows the sequence diagram of the user registration process to the application. After the user enters the registration information, the frontend transmits this data to the backend. The backend checks if there is an e-mail record in the database. Here, two scenarios come into play. If the e-mail is not registered in the database, the user's record is created in the database and the registration is successful. In addition, if the user has entered the preference information, a user preferences record is created in the database. If the e-mail is registered in the database, the user is shown an error message.

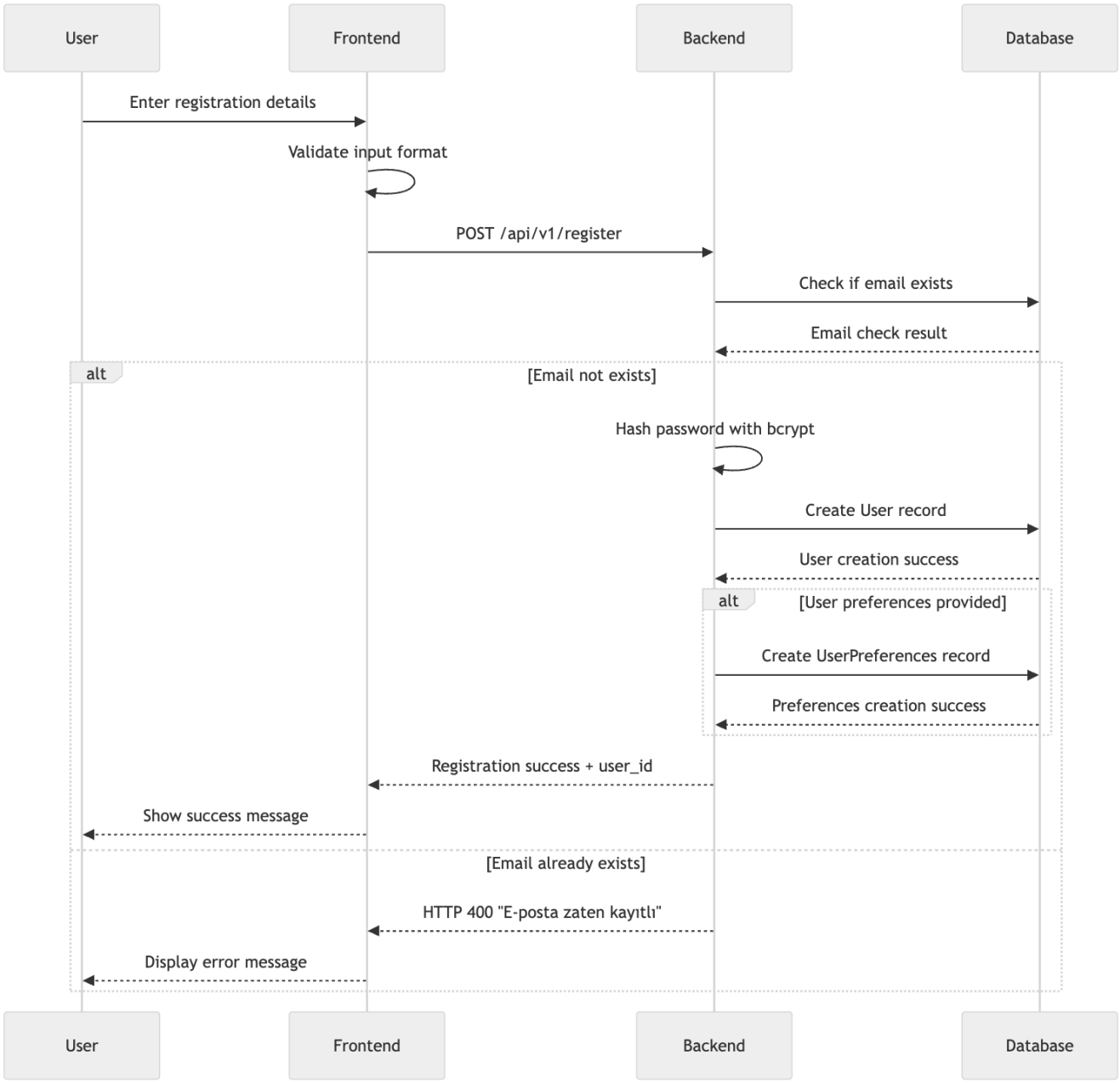


Figure 3.9: Register Sequence Diagram

The diagram in Figure 3.10 shows the sequence diagram design of the application login process. After the user enters their email and password information, the frontend verifies the format of this information and sends it to the backend. The backend checks if there is such a user in the database. If the user exists, the entered password is verified with the bcrypt algorithm. If the password is correct, the user's identity information is sent to the frontend. Thus, the user is directed to the main screen of the application. If the password is incorrect or the user is not found, an error is returned by the backend and displayed to the user. This process provides secure and controlled login.

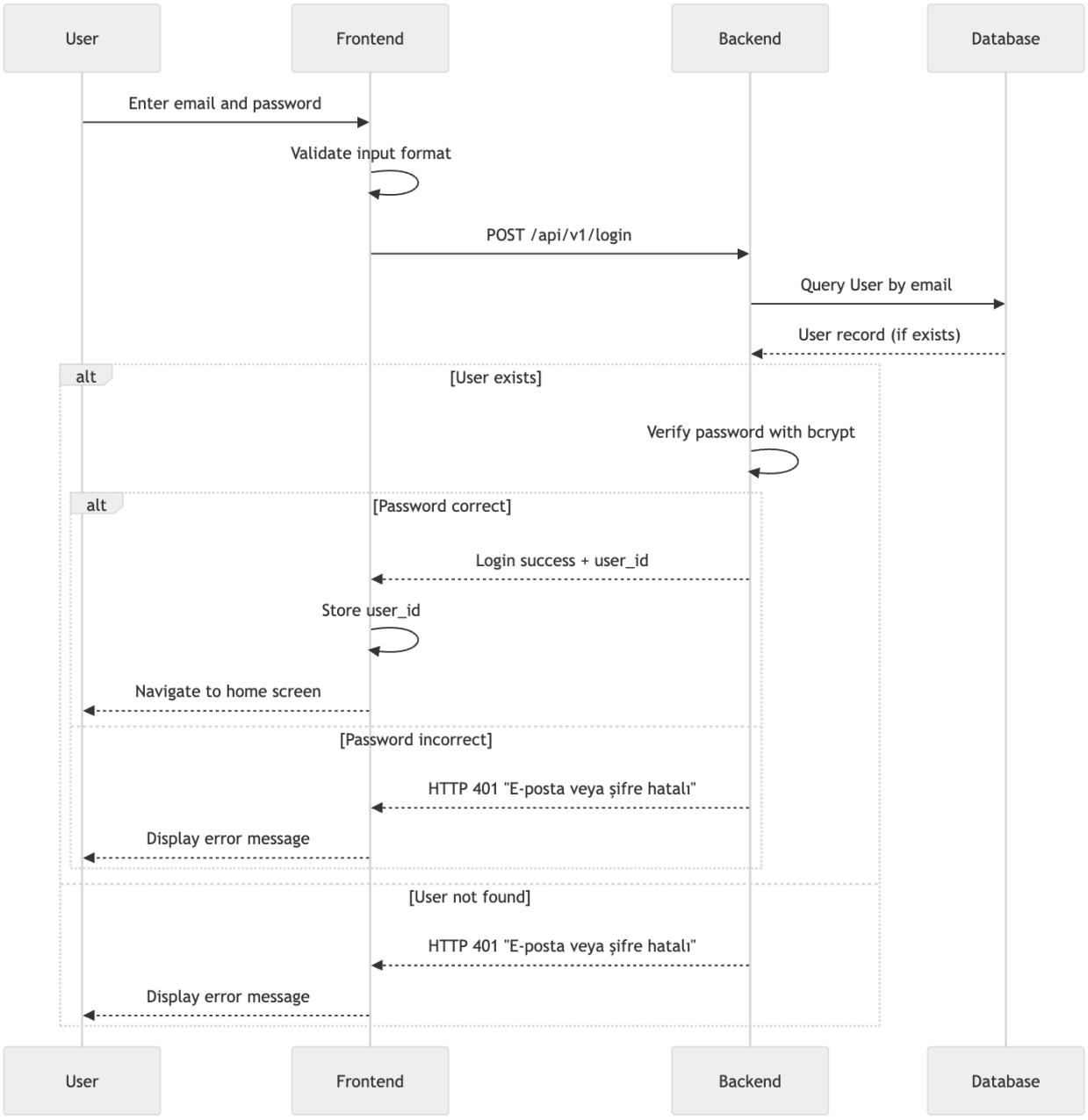


Figure 3.10: Login Sequence Diagram

I created a sequence diagram to show the interaction between the user and the chat system. As seen in the diagram in Figure 3.11, it shows how a chat message from the user is processed end-to-end. The message from the user is forwarded to the backend by the frontend. The backend retrieves the user's profile and preference data from the database and forwards the personalized message to the AI Agent. The agent calls the relevant tool and returns the result. In the meantime, the chat is saved to the database and the result is forwarded to the user.

I mentioned that AI Agent can pull data from external sources such as Food USDA API and Edamam API and use this data when responding to users. In Figure 3.12, we see the external API sequence diagram. When we examine this diagram, the natural language request from the user is transmitted to the AI Agent via the frontend. Here, the query type and which tool will be used for this query are determined. Since I will focus only on the flow from external sources in this diagram, we have 2 scenarios. If a nutritional information query is detected, the Agent will translate the Turkish input into English and send a search request to the USDA API through USDAFoodTool. The data returned from the API is parsed and presented to the user in a format appropriate for the user. If a meal plan is detected, this time it sends a request to the Edamam API via the AI Agent EdamamMealPlannerTool. The meal plan and details coming from the API are processed and made ready for the user. In both scenarios, the results are translated into Turkish and transmitted to the user. This process aims to reveal features such as the system's integration with external APIs and natural language processing and Turkish support.

The diagram in Figure 3.13 is a sequence diagram showing how the vector data-based information retrieval process is designed. A query from the user is forwarded to the AI Agent's RAG tool. This tool starts the process to provide the closest answer to the user's query. The user's query is embedded in the vector space through embeddings. A similarity search is performed between the data in the vector database and the user's query. The most relevant documents are determined from among the local documents. This data is forwarded to the AI Agent and the result is returned. This result is forwarded to the user.

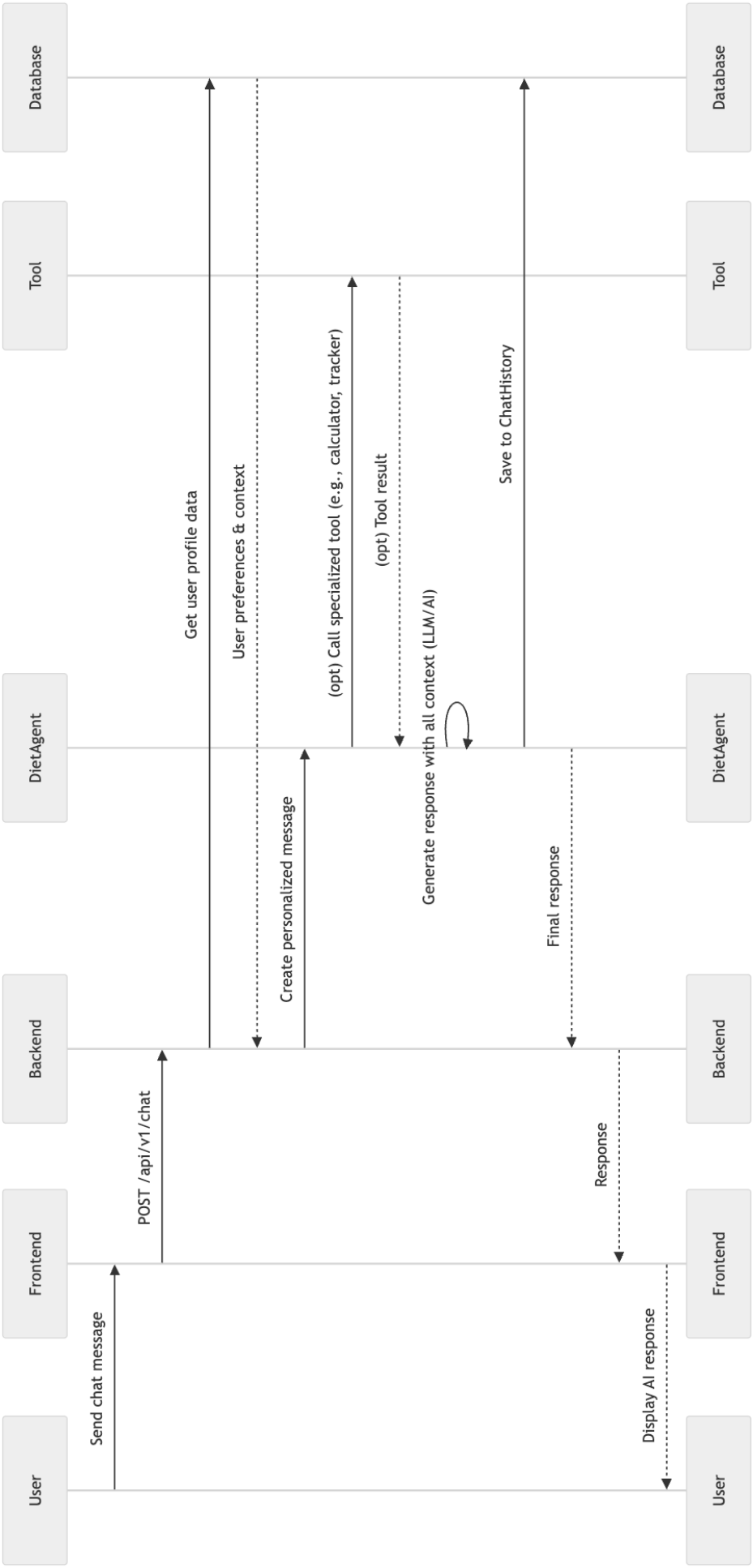


Figure 3.11: AI Agent Response Sequence Diagram

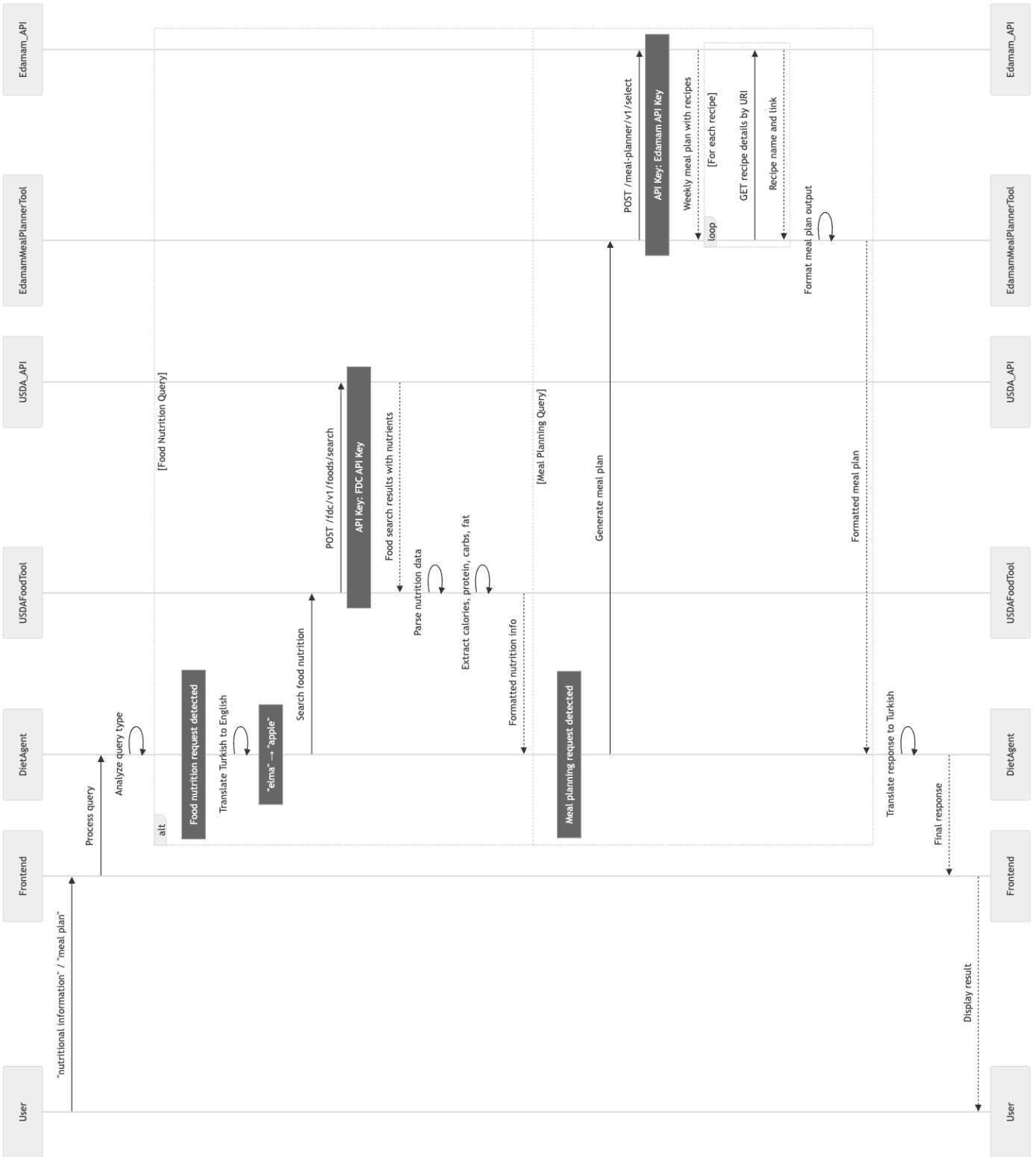


Figure 3.12: External API Sequences

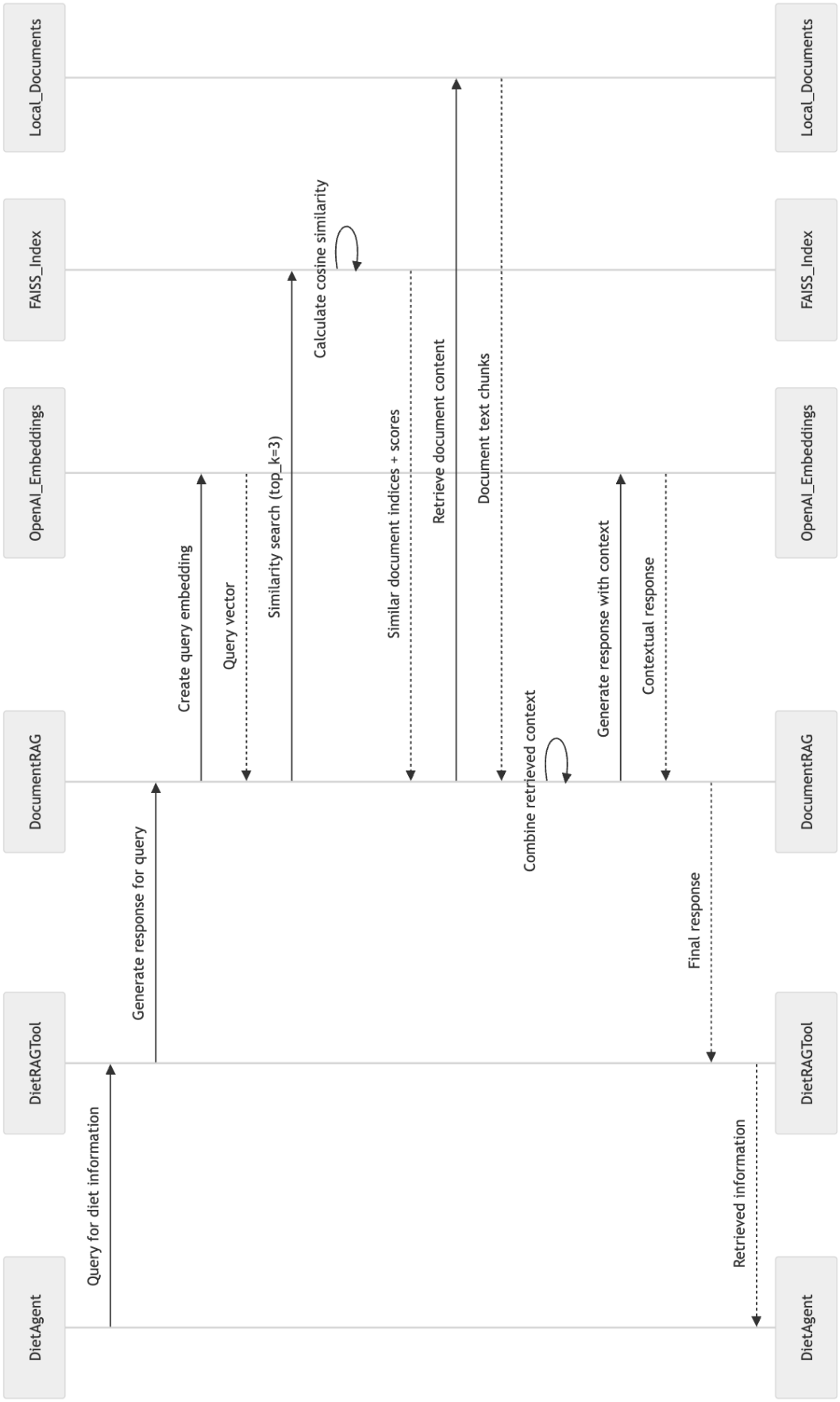


Figure 3.13: Vector Database Data Retrieval Sequence Diagram

3.5. Business Logic and Process Design

This section will include the pseudocode and flowcharts of the important algorithms of the application. This section will include the algorithm of the Personalized Agent system, which is one of the most important parts for my application, and the Agent's query processing and tool selection. First of all, I divided the pseudocode and flowchart of the personalized agent algorithm into 2 parts.

The first algorithm is one of the cornerstones of the application and is important for providing personalized guidance to the user. We can see the flow chart of the algorithm in Figure 3.14. The algorithm will start with user authentication and perform data collection processes such as demographic information, physical characteristics, and goals. After this data, the user's daily needs are calculated using scientific formulas. As a result of all these calculations, the information is synthesized and creates a context for the AI agent to produce personalized responses.

The second algorithm is a continuation of the first algorithm. We can see the flowchart of this algorithm in Figure 3.15. This algorithm analyzes user queries and produces the most appropriate answers. It analyzes user queries, extracts the necessary parameters, and determines whether the query requires special tools. If a special tool is required, the AI Agent selects the most appropriate one from among the available tools and runs it. After this step, the system customizes its response according to the user's personal context. In the final stage, the response is formatted with the user's profile information and goals and presented to the user.

Finally, the third and last algorithm is the AI Agent's query processing algorithm. There is a flow chart in Figure 3.16. The algorithm guarantees the stability of the system by working in a limited loop with maximum iterations. Thus, it is prevented from entering an unnecessary loop for a query with an unknown answer. The user query is analyzed and the query type is classified. It selects the most appropriate tool according to the categories of nutrient intake, water tracking, nutrient information, profile management, calculations and general information. After each tool is run, the success status is checked. In case of an error, the system uses a retry mechanism. After a successful tool run, the user's information, the original query and the tool result are combined to produce a special response. It is presented to the user in Turkish.

Algorithm 1 Personalized Context Builder - Part1

Require: user_identifier, incoming_message

Ensure: comprehensive_user_context

Identity and Access Control
user_profile ← authenticate_and_retrieve(user_identifier)
if user_profile IS NULL **then**
    return generic_context
end if
Multi-Dimensional Data Collection
demographic_data ← extract_demographic_info(user_profile)
physical_attributes ← extract_physical_metrics(user_profile)
goals_preferences ← extract_objectives_and_preferences(user_profile)
Data Completion and Validation
validated_data ← validate_and_complete_data(
        demographic_data,
        physical_attributes,
        goals_preferences)
Scientific Calculations
health_metrics ← calculate_health_indicators(validated_data)
metabolic_profile ← calculate_metabolic_requirements(validated_data)
nutritional_targets ← derive_nutritional_goals(health_metrics, metabolic_profile)
Goal-Oriented Strategy Determination
personal_strategy ← determine_personalization_strategy(
        goals_preferences.primary_objective)
Contextual Information Synthesis
comprehensive_context ← synthesize_context(
        validated_data,
        health_metrics,
        metabolic_profile,
        nutritional_targets,
        personal_strategy)
return comprehensive_context

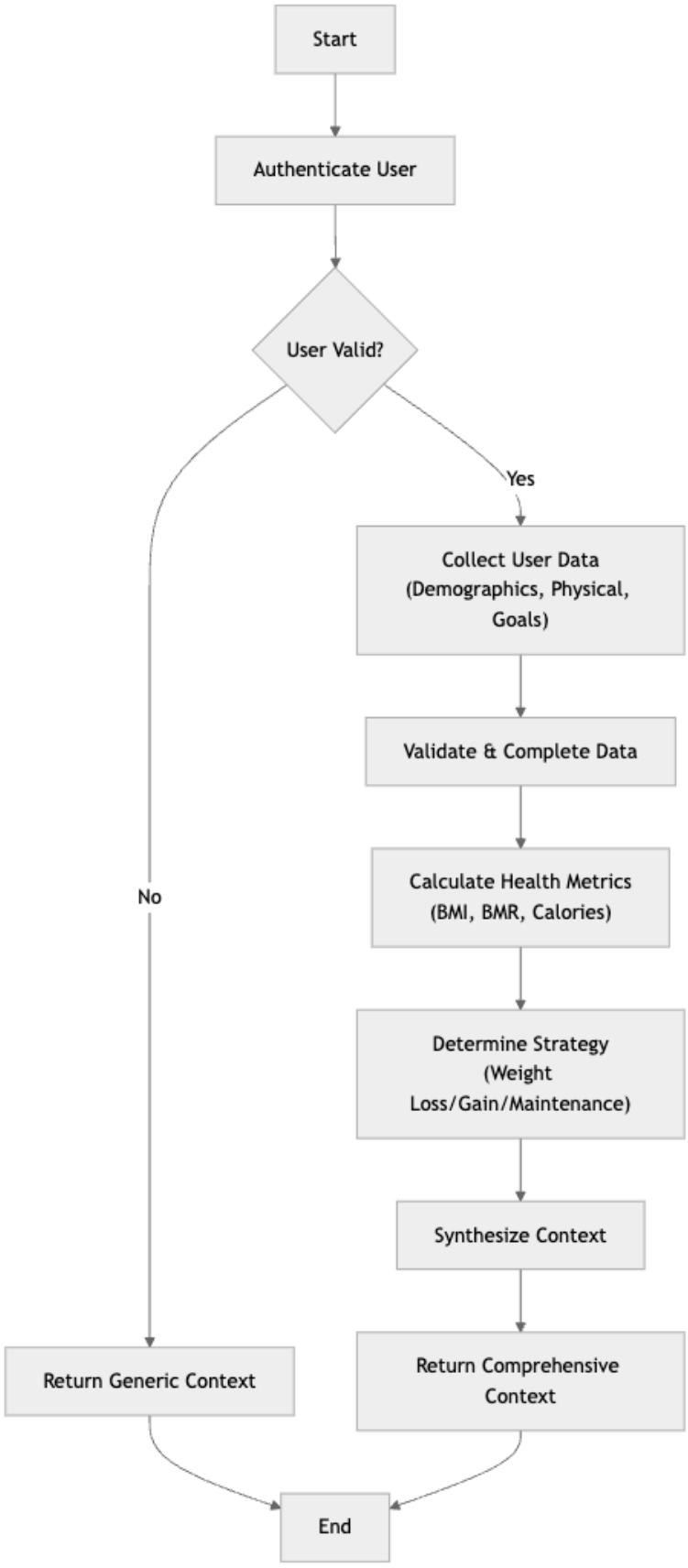


Figure 3.14: Personalized Context Builder Flow Chart - Part1

Algorithm 2 Intelligent Response Generator- Part2

Require: user_message, comprehensive_context

Ensure: personalized_response

Intent Recognition and Classification
message_intent ← analyze_user_intent(user_message)
intent_category ← classify_intent(message_intent)
Parameter Extraction and Validation
action_parameters ← extract_relevant_parameters(user_message, intent_category)
validated_parameters ← validate_with_context(action_parameters, comprehensive_context)
Intelligent Tool Selection
optimal_tool ← select_specialized_tool(intent_category, validated_parameters)
fallback_tools ← identify_alternative_tools(intent_category)
Context-Enhanced Execution
raw_response ← execute_tool_with_context(
    optimal_tool,
    validated_parameters,
    comprehensive_context)
raw_response ← execute_fallback_strategy(fallback_tools, validated_parameters)
Multi-Layer Personalization
strategy_aligned_response ← apply_goal_strategy(
    raw_response,
    comprehensive_context.personal_strategy)
motivational_response ← enhance_with_motivation(
    strategy_aligned_response,
    comprehensive_context.progress_indicators)
educational_response ← add_educational_value(
    motivational_response,
    comprehensive_context.knowledge_gaps)
Final Response Optimization
actionable_response ← synthesize_actionable_recommendations(
    educational_response,
    comprehensive_context.current_status)
validated_response ← quality_assurance_check(actionable_response)
return validated_response

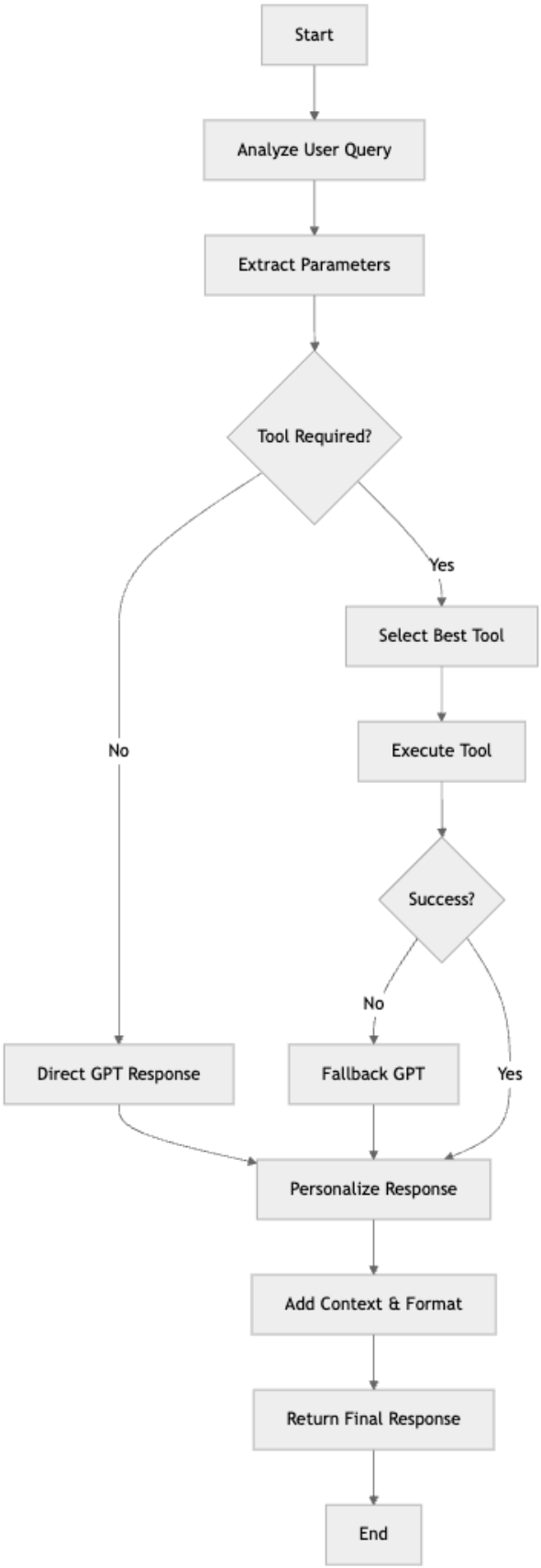


Figure 3.15: Intelligent Response Generator Flow Chart - Part2

Algorithm 3 DietAgent_Query_Processing

Input: `user_query (string)`

Output: `final_response (string)`

```
max_iterations ← 30
current_iteration ← 0
execution_success ← false
agent ← DietAgent.initialize()
agent_executor ← AgentExecutor.invoke(user_query)
Define list available_tools
while current_iteration < max_iterations AND NOT execution_success do
    query_type ← CLASSIFY_QUERY(user_query)
    if query_type == "natural_food_entry" then
        selected_tool = Smart_Food_Entry_Tool
    else if query_type == "water_tracking" then
        selected_tool = Water_Tracking_Tool
    else if query_type == "food_nutrition" then
        selected_tool = USDA_Food_Tool
    else if query_type == "profile_management" then
        selected_tool = Profile_Management_Tool
    else if query_type == "calculations" then
        selected_tool = BMI_Macro_Calorie_Calculator
    else if query_type == "general_knowledge" then
        selected_tool = Diet_Information_RAG_Tool
    else
        selected_tool = Web_Search_Tool
    end if
    Attempt to execute tool
    tool_result = EXECUTE_TOOL(selected_tool, user_query)
    if tool_result has error then
        LOG_ERROR(error_details)
        current_iteration = current_iteration + 1
        if current_iteration >= max_iterations then
            return "Error: Maximum iterations reached."
        end if
    else
        execution_success = true
    end if
end while
if execution_success then
    if DETECT_LANGUAGE(user_query) == "Turkish" then
        translated_result = TRANSLATE_TO_TURKISH(tool_result)
    else
        translated_result = tool_result
    end if
    final_response = GENERATE_RESPONSE(translated_result, user_query)
    return final_response
else
    return "Error: Unable to process query."
end if
```

Figure 3.16: AI Agent Query Processing and Tool Selection

4. IMPLEMENTATION

In this chapter , I am going to describe how I have implemented the application.I had analysed and designed in the previous chapter. I am going to highlight the actual technology stack used and define the implementation details based on the real codebase structure.

4.1. The Backend Implementation

I built the backend of my application on FastAPI. The backend of my application includes data management, artificial intelligence tasks and the logic of my entire application. I used RESTful API principles in this architecture. Thus, I provided optimal performance and scalability for asynchronous operations.I designed the backend system in a modular way by separating it into data access, business logic and presentation layers. If I need to explain the layers, the data access layer communicates with the database, the business logic layer determines the decision mechanisms of the application and the presentation layer is the layer that the user directly interacts with. I used SQLAlchemy as an object-relational mapping tool in my application, which provided an abstraction layer between the application and the PostgreSQL database I used. With this implementation, I efficiently handled data integrity, complex queries and database relationships. I chose the FastAPI framework because it is useful for automatic API documentation generation, data validation and asynchronous operations thanks to Pydantic models. The backend implements features such as error handling mechanisms, request validation and response serialization to ensure reliable communication with the frontend. It also includes COSR (Cross Origin Resource Sharing) to ensure reliable communication with the mobile application and the backend.

4.2. Languages and Frameworks

I carefully selected the technologies I used to build my project on performance, sustainability and scalability. I chose Python as the backend language because it has extensive libraries, is easy to integrate and provides machine learning. Python's asynchronous capabilities were used so that the backend of my application could execute simultaneous requests simultaneously. I used the Flutter framework for my mobile application, which is built on the Dart language. The reason I chose Flutter is that it can work cross-platform, meaning it allows easy development of iOS and Android applications. It also offers a wide range of features for responsive and modern user interfaces. I used the LangChain framework to create my AI Agent and tools. This framework was chosen because it facilitates the integration of language

models, embedding systems and AI agents and tools. In addition, a vector database had to be used to use my RAG (Retrieval Augmented Generation) system. I used FAISS (Facebook AI Similarity Search) as a vector database to quickly retrieve relevant information from large data sets and perform vector similarity operations.

4.3. Database Implementation

In this section, implementation details of the database systems used in the application are presented. In order to provide the functional and artificial intelligence-based requirements of the system, both a relational database and a vector database have been integrated. In this section, I will talk about their implementations in two parts, RDBMS and vector database.

4.3.1. Relational Database Management System Implementation

In the application, PostgreSQL, a relational database management system, is used for database implementation. It includes ACID compliance, complex query support and performance features. The database has been implemented in the most optimal way to ensure data integrity and query efficiency. The database schema of the application is designed to provide necessary functions such as user information, user management, nutrition tracking, water consumption tracking and AI chat histories. In this implementation, foreign key relationship has been used to ensure integrity between tables.

The database structure consists of 8 main tables in accordance with normalization rules:

- **User Management Tables:**
  - **users:** Contains user authentication information, account status control and registration date.
  - **user_preferences:** Contains personal information such as age, gender, weight, goal and activity level of the user.
  - A One-to-One relationship has been established between these two tables.

- **Nutrition Tracking Tables:**
  - **food_entries:** This table stores detailed daily nutrition data of users such as food_name, protein_g, carbs_g, fat_g and calories.

  - **nutrition_logs:** Contains daily total nutrition values of users such as total_protein_g, total_fat_g, total_carbs_g and total_calories.
  - A One-to-Many relationship exists between these two tables.

- **Water Tracking Table:**
  - **water_log:** Stores daily water consumption data of users with amount_ml, goal_ml and date information.

- **Chat Tables:**
  - **chat_sessions:** This table allows users to organize their conversations with AI by dividing them into different sessions.
  - **chat_messages:** This table keeps a record of all messages between the AI and the user.
  - **chat_history:** This table ensures that old chat data is not lost.

In addition, appropriate data types and rules have been applied to ensure data integrity and consistency. Alembic tool, which works integrated with SQLAlchemy, has been used to provide control for changes made to the database and schema updates.

4.3.2. Vector Database Implementation

In the project, FAISS (Facebook AI Similarity Search) vector database has been implemented as a second database layer. This system stores the vectorized versions of the documentation that constitutes the knowledge base of the AI Agent. In other words, this vector database contains embeddings of various nutrition, health and sports related data. The RAG system processes csv and pdf format files and divides them into chunks, and each chunk is embedded into the FAISS vector database using OpenAI's text-embedding-3-small model. Thus, semantic similarity search is performed between the user's queries and the data.

4.4. AI System Implementation

One of the most important parts of the application is the artificial intelligence system, an AI agent created using the LangChain framework. I chose LangChain for its flexibility in

the field of artificial intelligence and the possibility of easy integration with different tools. I want to describe AI agent for better understanding. The agent analyzes user queries, selects the appropriate tools and uses the tools. It synthesizes the results and provides meaningful answers to the user. AI Agent uses the ReAct (Reasoning and Acting) approach in this process. It analyzes the problem in the Thought phase, selects and runs the tool to be used in the Action phase, and analyzes the results in the Observation phase and determines the next steps. This cycle is repeated iteratively for complex queries. In addition, the role and capabilities of the AI agent are defined by the system prompt. Thus, the agent behaves correctly and consistently in every interaction. The AI agent uses the RAG (Retrieval-Augmented-Generation) architecture, which combines the process of retrieving information from the FAISS vector database with a large language model. OpenAI's 'gpt-4-mini' model was preferred for natural language processing and text generation. The RAG application processes large data sets in the fields of nutrition, sports and health, and these data are embedded into the FAISS vector database with OpenAI's 'text-embedding-3-small' model. According to the users' queries, a similarity comparison is made with the data in the FAISS vector database, preserving the context. The data closest to the query is retrieved. These data are synthesized with the gpt-4o-mini (LLM) model in the project and presented to the user in the form of meaningful and context-appropriate answers. I have also integrated various tools other than RAG into AI Agent architecture. I will explain these tools one by one:

BMI_Calculator: Calculates the body mass index of the users.

Diet_Information: Retrieves information from the vector database according to the user's query with RAG.

Calorie_Calculator: Calculates the daily calorie intake of the users.

Macro_Calculator: Calculates the daily macro intake of the users.

Water_Calculator: Calculates how much water users should drink per day.

Meal_Planner: Creates a meal plan for the users' diet type for as many days as they want using Edamam API.

Web_Search: Searches from Wikipedia.

USDA_Food: Retrieves food information securely using Food USDA Central API.

Water_Tracking: Users can consult AI about their daily water intake and do not have to manually enter water.

Food_Tracking: Users do not have to manually enter food, AI can do it for users.

Profile_Management: AI can see users' profile information and thus make personalized recommendations. Also, the user can edit the user's profile information if they want.

Smart_Food_Entry: If the food the user eats is in the Food USDA database, AI can look it up and fill in the calorie and macro information for the user.

AI Agent selects the most appropriate tool based on the queries asked by users in the chat section and responds to the user using that tool.

4.5. The Frontend Implementation

The mobile application was developed with the Dart programming language using Google's Flutter framework for cross-platform development. The main reasons I prefer Flutter are that it can work as multi-platform (iOS, Android, Mac, Web) and has a rich UI variety. The code organization creates a hierarchy with screens, widgets, services, models and utils subfolders under the lib folder. This structure increases the sustainability of the code. The application offers a modern, user-friendly and consistent UI. You can see the screenshots of the application in the appendix chapter, in figures 7.1 and 7.2. I aimed to improve the user experience with progress bars, calorie indicators, water tracking progress and interactive dashboards. There are a total of 11 main screens in the application. These screens:

(i) **Splash Screen (splash_screen.dart):**

- This screen is designed as the first door to the application and the name of the application is shown on this screen. After the screen is shown for 2 seconds, it directs users to the home screen so that they can log in.

(ii) **Login Screen (auth/login_screen.dart):**

- This screen is the main screen that manages the user authentication process. It contains 2 input fields for users to enter their e-mail and password to log in to the application. If the user is not registered to the application, the 'Don't have an

account? Register' button directs users to the registration screen. If users enter their e-mail or password incorrectly, they will encounter an error message.

(iii) **Register Screen (auth/register_screen.dart)**

- This screen allows users to fill in the information required for registration, such as name, e-mail, password, gender, age, height, weight, goal and activity selection. This information is very important for users' personalized goals.

(iv) **Home Screen (home/home_screen.dart):**

- This screen is designed as the main navigation center of the application and manages all other screens. It provides navigation between 6 different main sections.

(v) **Dashboard Screen(home/dashboard_screen.dart)**

- This screen acts as the central hub of the application. It collects all the user's basic metrics on a single screen. Synchronization with real-time data is provided. There are progress bars for users to track daily water and calories, shows the completed item and total item rate for the shopping list. There are also motivational messages every time to motivate the user.

(vi) **Calories Screen (home/calories_screen.dart))**

- This screen provides a comprehensive tracking system with a 3-tab architecture. The user can see their personal macro goals that they need to take daily according to their information, add food, and see all their nutritional entries.

(vii) **Water Screen(water/water_screen.dart)**

- This screen is designed as a personalized hydration tracking system. It allows for daily water consumption and goal comparison. Users' goal information is calculated according to their weight, but they can manually change their goals if they want. Users can add the amount of water they drink to be 1 glass of 200 ml. A special congratulatory message is shown when the goal is reached.

(viii) **Shopping List Screen(home/shopping_list_screen.dart)**

- This screen is designed for dynamic shopping list management for users. The screen is supported by the functionality of adding and removing items. The completion status for each item is provided with a checkbox.

(ix) **Chat Screen(chat/chat_screen.dart)**

- This screen is designed for the user to chat with the AI Agent. It differentiates the chatbot messages with the user. This UI allows users to have real-time messaging with the AI Agent. Based on users' queries, the AI Agent's context-aware responses are delivered to the user.

(x) **Chat Sessions Screen(chat/chat_sessions_screen.dart)**

- This screen is designed to manage multiple chat sessions. It allows users to have separate conversations. This screen includes features such as viewing the session list, creating new sessions, and managing existing sessions. Past conversations can be easily accessed.

(xi) **Profile Screen(profile/profile_screen.dart)**

- This screen is a comprehensive settings screen designed for user profile management and personal information editing. The user's basic information such as name, age, weight, and goal are displayed on this screen. The user can update this information if they wish. If they update the information required for water and calorie tracking, daily needs are automatically recalculated.

5. TESTS AND RESULTS

The testing phase is one of the most critical steps in the software development process. In this step, it is indispensable to objectively evaluate the system's compliance with its requirements, its reliability, and the experience it offers to users. In this context, the evaluation criteria are response time, usability, scalability, accuracy, and response time. In addition, tests are required for functional and non-functional requirements in the project. In this section, I will discuss the tests and their results.

5.1. Scalability and Performance Testing of the Application

Scalability describes the ability of a system to handle increasing workloads. I implemented a multi-tiered load test with JMeter to evaluate the scalability limits of the application.The test was designed with four different load levels to simulate real-world usage. These are Light load, Medium load, Heavy load and Stress load. Light load was tested with 10 concurrent users for 30 seconds, Medium load with 25 users for 60 seconds, Heavy load with 50 users for 90 seconds, and finally Stress load with 100 users for 120 seconds. Each test phase targeted different API endpoints including user management, nutrition tracking, water tracking, and main endpoint. Basic performances such as response time, success rate, throughput, and error distribution were measured with this test.

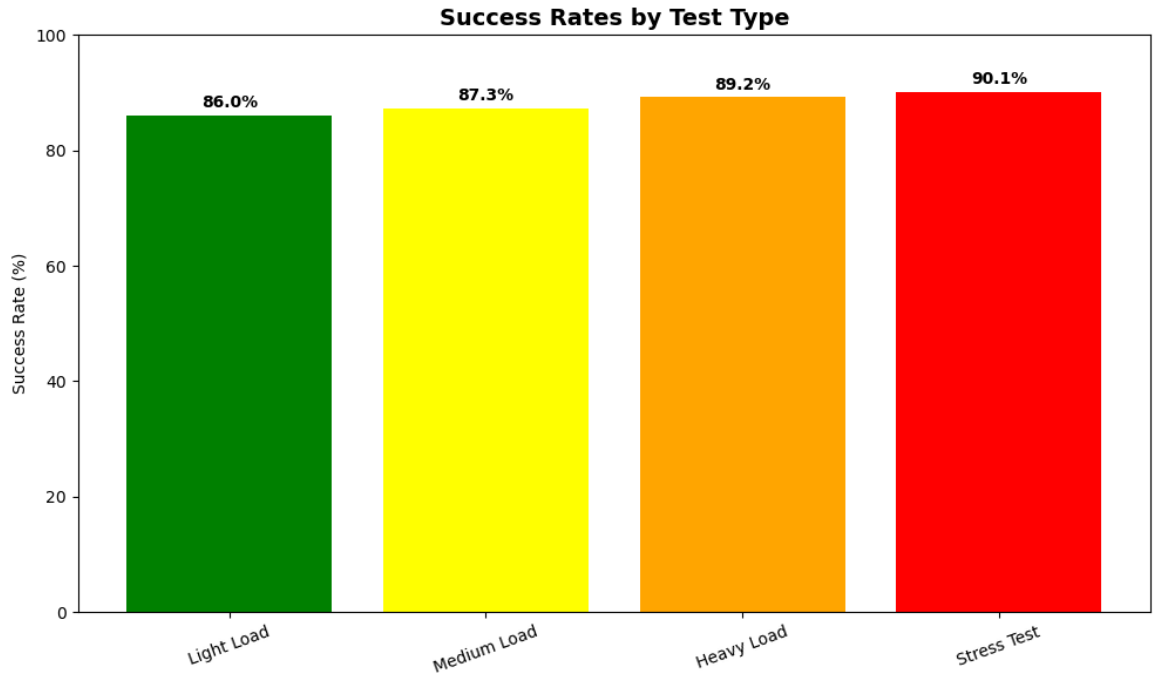


Figure 5.1: Scalability of Different Loads

Figure 5.1 shows the success rates for each test load. When we look at this graph, we see that the success rates for light, medium and heavy loads are very close to each other. As the load increases even more, we see an increase.This shows that the application can handle different users efficiently and protect its scalability.

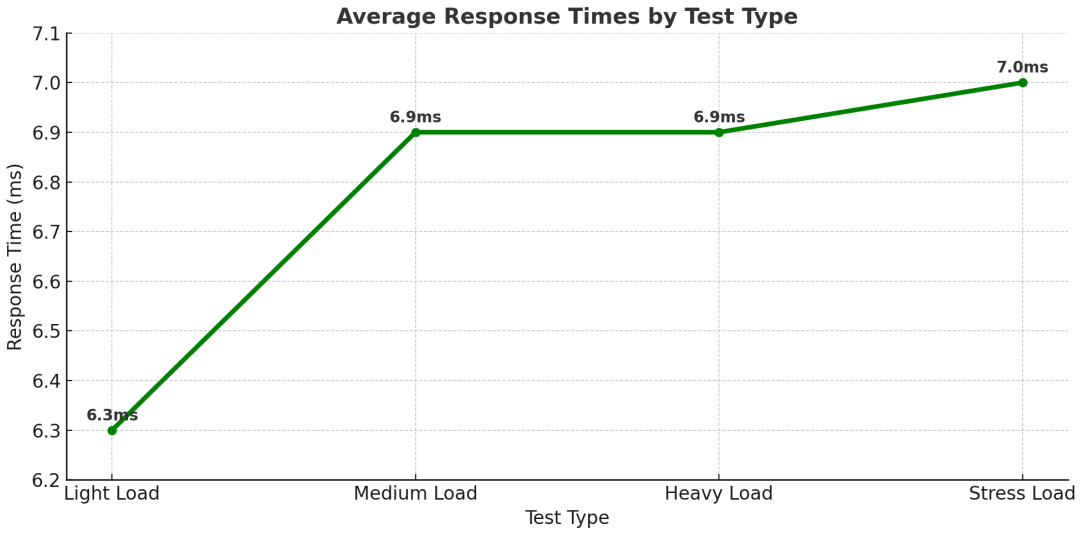


Figure 5.2: Response Times of Different Loads

Figure 5.2 was created to see how response times change at different load levels. When we look at this graph, we see that response times are in the range of 6.3-7.0 ms at all test levels. This shows that the system is stable in terms of latency and that the increase in load does not affect the response time much.

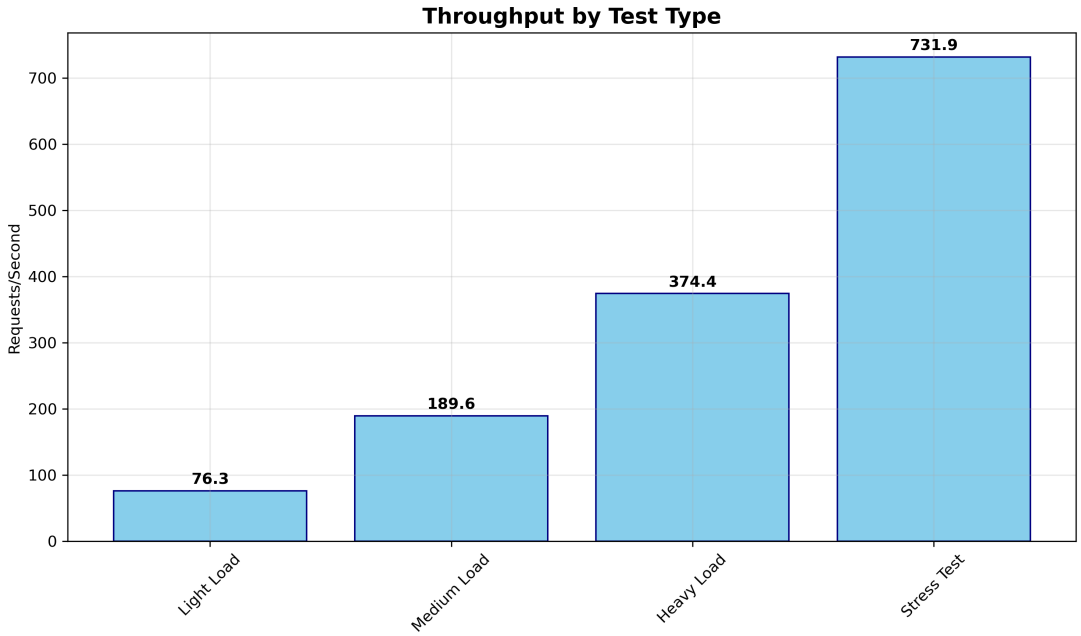


Figure 5.3: Throughput by Different Loads

Figure 5.3 shows a graph that measures the number of requests the system can process per second. When we examine this graph, we see that the throughput values increase from 76.3 to 731.9 requests/second as the load increases from light load to stress load. This shows that the parallel processing capacity of the system increases with the load.

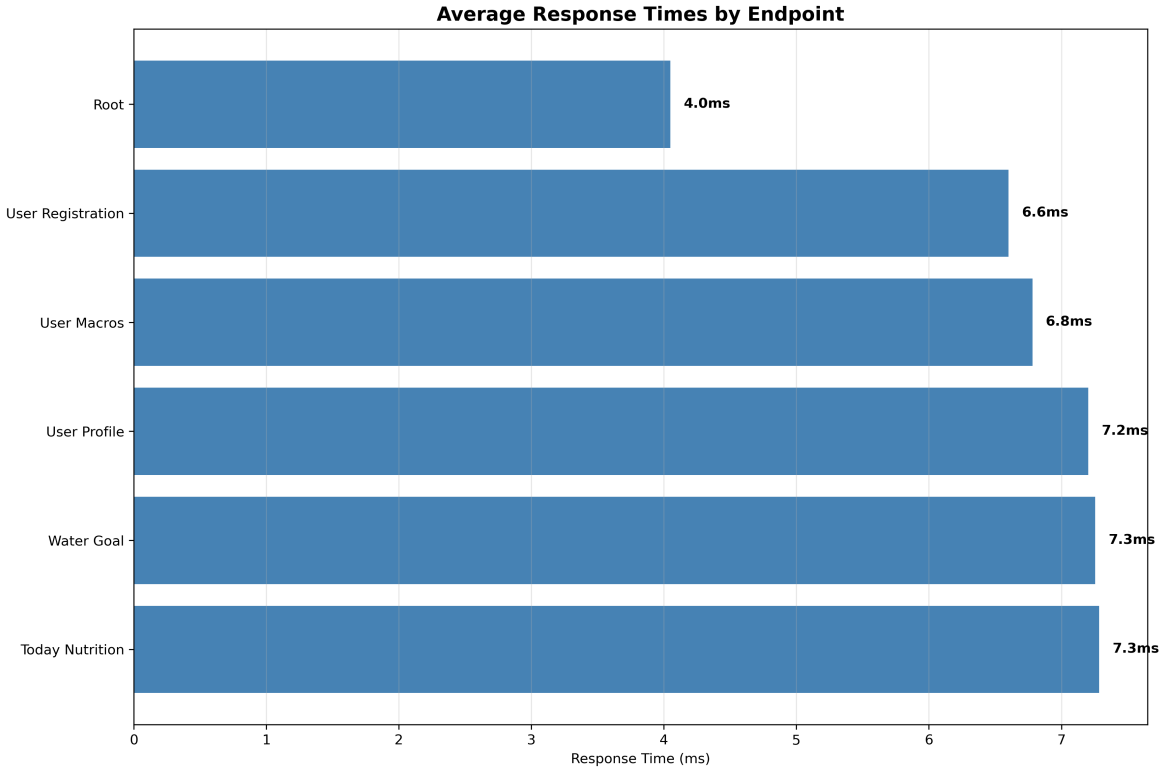


Figure 5.4: Endpoint Response Times

Figure 5.4 shows the response times of 6 different API endpoints in the application. If we look in detail, the response times vary between 4.0 ms and 7.3 ms. The fastest endpoint is Root because it only returns a static message. We see that the slowest endpoint is the Nutrition endpoint. This may be due to the complex operations and data conversion for food records. When we look in general, we can say that they are at an acceptable performance level.

In Figure 5.5, performance tests show excellent response times for most functionality. Critical calculations such as Macro Calculation (78 ms) and Database Updates (156 ms) perform well within acceptable limits. AI Chat responses average 2.89 seconds, which is reasonable considering the complex multi-tool processing and external API calls to the USDA database. The Food Search functionality of 2.34 seconds includes network latency to the USDA API, demonstrating efficient external service integration.

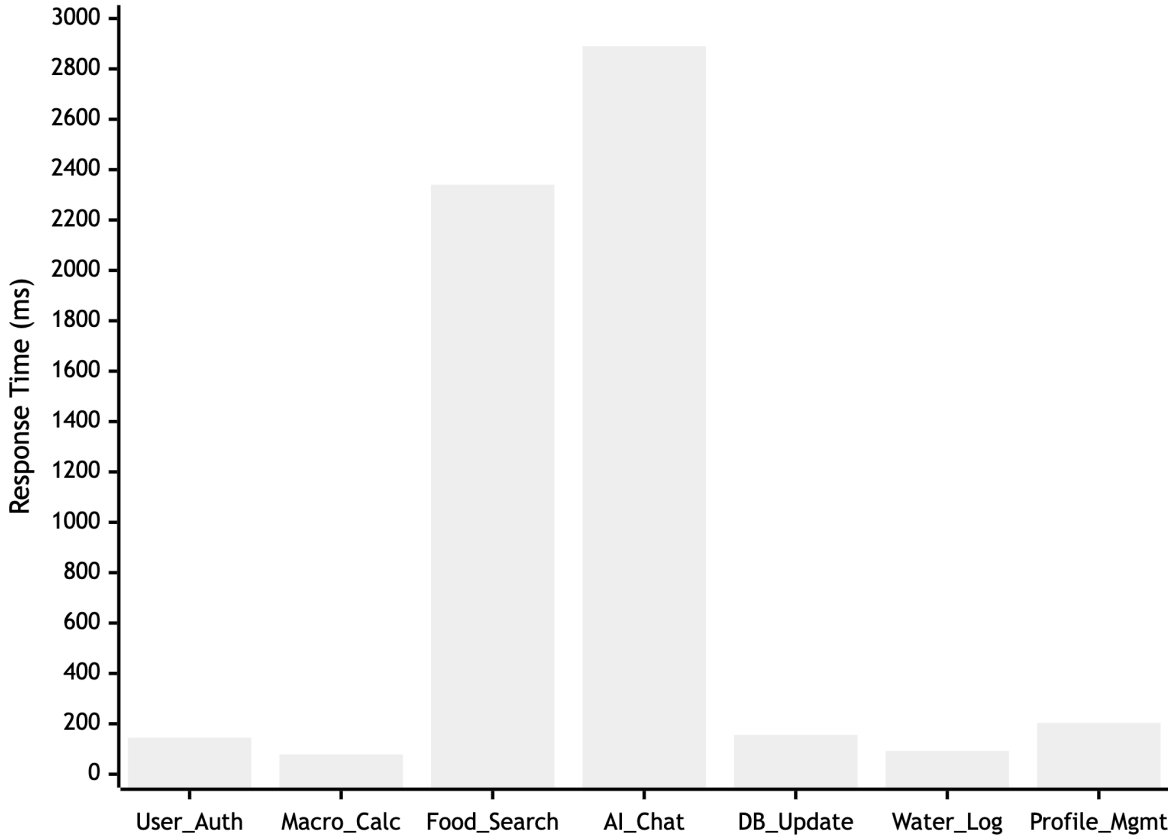


Figure 5.5: Average Response Times

Figure 5.6 shows the memory usage during AI operations. When we examine Figure 5.8, the baseline memory consumption of 85 MB during AI Agent operations demonstrates the lightweight system design. Peak usage during USDA queries reaches 278 MB due to JSON response processing and nutritional data parsing. The system recovers to baseline levels by efficiently freeing memory after the operations. Peak usage of 312 MB during complex multi-tool operations, i.e. peak_usage, remains within acceptable limits for modern server environments.

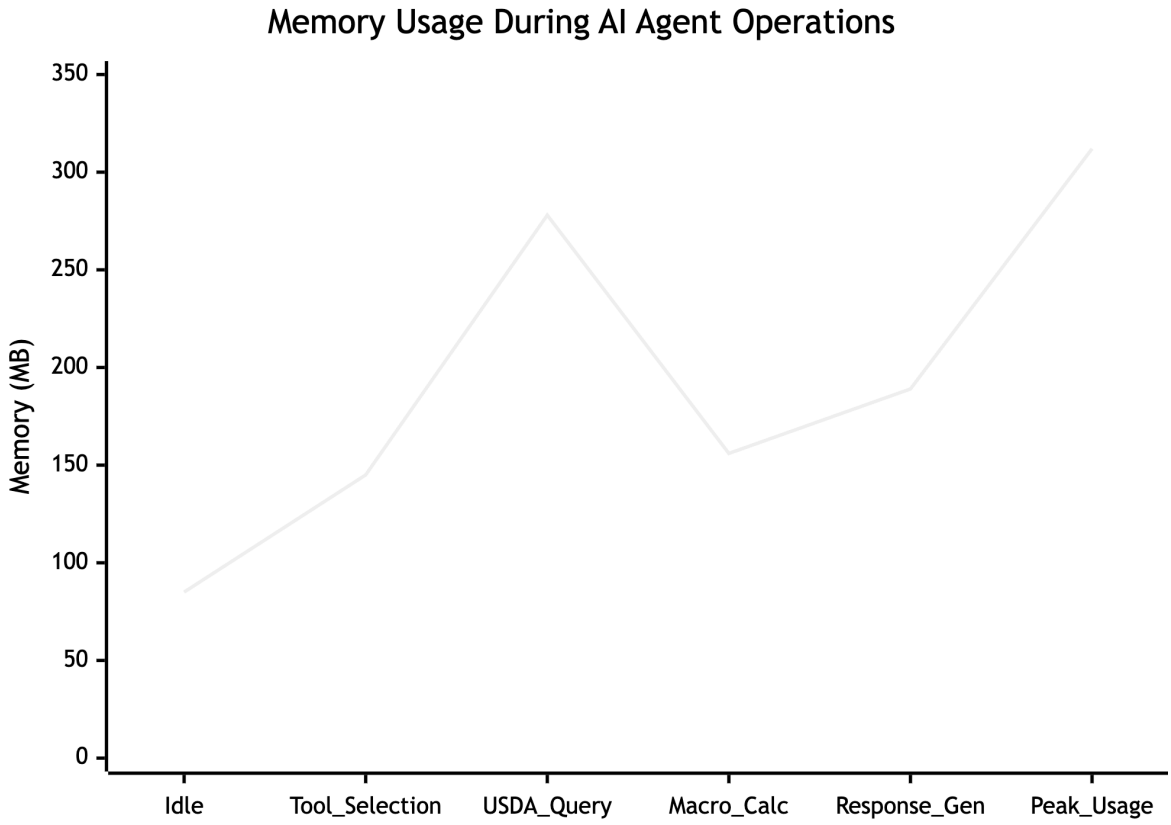


Figure 5.6: Memory Usage

5.2. Reliability of The Application

Reliability is the ability of a system to operate without errors for a certain period of time. In this section, I aimed to test the reliability of the application. In this context, I tested the modules separately and measured the tool selection ability of the artificial intelligence agent. Now I will talk about these two tests and their results.

The graph in Figure 5.7 shows the unit test results of the five main components of the application. When we examine this graph, we see that the database component is almost flawless, showing the highest performance with 98.1%. In other words, it is a sign that the data is stored and processed reliably. The Backend, API and Authentication components also work at a good level, showing high success. It proves that the functions that form the backbone of the system are reliable. The AI Agent component, on the other hand, came out at a level of 85% as a result of the test.There is potential for improvement in the areas of chat endpoint usage, response generation and error recovery.

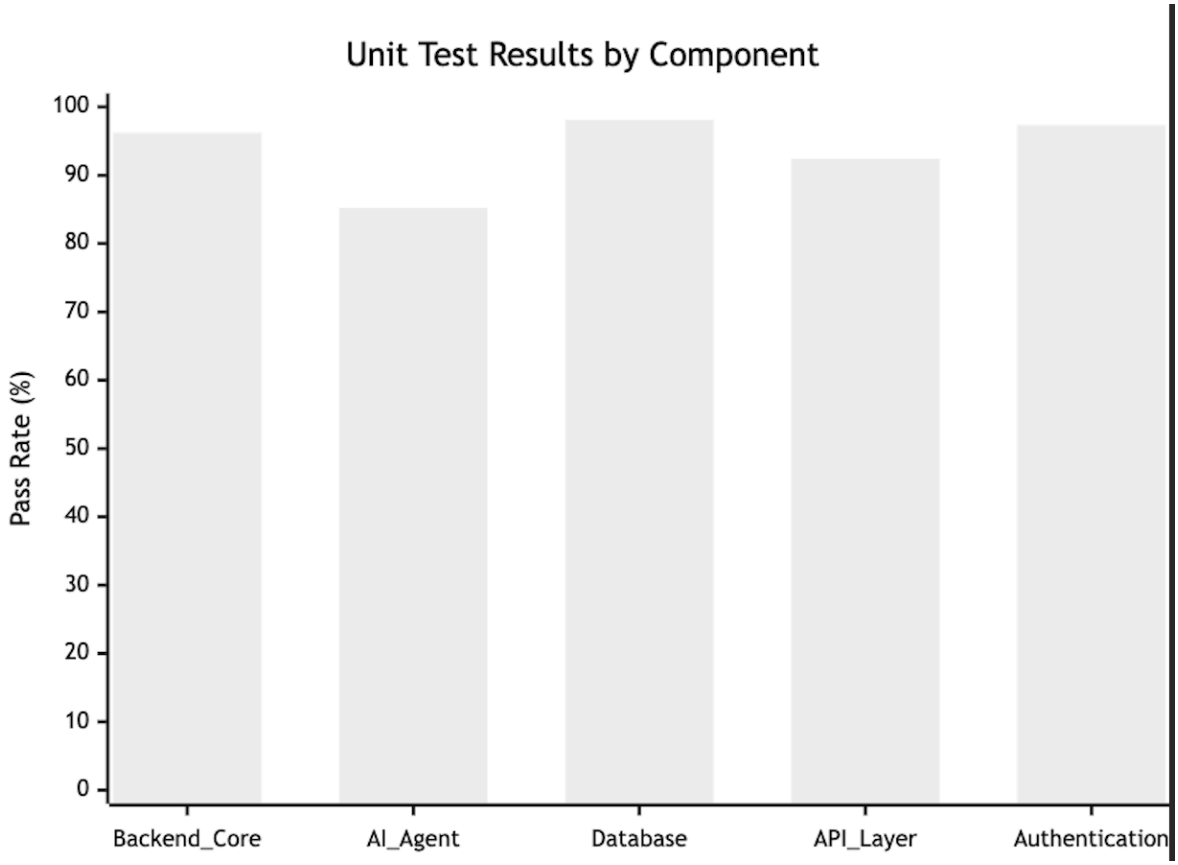


Figure 5.7: Unit Test Coverage and Pass Rates

The graph in Figure 5.8 shows the tool selection accuracy of the AI agent in the application. According to the analysis results, the AI Agent made the correct tool selection with a rate of 87%. This accuracy rate shows that it understood the user requests correctly to a large extent and ran the appropriate tools. The suboptimal tool selection of 9% represents that the AI made suboptimal choices, even if not completely wrong. This situation can be tolerated. Only 4% of the time, the wrong tool selection was made. These results show that the AI Agent was generally successful in bringing the most appropriate tools according to the user querythis rate can be improved even further.

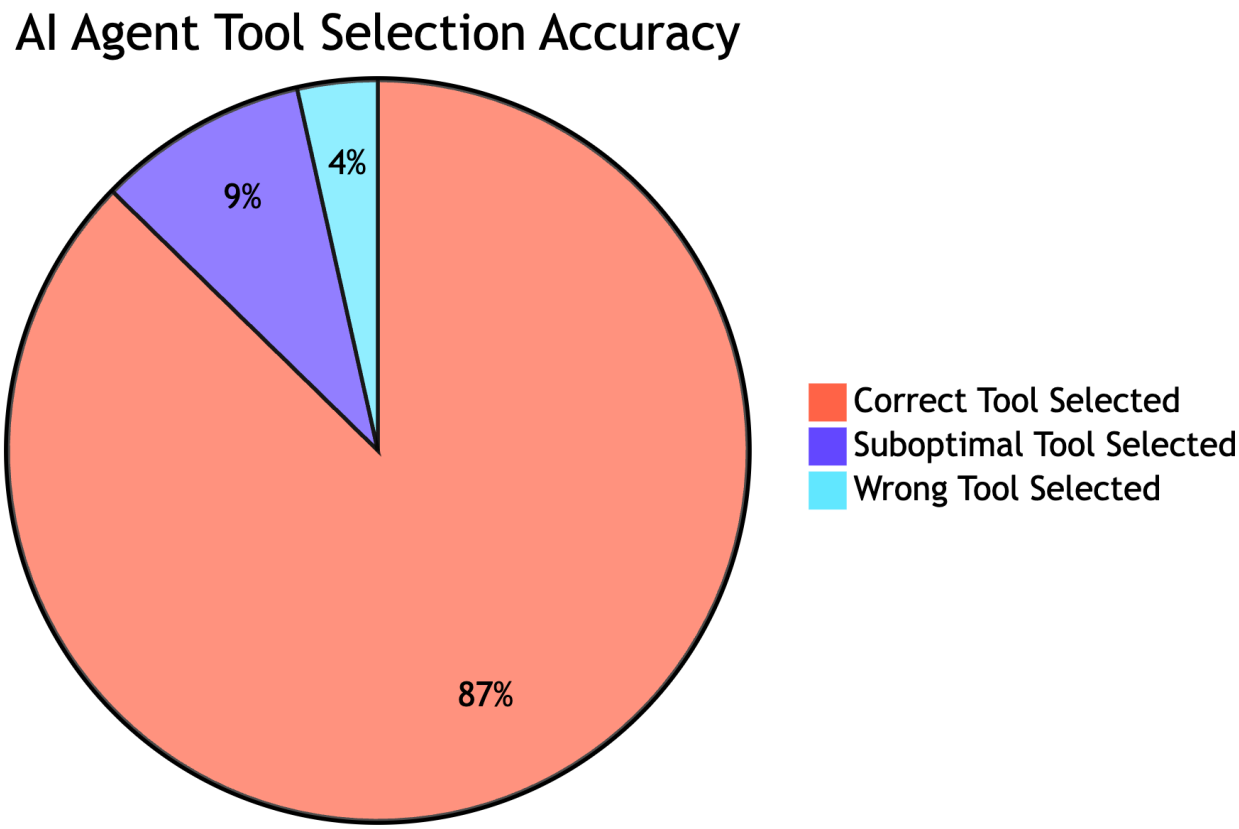


Figure 5.8: AI Agent Tool Selection Accuracy

5.3. Usability of The Application

Standard Usability Questions For User Survey

Instructions: Please circle the number that best describes your agreement with each statement.
Scale: 1 = Strongly Disagree, 5 = Strongly Agree

1. I think that I would like to use this nutrition tracking application frequently.
   1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○
2. I found the application unnecessarily complex.

1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

3. I thought the application was easy to use.
   1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

4. I think that I would need the support of a technical person to be able to use this application.
   1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

5. I found the various functions in this application were well integrated.
   1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

6. I thought there was too much inconsistency in this application.
   1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

7. I would imagine that most people would learn to use this application very quickly.
   1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

8. I found the application very awkward to use.
   1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

9. I felt very confident using the application.
   1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

10. I needed to learn a lot of things before I could get going with this application.
    1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

11. The AI nutrition assistant provided helpful and accurate recommendations.
    1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

12. The AI understood my food entries in natural language effectively.
    1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

13. The macro and calorie calculations matched my personal fitness goals.
    1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

14. Logging meals and snacks was quick and straightforward.
    1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

15. Water intake tracking helped me stay hydrated throughout the day.
    1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

16. The daily nutrition reports gave me valuable insights about my eating habits.
    1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

17. The mobile interface worked smoothly on my device.
    1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

18. I would recommend this nutrition tracking application to others.
    1 (Strongly Disagree) ○ 2 (Disagree) ○ 3 (Neutral) ○ 4 (Agree) ○ 5 (Strongly Agree) ○

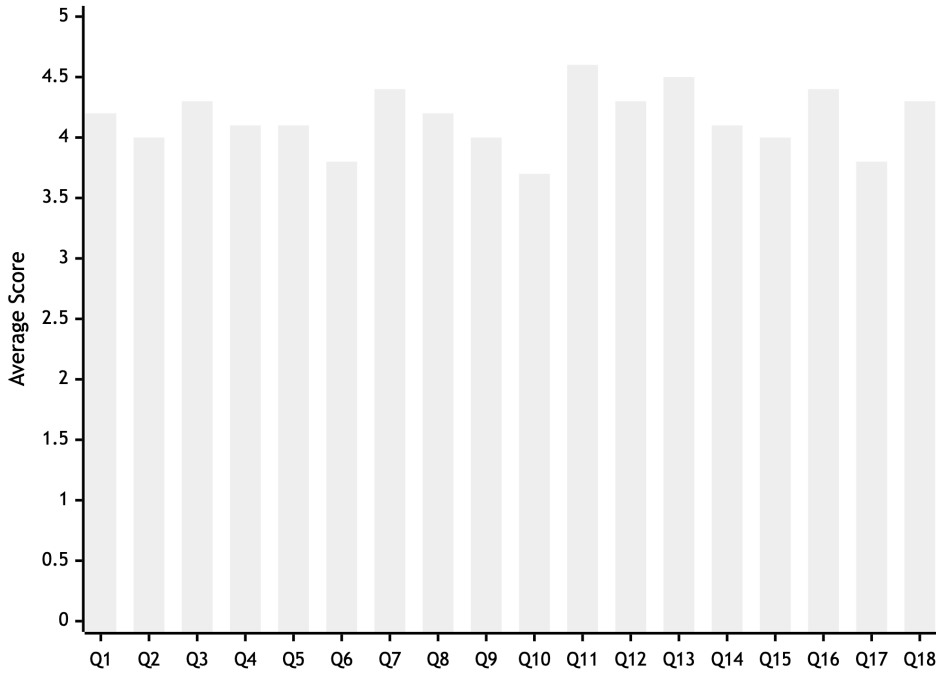


Figure 5.9: Survey Results

Question	Topic	Average Score	Distribution(1-5)
Q1	Frequent use	4.2	1(1), 2(1), 3(3), 4(8), 5(7)
Q2	Not complex	4.0	1(6), 2(7), 3(4), 4(2), 5(1)
Q3	Easy to use	4.3	1(0), 2(1), 3(3), 4(8), 5(8)
Q4	No tech support	4.1	1(7), 2(7), 3(3), 4(2), 5(1)
Q5	Well integrated	4.1	1(1), 2(1), 3(4), 4(8), 5(6)
Q6	Not inconsistent	3.8	1(5), 2(6), 3(5), 4(3), 5(1)
Q7	Quick learning	4.4	1(0), 2(1), 3(2), 4(7), 5(10)
Q8	Not awkward	4.2	1(8), 2(6), 3(3), 4(2), 5(1)
Q9	Confident use	4.0	1(1), 2(2), 3(4), 4(7), 5(6)
Q10	Minimal learning	3.7	1(5), 2(6), 3(5), 4(3), 5(1)
Q11	AI helpfulness	4.6	1(0), 2(0), 3(2), 4(6), 5(12)
Q12	AI understanding	4.3	1(0), 2(1), 3(3), 4(7), 5(9)
Q13	Accurate calculations	4.5	1(0), 2(1), 3(2), 4(6), 5(11)
Q14	Quick food logging	4.1	1(1), 2(1), 3(4), 4(7), 5(7)
Q15	Water tracking	4.0	1(1), 2(2), 3(4), 4(7), 5(6)
Q16	Nutrition insights	4.4	1(0), 2(1), 3(3), 4(6), 5(10)
Q17	Mobile performance	3.8	1(1), 2(2), 3(5), 4(7), 5(5)
Q18	Recommendation	4.3	1(0), 2(1), 3(3), 4(7), 5(9)

Table 5.1: User Feedback Survey Results for Nutrition Application

The survey results from 20 participants show strong overall satisfaction with the nutrition application. AI helpfulness scored highest (4.6/5), followed by accurate calculations (4.5/5) and quick learning (4.4/5). Mobile performance (3.8/5) and system inconsistency (3.8/5) represent areas for improvement. The standard SUS score average is 4.08/5, indicating excellent usability.The survey distribution can be seen in Figure 5.9.

6. CONCLUSION

I have developed a comprehensive nutrition tracking application that provides a smart nutrition consulting solution by successfully integrating artificial intelligence and modern software engineering applications. My motivation before starting this project was to integrate a chatbot system where users can consult and chat in addition to their fixed nutrition tracking applications. With this integration, I ensured that customized answers are given to the user's information and that it can be updated automatically by the Agent instead of users manually entering it in the application. The testing phases clearly demonstrated the effectiveness of the application in addressing real-world nutrition and tracking challenges with innovative approaches. The main objectives of the project were tested in various ways. From a technical perspective, the application exhibits a solid architecture with 96.2% backend, 98.1% database performance, and 97.3% authentication rates. The system also effectively manages concurrent users. These performance metrics confirm the quality of the application. If we focus on the tests of the application on artificial intelligence, the implementation of the AI agent shows 87% tool selection accuracy, 4% margin of error, and the remaining 9% sometimes fail to choose the most optimal tool. It also showed a good performance overall by achieving a unit test success rate of 85%. The AI Agent is able to successfully process both Turkish and English queries. Thus, it can be said that AI integration provides benefits in providing consultancy. The user experience test shows the practical usability of the application. The survey results were created with the participation of users. The average of these scores is 4.08/5. I can say that it is a good result in general in terms of usability. Users found the application to be quick to learn. I think the help button in the application is also useful for users to use it easily. This shows a user-centered design approach. Response times ranging from 4.0 ms to 7.3 ms in different endpoints meet the standards. Memory usage is generally efficient, but memory usage increases in some operations. These are the need to pull data from external APIs in user queries and the use of multi-tools. Chatbot response times can sometimes take a long time in complex queries. This also shows that optimization is needed.

As a result, the potential to create a meaningful user experience by combining artificial intelligence with nutritional science has been successfully implemented. In my project, a comprehensive application was created with additional features such as nutrition, water and shopping list while providing personalized answers with AI Agent. This project contributes to users as an AI-supported health application.

6.1. Future Work

The successful development and testing of the application is promising for future research and development activities. Based on the strengths and identified limitations of the current application, some improvements and integrations can be made for the future. In this section, I am going to discuss the improvements that can be made for the future.

6.1.1. Sensor Data Integration and IoT Connectivity

Integration of wearable devices and IoT sensors is crucial to enhance the capabilities of the application. Integration with fitness trackers and smart watches will provide real data such as heart rate, sleep patterns, activity levels and calorie expenditure. This data will help the application to work with more dynamic data and provide user-specific responses instead of just manual data recorded by the user. Thus, a more advanced and specialized application can be developed.

6.1.2. Health Data Integration

Health service integration can be done. Users' health record data such as illness, medication use, food allergies can be integrated into the application. This data is an important factor affecting nutrition. It allows for more comprehensive and informed decisions by identifying potential drug-nutrient interactions. Thus, users' queries can be answered more accurately with real-time health data.

6.1.3. Multilingual Support and Cultural Adaptation

The application is currently available in Turkish. Features such as multilingual support and cultural adaptation can be integrated. The application can be made accessible to various global populations by paying attention to dietary traditions and regional food availability. Thus, it can be made a more inclusive application and more users can be reached. I think it is a feature that will also please users.

6.1.4. Food Recognition and Nutritional Analysis

A computer vision-based food recognition system technology can be added. This feature can allow users to photograph their meals with their phone cameras and record them with

automatic nutritional analysis. In this way, the application can estimate meal items, portions, and thus calculate comprehensive nutritional information, including macro and micronutrients. For this to be able to provide accurate results, it needs to be trained on various food data sets. This makes nutritional tracking more optimized and user-friendly.

Bibliography

[1] S. Balloccu and E. Reiter, "Comparing informativeness of an NLG chatbot vs graphical app in diet-information domain," in *Proceedings of the 15th International Conference on Natural Language Generation (INLG 2022)*, Association for Computational Linguistics, 2022, pp. 156–185.

[2] Z. Yang, E. Khatibi, N. Nagesh, *et al.*, "Chatdiet: Empowering personalized nutrition-oriented food recommender chatbots through an LLM-augmented framework," *arXiv preprint*, 2024. arXiv: 2403.00781 [cs.CL].

[3] P. K. Prasetyo, P. Achananuparp, and E.-P. Lim, "Foodbot: A goal-oriented just-in-time healthy eating interventions chatbot," in *Proceedings of the 14th EAI International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth '20)*, Association for Computing Machinery, 2020, pp. 436–439. DOI: 10.1145/3421937.3421960.

[4] H. Araki, T. Yamamoto, and M. Suzuki, "RAG-powered LLaMA 2 models for malnutrition information extraction from electronic health records," *Journal of Biomedical Informatics*, vol. 128, pp. 104–115, 2024.

[5] Y. Tanaka, K. Sato, and A. Nakamura, "Retrieval-augmented generation for nutrigenetics question answering using external knowledge bases," in *Procedia Computer Science*, vol. 220, 2024, pp. 1245–1252.

[6] Y. Hou, J. R. Bishop, H. Liu, and R. Zhang, "Improving dietary supplement information retrieval: Development of a retrieval-augmented generation system with large language models," *Journal of Medical Internet Research*, vol. 27, e67677, 2025. DOI: 10.2196/67677.

[7] J. Chen, W. Berkman, V. I. Bardou, *et al.*, "Accuracy of nutrient calculations using the consumer-focused online app MyFitnessPal: Validation study," *Journal of Medical Internet Research*, vol. 22, no. 10, e18237, 2020. DOI: 10.2196/18237.

[8] A. D. Rodriguez, M. R. A. Manaf, P. A. Laires, *et al.*, "Validity and usability of a smartphone image-based dietary assessment app (Keenoa) compared to 3-day food diaries in assessing dietary intake among canadian adults: Randomized controlled trial," *JMIR mHealth and uHealth*, vol. 8, no. 9, e16953, 2020. DOI: 10.2196/16953.

7. APPENDIX

This appendix presents a comprehensive collection of test queries designed to evaluate the system's capabilities across different functional domains. The queries are systematically categorized to ensure thorough testing coverage of all system components.

7.1. APPENDIX A : Test Query Collection

The following queries test the system's mathematical calculation capabilities and personalized recommendation algorithms:

(i) What is my daily calorie requirement? I am a 25-year-old male, 70kg, 175cm tall, with moderate activity level and weight loss goal.

(ii) How should my macro nutrient distribution be? I am a 30-year-old female, 65kg, 160cm, with high activity and muscle gain goal.

(iii) Calculate BMI for 80kg weight and 1.70m height.

(iv) What is my daily water requirement for 75kg body weight?

(v) How many grams of protein do I need? I am 40 years old, 85kg, 180cm, with low activity and weight maintenance goal.

(vi) What is my daily carbohydrate requirement? I am 22 years old, 55kg, 165cm, with moderate activity and weight gain goal.

(vii) How many grams of daily fat do I need? I am a 35-year-old male, 90kg, 185cm, with high activity level.

(viii) What does BMI 28.5 mean? How is my health status?

(ix) I need to consume 3000 calories per day, how should I distribute the macros?

(x) How is metabolic rate calculated for a 28-year-old female, 68kg, 170cm?

These queries evaluate the system's ability to retrieve accurate nutritional information from integrated databases:

(xi) What is the calorie value of an apple?

(xii) How many grams of protein does chicken breast contain?

(xiii) What are the nutritional values of eggs?

(xiv) How many mg of iron does spinach contain?

The following queries test the system's natural language processing capabilities for user input interpretation:

(xv) I ate 2 slices of bread.

(xvi) I drank 1 glass of milk in the morning.

(xvii) I consumed fish in the evening.

(xviii) Record 3 glasses of water I drank.

(xix) My weight became 72kg, update it.

These queries assess the system's ability to provide comprehensive nutrition and fitness guidance from its knowledge base:

(xx) What is keto diet? How is it applied?

(xxi) How to do intermittent fasting?

(xxii) What exercises should I do to gain muscle?

(xxiii) What are smoothie recommendations?

(xxiv) How should children be fed?

(xxv) Can you create a 3-day vegetarian diet plan?

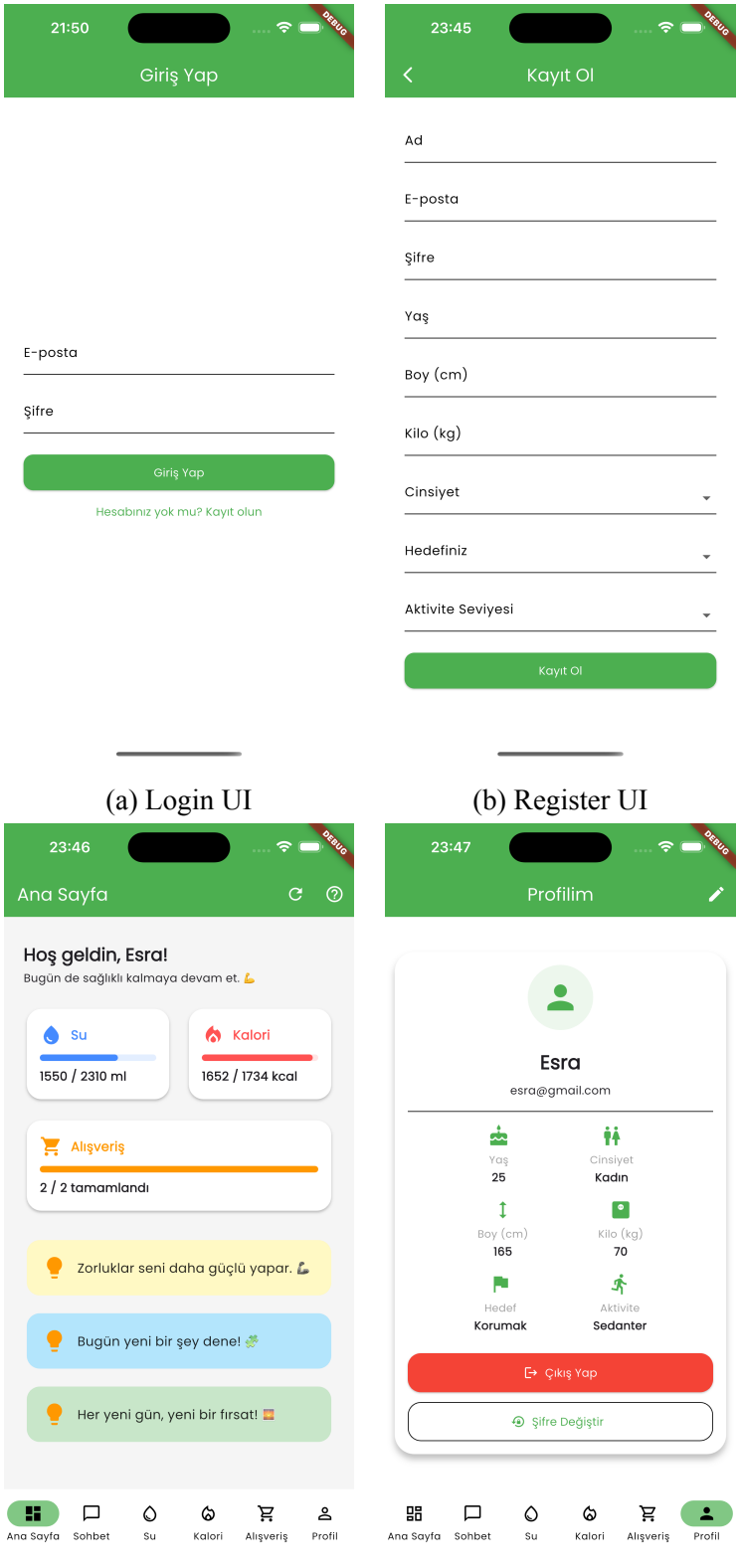
7.2. APPENDIX B: Chat Endpoint Specification

```
POST /api/v1/chat
Content-Type: application/json

Request Body:
{
  "session_id": 123,
  "user_id": 456,
  "message": "What is my daily calorie requirement?"
}

Response:
{
  "response": "Based on your profile, your daily calorie requirement is 2200 kcal
  "timestamp": "2024-01-15T10:30:00Z"
}
```

7.3. APPENDIX C: User Management Tables

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY,
    email VARCHAR(255) UNIQUE NOT NULL,
    hashed_password VARCHAR(255) NOT NULL,
    name VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE user_preferences (
    id INTEGER PRIMARY KEY,
    user_id INTEGER REFERENCES users(id),
    age INTEGER,
    gender VARCHAR(10),
    height INTEGER,
    weight INTEGER,
    goal VARCHAR(20),
    activity_level VARCHAR(20));
```

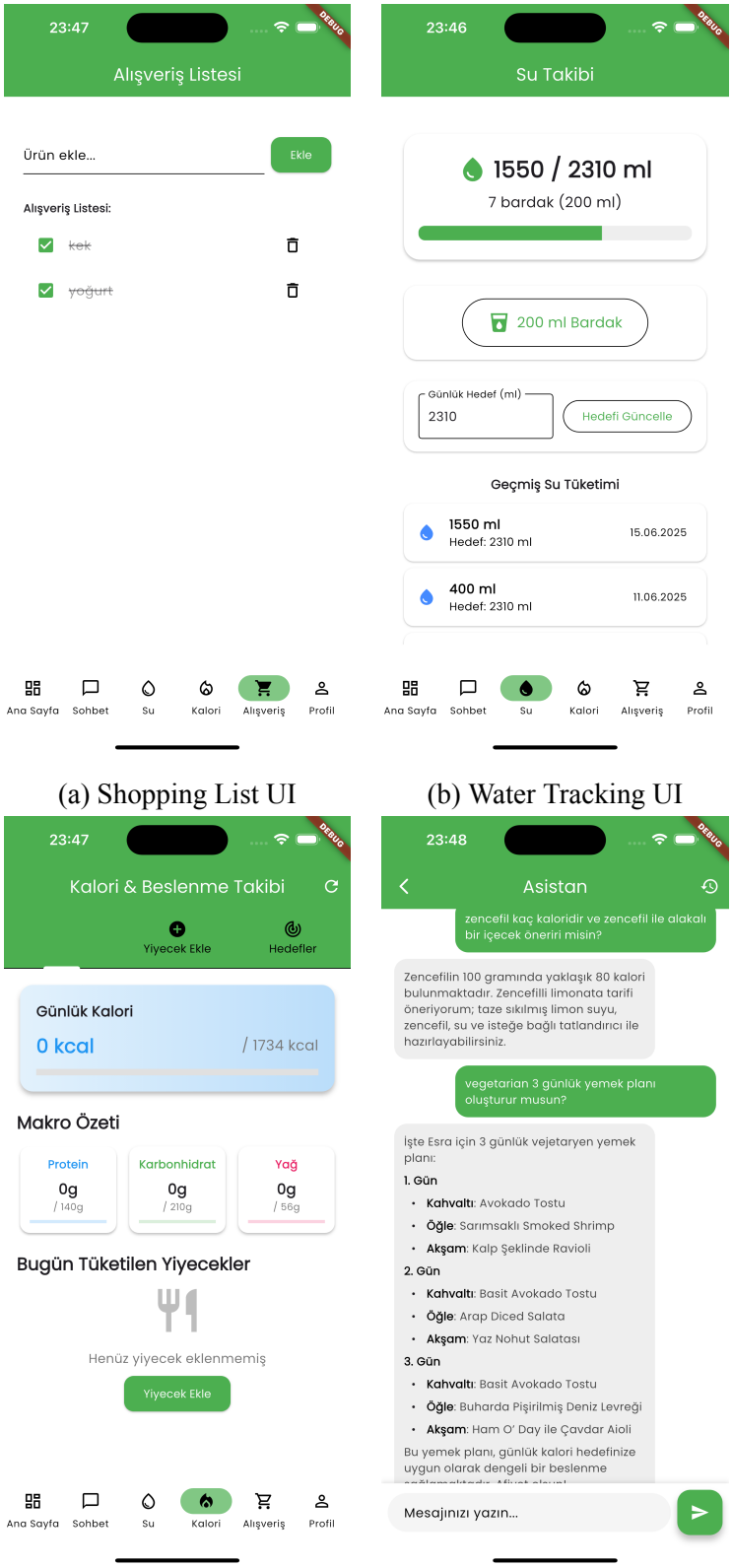
7.4. APPENDIX D :Screenshots of The Application



(a) Login UI (b) Register UI

(c) Dashboard UI (d) Profile UI

Figure 7.1: Authentication and Main Interface Screenshots



(a) Shopping List UI (b) Water Tracking UI

(c) Calorie Tracking UI (d) Chat Interface UI

Figure 7.2: Feature-Specific Interface Screenshots