** **

**T.C.**

**MARMARA UNIVERSITY**

**FACULTY of ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

**CSE3044 Software Engineering Homework #4**

**Project Name :FitEats**

Group Members

Fatih Genç 150119905

Umut Berke Pezük 150119712

Alperen Koruyucu 150119734

Hakan Adaklı 150116069

Mevlüt Eren Topal 150117025

Ceyhun Erdönmez 150120851

Ahmet Bozbay 150119861

Supervised by

Mustafa Borahan Tümer

**Table of Contents**

[**1. Introduction** 4](#_Toc134637012)

[**1.1. Purpose** 5](#_Toc134637013)

[**1.2. Statement of scope** 5](#_Toc134637014)

[**1.4. Major constraints** 5](#_Toc134637015)

[**1.5. Definitions** 6](#_Toc134637016)

[**1.6. Acronyms and Abbreviations** 6](#_Toc134637017)

[**2. Design Consideration** 6](#_Toc134637019)

[**2.1.** **Design Assumptions and Dependencies** 6](#_Toc134637020)

[**2.2.** **General Constraints** 7](#_Toc134637021)

[**2.3.** **System Environment** 7](#_Toc134637022)

[**2.4.** **Development Methods** 7](#_Toc134637023)

[**3. Architectural and component-level design** 7](#_Toc134637024)

[**3.1. System Structure** 7](#_Toc134637025)

[**3.1.1. Architecture diagram** 9](#_Toc134637026)

[**3.2. Description for Component** 10](#_Toc134637028)

[**3.2.1 Description of the Login Screen** 10](#_Toc134637029)

[**3.2.2 Description of the SignUp Screen** 11](#_Toc134637030)

[**3.2.3 Description of the Navigation Bar component** 11](#_Toc134637031)

[**3.2.4 Description of the User Profile Screen** 12](#_Toc134637032)

[**3.2.5 Description of the Edit Profile Screen** 13](#_Toc134637033)

[**3.2.6 Description of the LogOut Component** 13](#_Toc134637034)

[**3.2.7 Description of the Recipe Component** 14](#_Toc134637035)

[**3.2.8 Description of the BMI Calculator Component** 14](#_Toc134637036)

[**3.2.9 Description of the RecipeFromCategories Component** 15](#_Toc134637037)

[**3.2.10 Description of the Recipe Screen** 15](#_Toc134637038)

[**3.2.11 Description of the calculateCalorieDaily Component** 16](#_Toc134637039)

[**3.3. Dynamic Behavior for Component n:** 17](#_Toc134637040)

[**3.3.1. Interaction Diagrams** 17](#_Toc134637041)

[**4. Restrictions, limitations, and constraints** 22](#_Toc134637042)

[**5. Conclusion** 23](#_Toc134637043)

# **1. Introduction**

Maintaining a healthy diet can be challenging in today's busy world. With long work hours, busy schedules, and endless to-do lists, it can be challenging to find the time and energy to plan and prepare nutritious meals. Fiteats is a web application designed to address these challenges. By providing personalized meal recommendations based on users' individual preferences, Fiteats makes it easy to stay on track and achieve dietary goals. Our platform offers a range of nutritious and delicious meal options that are simple to prepare and fit seamlessly into users' busy lifestyles. With Fiteats, users can take control of their diets

## **1.1. Purpose**

Fiteats aims to simplify healthy eating by providing users with personalized meal recommendations that meet their daily calorie needs. Our platform is designed to help people achieve their desired outcomes, whether they're trying to lose weight, build muscle, or eat more healthily. Fiteats provides users with the tools and resources they need to stay on track and reach their goals. Our ultimate purpose is to empower users to take control of their diets and achieve optimal health and well-being.

## **1.2. Statement of scope**

Fiteats is a web application that provides personalized meal recommendations to users based on their daily calorie needs and individual preferences. Our platform allows users to create a profile and receive a list of recommended meals for each day based on their preferences and goals. Fiteats includes features for storing user information and pulling recipes and ingredients from external datasets. The application determines the maximum and minimum daily calorie intake for each user based on their specific goals and needs, such as building muscle, losing weight, or gaining weight. Fiteats then divides this by the number of meals per day to recommend suitable calorie limits for each meal. The software recommends meals within the recommended calorie limits for each meal based on the user's preferences and dietary restrictions. Users can also search for recipes based on specific ingredients. These features are designed to simplify the meal planning process, provide users with quick and easy access to healthy meal options, and empower them to take control of their diets and make informed choices about their health.

**1.3. Software context**

Fiteats is a software development project that involves a variety of design and development tasks. Throughout the development process, our team will be creating a range of diagrams and models to guide the project. These may include UML diagrams that outline the system's structure and components, as well as sequence diagrams that depict the flow of actions between different parts of the system. By using these diagrams and models, we can ensure that the software is well-organized, efficient, and easy to maintain. This sofware will result in a better user experience for Fiteats users.

## **1.4. Major constraints**

Fiteats is developed using the Python programming language and the Django web framework. The software is limited to web-based access, and users must have an internet connection to use the platform. Fiteats relies on external datasets to provide users with recipe and ingredient information. As a result, the accuracy and availability of this information may affect the quality of meal recommendations provided by the application.

## **1.5. Definitions**

**User:** An individual who wants to keep track of their training program, diet, and body measurements using Fiteats.

**Calorie:** A unit of energy used to measure the amount of energy in food.

**BMI:** Body Mass Index is a value used to determine healthy body weight calculations.

**UML:** Unified Modeling Language is a standardized language used to design software systems.

**Sequence Diagram:** A diagram that depicts the interactions between system components in a particular order.

**Python:** A popular programming language that is used for the development of the Fiteats software.

**Django:** A high-level Python web framework that enables rapid development of secure and maintainable websites.

**HTML:** Hypertext Markup Language, a standard markup language used to create web pages.

## **1.6. Acronyms and Abbreviations**

**RSD:** Requirements Specification Document

**UCD:** Use Case Diagram

**UI:** User Interface

**Python:** PY

**Django:** DJ

**HTML:** HTM

**BMI:** Body Mass Index

# **2. Design Consideration**

## **2.1. Design Assumptions and Dependencies**

**Assumptions:**

1. To facilitate the application, it is assumed that the application will not work with a large number of users.

2. All group members will do their responsibility.

**Dependencies:**

1. Skill: The application that the user has enough skill to use on the phone and computer.

2. Operating System: Users using chromium-based browsers will be able to use the site..

3. Internet connection is needed.

## **2.2. General Constraints**

Our mobile app can typically have access to various types of personal/sensitive data (such as weight and age) provided by users. Personal data can be stolen, which should be protected and this can cause unexpected privacy impacts.

We don't know exactly the size of the user data we will store, so we have no idea yet if the free storage spaces we can use will be enough.

Since our web application is not finished, we cannot say exactly how much size is needed

## **2.3. System Environment**

The mobile application has been developed on Windows 10 and Windows11 operating system.

We used VSCODE as an IDE for developing the web app.

We also used some Bootstrap packages in our project.

## **2.4. Development Methods**

After deciding what kind of application we want to develop. We chose the agile methodology that adapts easily to changes and can get results faster. We created a WhatsApp group to stay in touch. During the project, we make voice calls on Discord on average 2 days a week. If we encounter a problem, we explain our problem and produce solutions through these communication channels. We divided the project into two groups as frontend and backend and continued our work like this. We tried to build the app as soon as possible.

# **3. Architectural and component-level design**

## **3.1. System Structure**

The system consists of two main components: the frontend and the backend. The frontend is responsible for providing a user interface that allows users to interact with the system. The backend is responsible for processing user requests, calculating personalized meal recommendations, and providing the necessary data to the frontend.

The backend component consists of several sub-components that work together to provide the required functionality. The core of the backend is a set of algorithms that calculate the user's daily calorie needs and recommend meals based on their preferences and goals. The algorithm will also determine the maximum and minimum daily calorie intake for the user based on their goals and needs. Additionally, the backend component includes a database that stores user information, such as preferences and dietary restrictions.

To generate meal recommendations, the backend component will pull recipes and ingredients from external data sources. The external data sources could be nutrition databases, recipe databases, or other relevant sources. The database component stores the data pulled from the external data sources, as well as user information.

Finally, the frontend component provides a user interface that allows users to create a profile, search for recipes, and view recommended meals. The frontend component interacts with the backend component through APIs and web services.

Overall, this architecture is designed to provide users with a simple, streamlined way to manage their diet and make informed choices about their health.

### **3.1.1. Architecture diagram**

## 

## **3.2. Description for Components**

### **3.2.1 Description of the Login Screen**

|  |  |
| --- | --- |
| Identification | Login Screen |
| Type | Class/Form |
| Purpose | The login screen ensures that only users can access the system |
| Subordinates | * Main Menu Screen * Sign Up Screen |
| Dependencies | * Main Menu Screen * User Profile Screen |
| Interfaces | The links are contained in the center of the screen. |
| Resources | SQLite: enables the user to sign into app |
| Processing | The only type of processing required is inputting information into the text boxes and navigating to other screens using buttons in the half of the screen. |
| Data | The data for the Login Screen is username and password entered by the user. |

### **3.2.2 Description of the SignUp Screen**

|  |  |
| --- | --- |
| Identification | Sign Up Screen |
| Type | Class/Form |
| Purpose | Sign Up screen allows new users to sign up to application |
| Subordinates | * Login Screen |
| Dependencies | * Login Screen |
| Interfaces | The link is contained at the center of the screen. |
| Resources | SQLite: When the user enters required data to create a new account authentication system creates a new user and stores the data in the database. |
| Processing | The user will enter the required information in the text boxes on the screen and press the login button. |
| Data | The data supplied by the user are name, birthdate, email, password, height, weight and gender. After these data is taken from the user, SQLite keeps these in the database. |

### **3.2.3 Description of the Navigation Bar component**

|  |  |
| --- | --- |
| Identification | Navigation Bar |
| Type | Class |
| Purpose | The Navigation Bar is a navigator of the application. |
| Subordinates | All of the pages |
| Dependencies | All of the pages |
| Interfaces | The buttons are contained at the top of the screen. |
| Resources | None |
| Processing | Thanks to the navigation bar, the user will be able to switch between the pages with the page keys in the navigation bar. |
| Data | No data for this screen. |

### **3.2.4 Description of the User Profile Screen**

|  |  |
| --- | --- |
| Identification | User Profile Screen |
| Type | Class |
| Purpose | This screen enables the user to see user information and edit their data. |
| Subordinates | * Edit Profile Screen * Logout Screen |
| Dependencies | * Main Menu Screen * Edit Profile Screen |
| Interfaces | Username is at the right top of the screen and user information. The buttons for editing screens are at the end of the screen. |
| Resources | SQLite: Required data for the user is taken from the database and if the user updates their data it is stored in the database system. |
| Processing | Users can only select one of the buttons at the end of the screen. Users can edit their profile data. |
| Data | All the following data are stored in the SQLite.   * Name * Email * Birthdate * Gender * Height * Weight |

### **3.2.5 Description of the Edit Profile Screen**

|  |  |
| --- | --- |
| Identification | Edit Profile Screen |
| Type | Class / Form |
| Purpose | The purpose of this screen is to update the height and weight of the user. |
| Subordinates | * User Profile Screen |
| Dependencies | * User Profile Screen |
| Interfaces | There are text fields at the center of the screen where the user can enter his height and weight. On the screen we have one button for saving the new user data. |
| Resources | SQLite: When a user updates their height and weight our database is updated. |
| Processing | Users must enter their height and weight and click the save button. After that our database handles the rest of the process. |
| Data | Height and weight are a must here. |

### **3.2.6 Description of the LogOut Component**

|  |  |
| --- | --- |
| Identification | LogOut |
| Type | Class |
| Purpose | The purpose of this component enables the logout |
| Subordinates | * Login Screen |
| Dependencies | * Navigation Bar |
| Interfaces | There is no interface. Just a button for logging out right top of the page. |
| Resources | No resource is required. |
| Processing | This component allows the user to log out |
| Data | No data |

### **3.2.7 Description of the Recipe Component**

|  |  |
| --- | --- |
| Identification | Recipe |
| Type | Class |
| Purpose | This component enables the user to get some recommendations about what to eat. |
| Subordinates | * Login Screen |
| Dependencies | * Main Menu Screen * User Profile Screen |
| Interfaces | Center of the page we have a search bar. |
| Resources | No resource is required. |
| Processing | The user can enter their ingredients in a search bar. After that our application will make suggestions based on what is entered. |
| Data | No data |

### **3.2.8 Description of the BMI Calculator Component**

|  |  |
| --- | --- |
| Identification | BMI Calculator |
| Type | Class / Form |
| Purpose | This component helps the user to calculate the body mass index. |
| Subordinates | This component contains links to the following screen:   * Profile page |
| Dependencies | The following screens links to this component:   * Profile page |
| Interfaces | If the user creates an account, we can get their information and make a BMI information. And we can show it in user info section on profile page |
| Resources | No resource is required. All data is provided by the user. |
| Processing | **BMI** = weight(KG) / (height(CM)^2 / 10000)  **idealWeightDownLimit** = height(CM)^2 \* (18.5 / 10000)  **idealWeightUpperLimit** = height(CM)^2 \* (24.9 / 10000) |
| Data | Body weight and height values are provided by the user. |

### **3.2.9 Description of the RecipeFromCategories Component**

|  |  |
| --- | --- |
| Identification | RecipeFromCategories |
| Type | Class |
| Purpose | This components allows to find recipes from choosing categorie |
| Subordinates | This component contains links to the following screen:   * Main menu |
| Dependencies | The following screens links to this component:   * Main Menu Screen |
| Interfaces | There is multiple selection form for this |
| Resources | No resource is required. |
| Processing | If user wants to find some recipes ,user can search according to the categories |
| Data | Our recipe database is includes categories |

|  |  |
| --- | --- |
| Identification | Recipe Screen |
| Type | Card |
| Purpose | Recipe screen show the recipes using card component |
| Subordinates | This screen contains links to the following screen:   * Main screen * Profile screen * Ingredients screen |
| Dependencies | The following screen links to this screen   * Main screen * Profile screen |
| Interfaces | The link will show to user from meal cards |
| Resources | No resources need,user should decide the meal only |
| Processing | İf user click the meal ,the recipe screen contains more detailed informations about that meal. |
| Data | Our recipe database is includes detailed recipes. |

### **3.2.10 Description of the Recipe Screen**

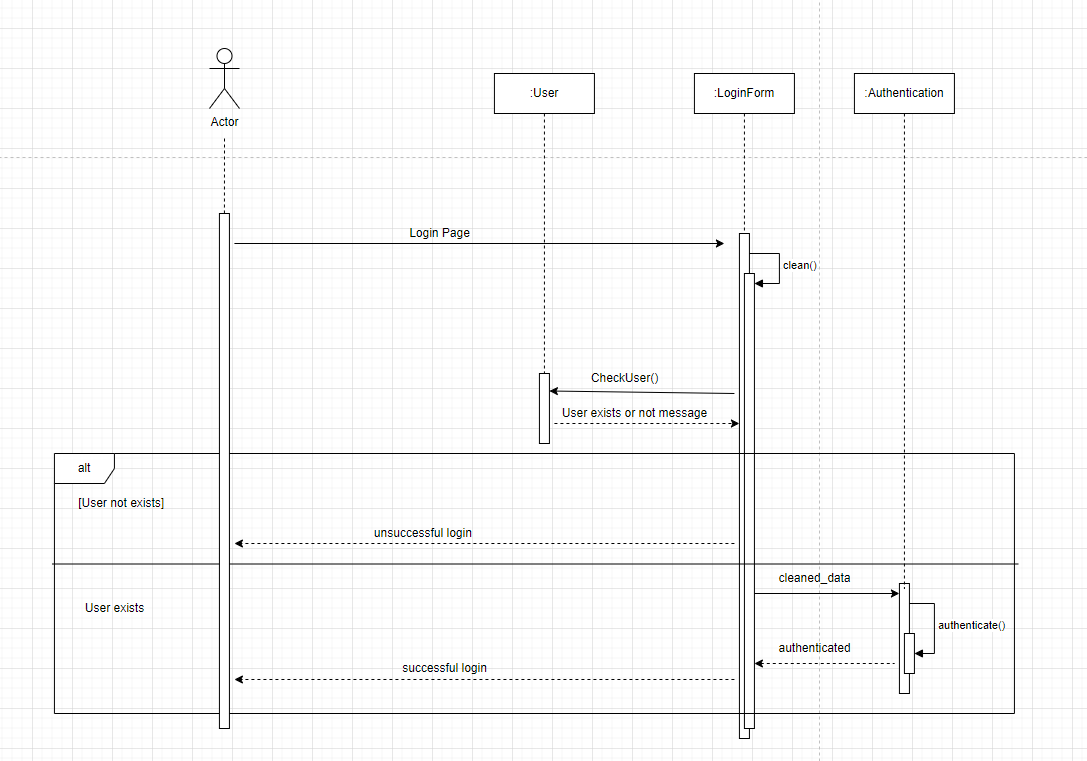
### **3.2.11 Description of the calculateCalorieDaily Component**

|  |  |
| --- | --- |
| Identification | calculateCalorieDaily |
| Type | Class |
| Purpose | This component guess to user calori daily. |
| Subordinates | This component contains links to the following screen:   * Profile page |
| Dependencies | The following screens links to this component:   * Profile page |
| Interfaces | If the user creates an account, we can get their information and make to guess daily calorie information. And we can show it in user info section on profile page |
| Resources | No resource is required. All data is provided by the user. |
| Processing | User can choose activity |
| Data | Body weight and height values are provided by the user. |

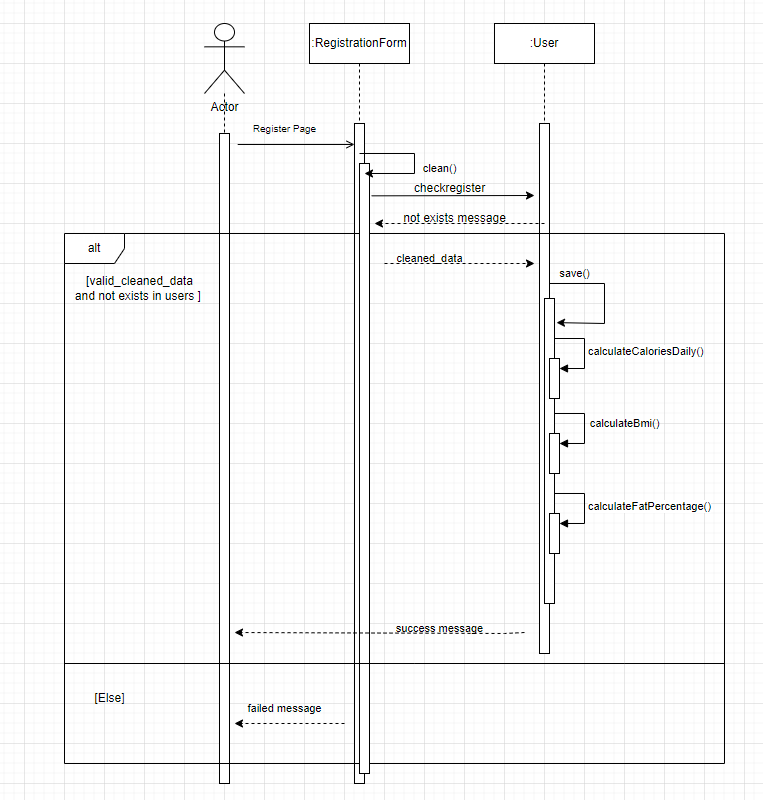
# **3.3. Dynamic Behavior for Component n:**

# **3.3.1. Interaction Diagrams**

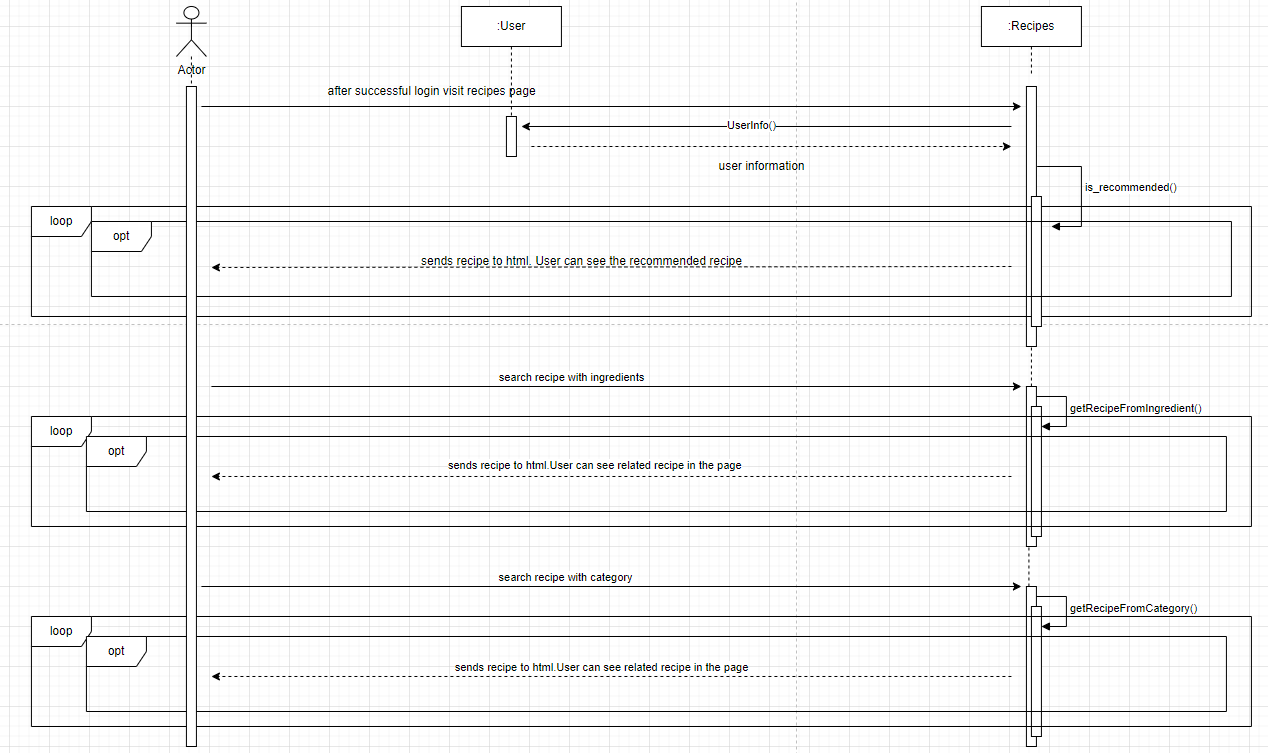
**Sequence Diagram for Login Page Component:**

****

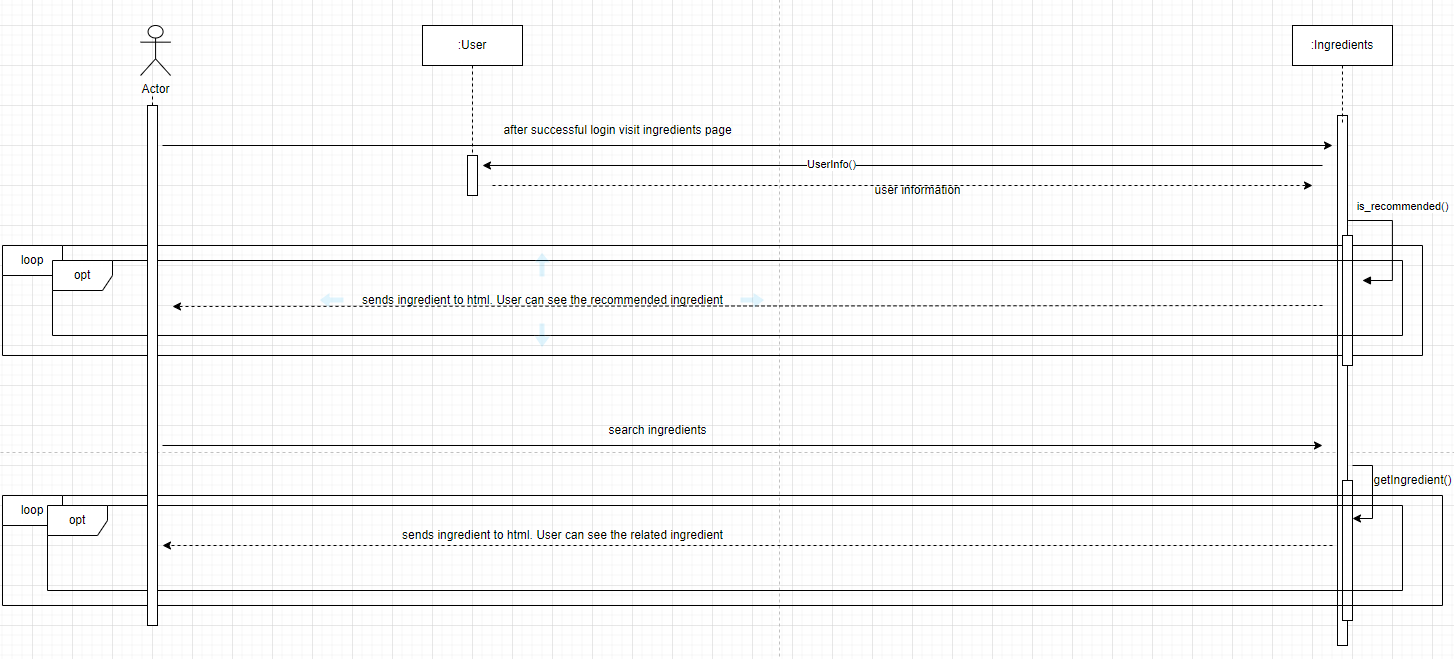
**Sequence Diagram for Register Page Component:**

****

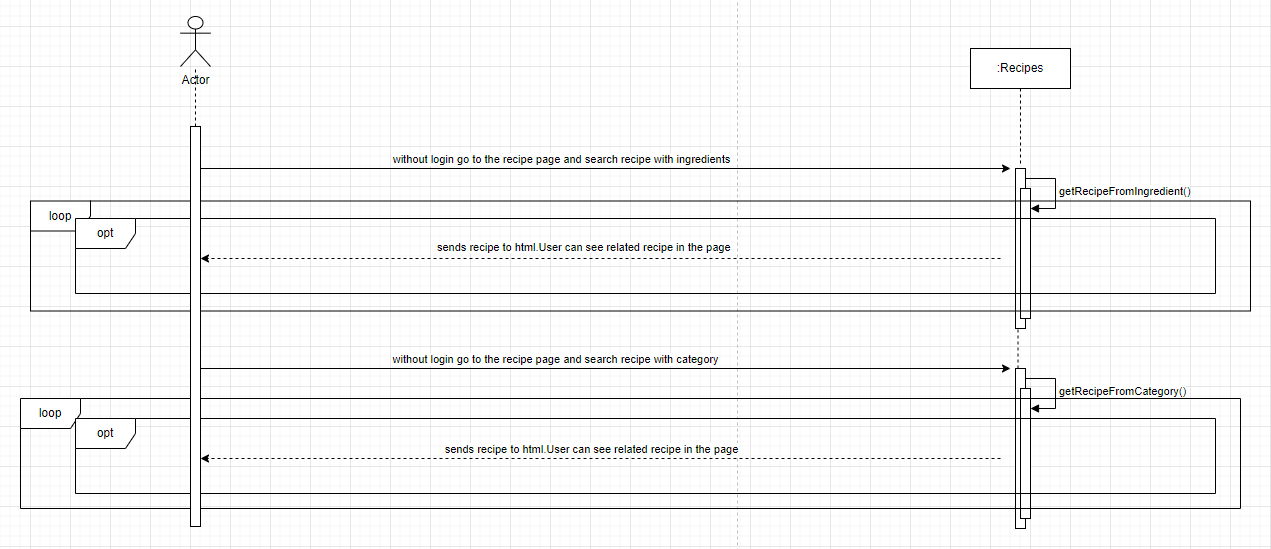
**Sequence Diagram for Recipes Page After Login Component:**

****

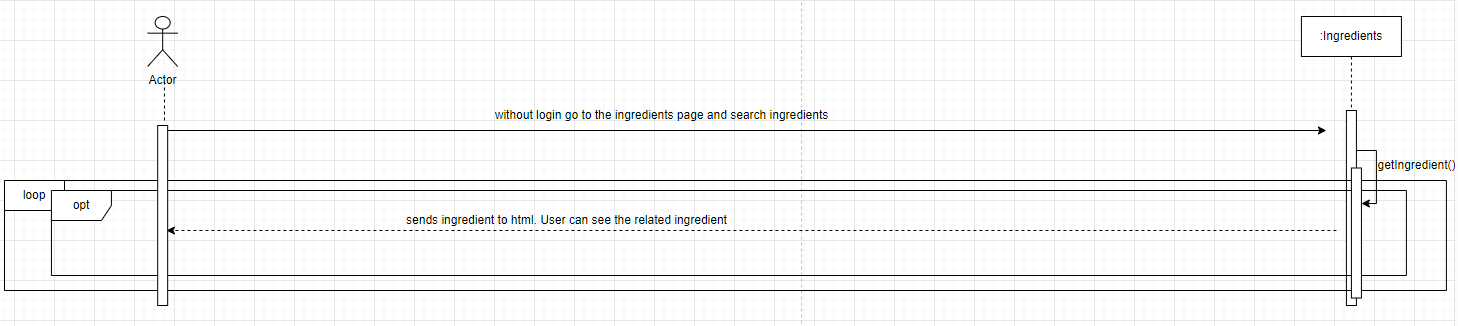
**Sequence Diagram for Ingredients Page After Login Component:**

****

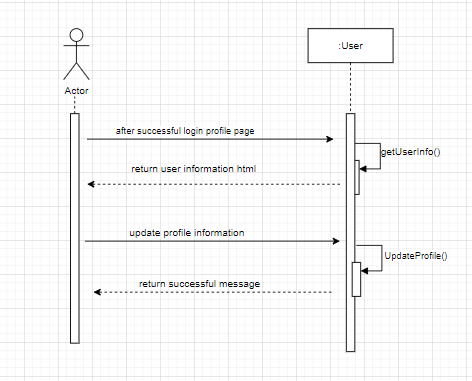
**Sequence Diagram for Recipes Page Without Login Component:**

****

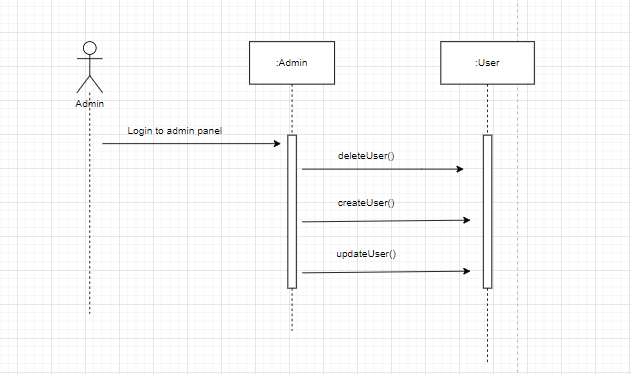
**Sequence Diagram for Ingredients Page Without Login Component:**

****

**Sequence Diagram for Profile Component:**

****

**Sequence Diagram for Admin Component:**

****

# **4. Restrictions, limitations, and constraints**

**Data Privacy and Security:** Since the application will store users' sensitive data such as weight and age, we need to ensure that data privacy and security are maintained. We must protect users' personal data from unauthorized access or disclosure.

**Storage Constraints:** We do not know the size of the user data we will store, so we cannot say exactly how much storage space we will need. We need to ensure that we have enough storage space to store all user data.

**Development Time:** Our team has a limited amount of time to complete the project. We need to ensure that we complete the development process on time and deliver the application on time.

**Hosting Costs:** Domain, hosting and server maintenance costs will occur when the project is desired to be implemented in the future. Increasing the bandwidth when needed can create problems when it is necessary to meet these costs.

**Internet Connection:** Internet connectivity is a major constraint for the application as it will rely on internet connectivity to function. The application may not function properly or may be inaccessible without a stable internet connection.

**Technical Limitations:** The mobile app must work on a wide range of devices and operating systems, which may limit the use of certain features or technologies.Also, the web app must be optimized for different screen sizes and resolutions, which can pose technical challenges during development.

# **5. Conclusion**

# During the project process, we developed a user-friendly and functional recipe platform. This platform has a sustainable database enriched with both visual and written recipes, providing users with an enjoyable experience. In addition, with the feature to search for recipes based on ingredients, users can easily create creative meals with what they already have.

# The success of our project stems from its dynamic structure, which can be customized and continuously updated according to users' needs. This platform will facilitate cooking and help users save both time and energy.

# In future work, we aim to further develop our platform and add new features. In particular, social media integration will encourage community participation by allowing users to share their recipes. Additionally, with features such as smart suggestions and personalized menu options, we will enhance the user experience.

# In conclusion, the recipe platform we developed takes users' cooking experiences to a new level, making the process more enjoyable, easy, and efficient. Through this project, users can improve their cooking skills and contribute to a global gastronomy community by exploring the flavors of different cultures.

# **Distribution of Tasks**

|  |  |
| --- | --- |
| Fatih Genç | Dynamic Behavior for Component n, Overall design of DSD and General Control of DSD |
| Umut Berke Pezük | Conclusion and General Control of DSD |
| Alperen Koruyucu | Design Consideration and Description for Components |
| Hakan Adaklı | Overall design of DSD and General Control of DSD |
| Mevlüt Eren Topal | Purpose, Statement of scope, Major constraints, Definitions, Acronyms and Abreviations |
| Ceyhun Erdönmez | Restrictions, limitations, and constraints, Overall design of DSD and General Control of DSD |
| Ahmet Bozbay | System Structure and Architecture diagram |