TED UNIVERSITY

CMPE 491

Senior Design Project I

MESAJLA

Analysis Report

22.11.2024

Group Members:

- **Ahmet Tunahan Küçükgökce**
- **Altar Gürsoy**
- **Burak Eren Birinci**
- **Serdar Kemal Topkaya**

1. Introduction

This analysis report focuses on the implementation of the systems and the integration of the functionalities of our project which combines chatting application with responses that are created from generative AI. This report is also backed up with the requirements that we follow during the implementation. Other than that, there are UML diagrams which show the inter-connections between the subsystems(such as class diagram, use case diagram etc.). Here, the concepts and the goals of the project are being shown. Finally, the report introduces a brief mock-up to display the outline of the interface and functionalities of this project.

2. Current System

In the current state, there are some chatting applications that are being used like WhatsApp or Telegram comes with a simple and easy to use design, provides their users to communicate peer-to-peer or group communication. They do their job well. However, when it comes to generating a response for user, they only offer limited short answers and can’t follow up the overall chatting that is continuing. Also, the implementation of the AI to these programs hasn’t been done(yet).

3. Proposed System

In our system, we aim to provide a chatting application that is secure, reliable and simple to do tasks. Besides those, we construct a generative AI that provides options to fast reply based on the given messages or from the dataset that we produced. These responses should follow the ethical manners in the existing chatting environment. Also, we create a secure database for user authorization to prevent data leakage.

3.1 Overview

Here, the outlines of the system are divided into sections. The main goal in this chapter is to document the capabilities and responsibilities of the application that is being constructed. It mentions the requirements, structures and subsystems of the project and explains them.

Also, there's images to show the key aspects of the project such as login/sign-up page, chatting environment, suggestions of the generative AI and so on.

3.2 Functional Requirements

- **User Authentication**
    - **Sign up:** Users should be able to create an account using their email address and password.
    - **Login:** Users must log in to access the interface via their email address and password.
    - **Changing Password:** Users should have an option to change their password to recover their account.
    - Security Question: Users should have an option to create a security question which only the one who sets the question knows the answer for their account security.
- **User Profile Management**
    - **Profile Editing:** Users can edit their profiles information (name, profile photo, status).
    - **Privacy Settings:** Users can control who sees their name, photo and status.
    - **Security Settings:** Users can manage other accounts they communicate with (block, unblock, delete chat).
    - **Friend Request:** Users can send friend requests to other users using their email address to start communicating with them.
    - **Friend List:** Users can manage their friend list (add, delete) and friend request (accept, decline).
- **Messaging Features**
    - One-on-One Chat: Users can send and receive text messages in real time with another user.
    - Multimedia Sharing: Users can send images, videos, and documents.
    - Message Status Indicators: Users see the indicators for sent.
- **Notifications**
    - Notifications: Users receive notifications for new messages.

3.3 Non-Functional Requirements

- **Performance**
    - The application should handle many users at the same time with minimum deceleration.
    - Messages should be delivered between the users in real-time (in a second approximately).
- **Security**
    - The data of user stored securely, and user privacy-compliant not be violated.
- **Usability**
    - Simple and practical interface for the user comfort and usage of the program.
- **Scalability**
    - The application must be capable of scale system to adapt the user growth and message load without any performance degradation.
- **Compliance and Regulations**
    - Accept and comply with the terms of data protection regulations (GDPR).
    - Take consent of users for collecting and using their data.

3.4 Pseudo Requirements

- **User Friendly**
    - The app should be easy to navigate for all types of users, even if the app doesn't have a guide that shows how to use it.
- **Browser Compatibility**
    - Although not specified in the requirements, the software must be compatible with all web browsers and work seamlessly on them.
- **System Performance**
    - Even without any performance metrics, the app should open in under 3 seconds.
- **Data Recovery**
    - In the case of failure, even if no data recovery action has been requested, the system must be built to allow for data recovery

3.5.1 System Models

Scenario 1: New User Registration

Description: A new user wants to create an account on the platform.

1. The user navigates to the Sign-up Page.
2. The user fills out all required fields (email, password, and other information).
3. The user selects a security question and provides an answer.
4. Once all required fields are completed, the "triangle" button becomes active.
5. The user clicks the "triangle" button to complete the registration process.
6. If successful, the user is directed to the Profile Setup Page.

Scenario 2: User Login

Description: A user with existing account logs into the application.

1. The user navigates to the Sign-in Page.
2. The user enters their registered email and password.
3. If the information is valid, the "triangle" button activates.
4. The user clicks the button to log into the application.

Scenario 3: Setting Up a Profile

Description: A newly registered user sets up their profile for the first time.

1. After signing up, the user is redirected to the Profile Setup Page.
2. The user enters their name and surname to complete the setup.
3. Once all fields are completed, the "triangle" button activates, allowing the user to go to the next page.

Scenario 4: Messaging a Friend

Description: A user accesses the chat page to message a friend.

1. The user navigates to the Chat Page.
2. The user selects a friend from the list of Direct Messages (DMs).
3. The user's chat box opens, displaying the profile picture and name of the friend.
4. The user types a message or chooses an AI-generated response to send.
5. If needed, the user can send emojis or files or generate new AI options.

Scenario 5: Creating and Managing Groups

Description: A user creates and interacts with a group on the platform.

1. The user navigates to the Groups section on the Chat Page.
2. The user creates a new group and adds friends.
3. The user views the list of groups, the latest messages, and details such as the name of the last sender and the message time.

3.5.2 Use Case Model

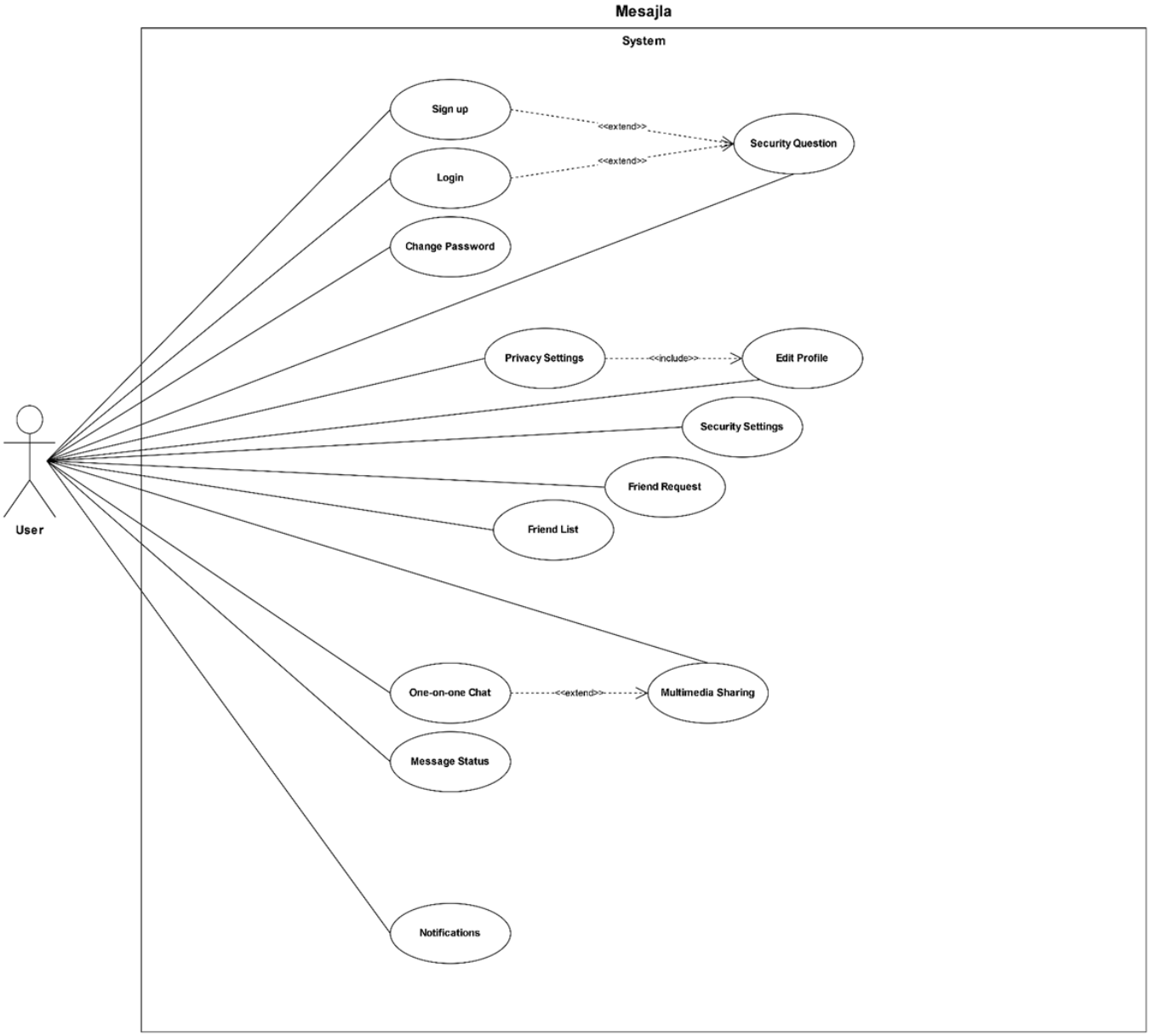

Figure 1: Use Case Model to show the functionalities of the system.

This use case diagram shows the main features of the Mesajla System and how a user interacts with them. The user can:

• Basic actions: Sign up, log in, and change their password.

• Settings: Manage privacy and security settings and edit their profile.

• Social Functions: Send friend requests, view a friend list, and chat one on one.

• Additional features: Share multimedia, check message status, and receive notifications.

Some optional actions, like after signing up and logging in, involve answering security questions. Multimedia sharing is an extra feature added to chatting. This diagram shows all the main things users can do in the system.

3.5.3 Object and Class Model

Class Model

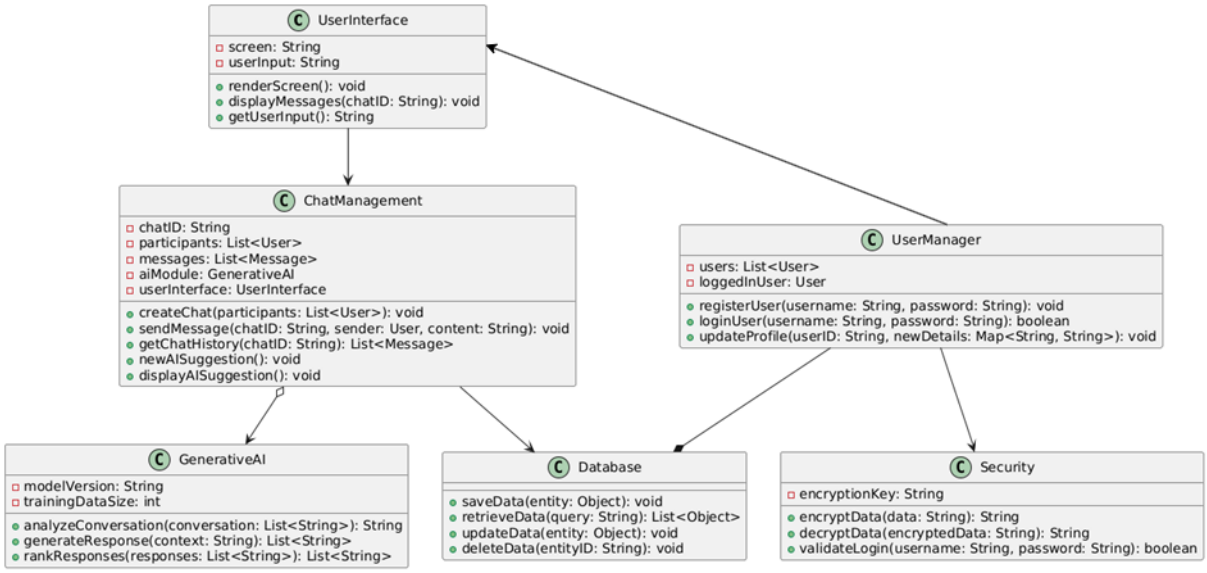

Figure 2: A class diagram for the chatting project.

- **Security class:** To handle authentication of the users based on their inputs.
    - encryptData(String data): It encrypt the input (specifically password) of the user during the login.
    - decryptData(String encryptedData): The decrypting process of the password when the original password is needed.
    - validateLogin(String username,String password): Users have access to their page if their inputs match.
- **UserManager class:** To let users login/register or update their information.
    - registerUser(String username,String password): Let's user to create an account.
    - loginUser(String username,String password): returns true if the values of the input match.
    - updateProfile(String userID, String newDetails): Allows user to update their information.

- **Database class:** Stores the key collection of data such as chat history or user password/username.
    - saveData(Object entity): Pushes an object into database.
    - retrieveData(String query): Selects data with a query line.
    - updateData(Object entity): Updates an entity inside database.
    - deleteData(String entityID): Deletes an entity inside database.
- **GenerativeAI class:** Generates responses to user from either the chat history or the dataset based on the version of the AI.
    - analyzeConversation(List<String> conversation): Training the AI with the chat history
    - generateResponse(String context): Generates suggestion to the conversation.
    - rankResponses(List<String> responses): Sets precedences for the responses.
- **ChatManagement class:** To control the chat environment between the parties.
    - createChat(List<User> participants): Creates chat room for users.
    - sendMessage(String chatID, User sender, String content): Pushes message into environment.
    - getChatHistory(String chatID): Returns chat history as a list.
    - newAISuggestion(): Lets' users to delete the previous responses from AI and get a new one.
    - displayAISuggestion(): To show the response of the AI.

Object Model

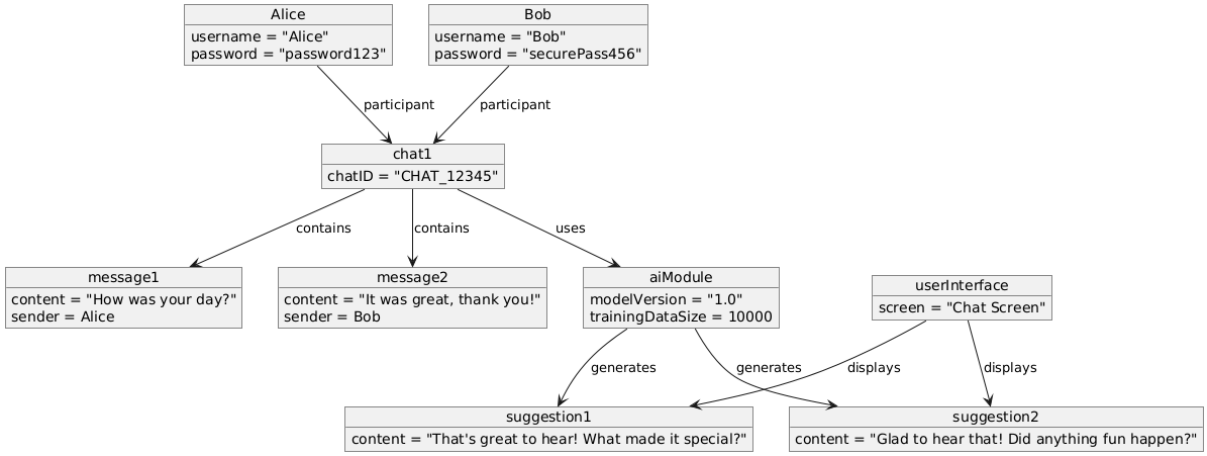

Figure 3: An object class based on a scenario of chatting between ‘Alice’ and ‘Bob’ which shows the responsibility of AI.

Here, there’s a case scenario of two users (Alice and Bob) which communicate in a chat room. Alice starts the conversation and asks something. Bob replies relevantly to the question. After that, the generative AI module, which is responsible for training chat history, analyzes the conversation, and produces 2 suggestions based on that chatting. Then, this suggestions is displayed in the Alice’s interface.

3.5.4 Dynamic models

1. Login and Registration Sequence Diagram:

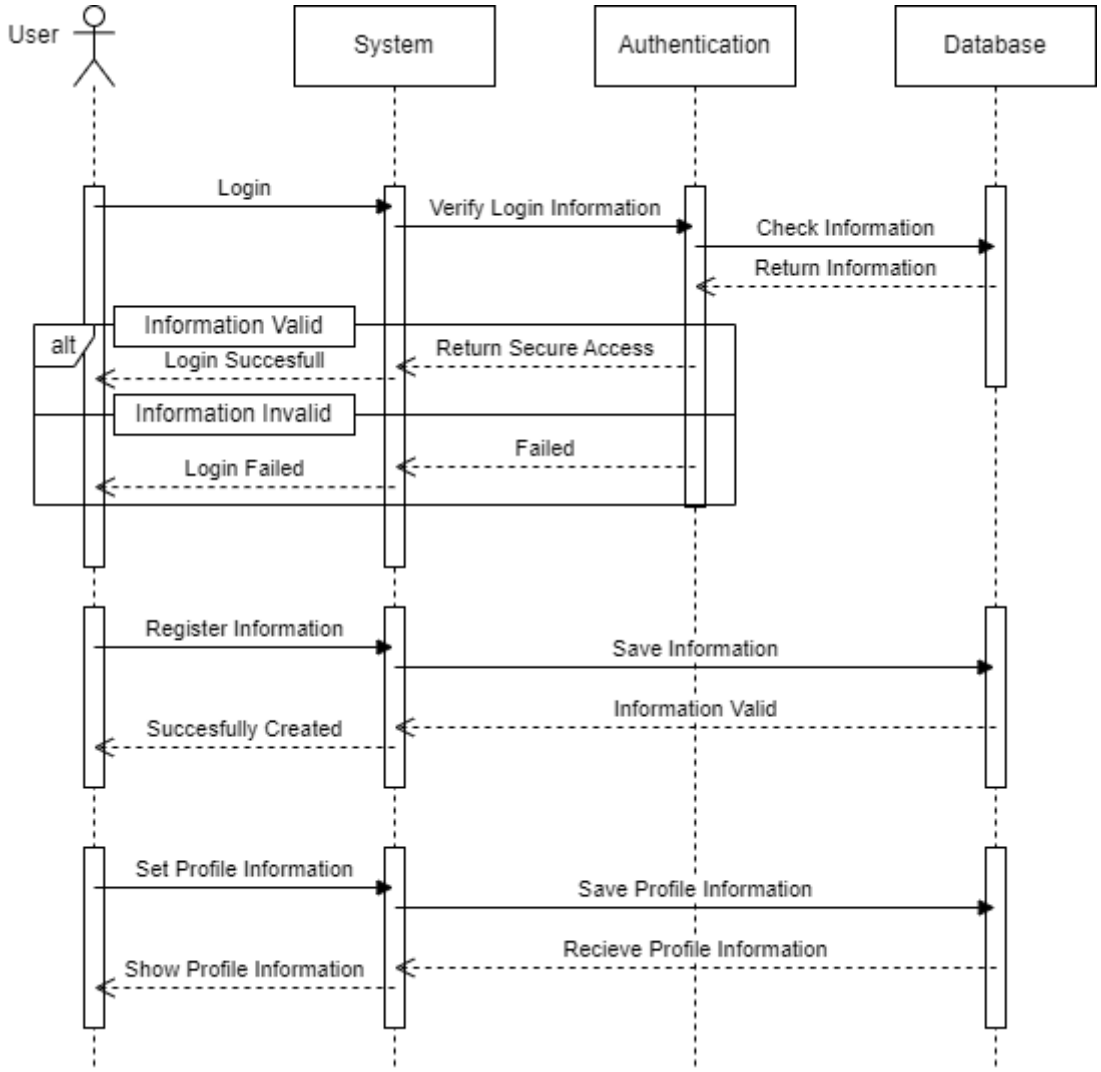

Figure 4: A sequence diagram that shows the process of login/registration process of a user.

- **Login Process**:
  - **User logs in**: The user submits login credentials.
  - **Authentication**: Information is verified by the system, and the database checks if it is valid.
  - **Success or Failure**:
    - If valid: The system grants access and returns a secure token.
    - If invalid: An error is shown to the user.

- **Registration Process**:
  - **User registers**: Inputs personal information.
  - **Data is saved**: The database validates and stores the user data.
  - **Success**: Confirmation is shown to the user.
- **Profile Setup**:
  - The user sets up their profile, and the data is saved in the database.
  - The profile information is retrieved and displayed.

2. Chat Messaging Sequence Diagram:

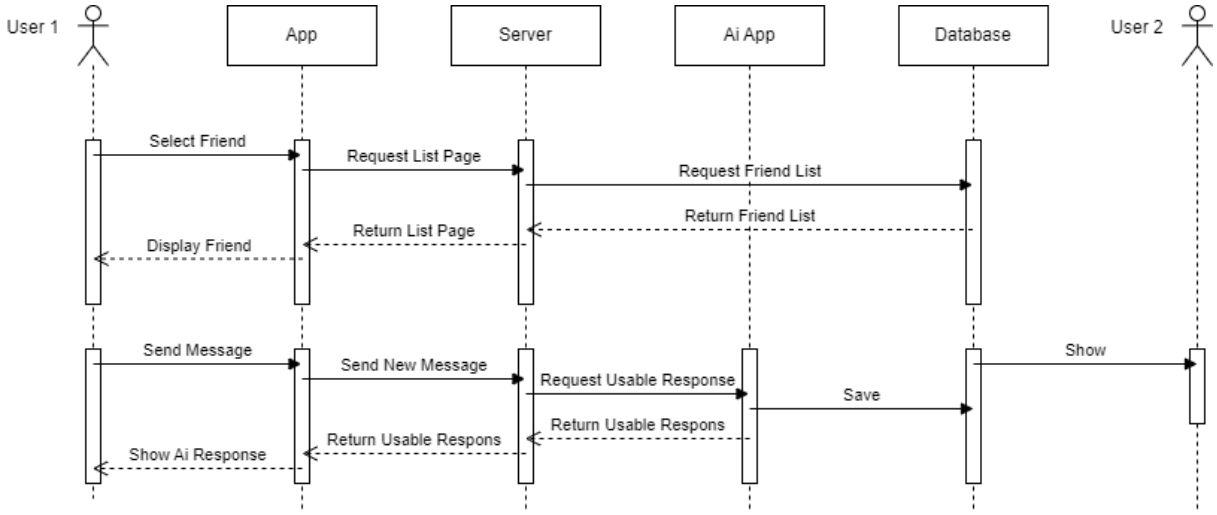

Figure 5: A sequence diagram that shows the communication processes of two unique users.

- **Friend Selection**:
  - **User selects a friend**: The app requests the friend list from the server.
  - **Friend list is displayed**: The server fetches and returns the friend list from the database.
- **Message Sending**:
  - **User sends a message**: The app sends the message to the server.
  - **AI or Usable Responses**:
    - If AI is involved, a request is made for usable responses.
    - The server processes and stores the message.
- **Response**:
  - The AI or database returns a response.
  - The app displays the response to the user.

3. Adding a Friend Sequence Diagram:

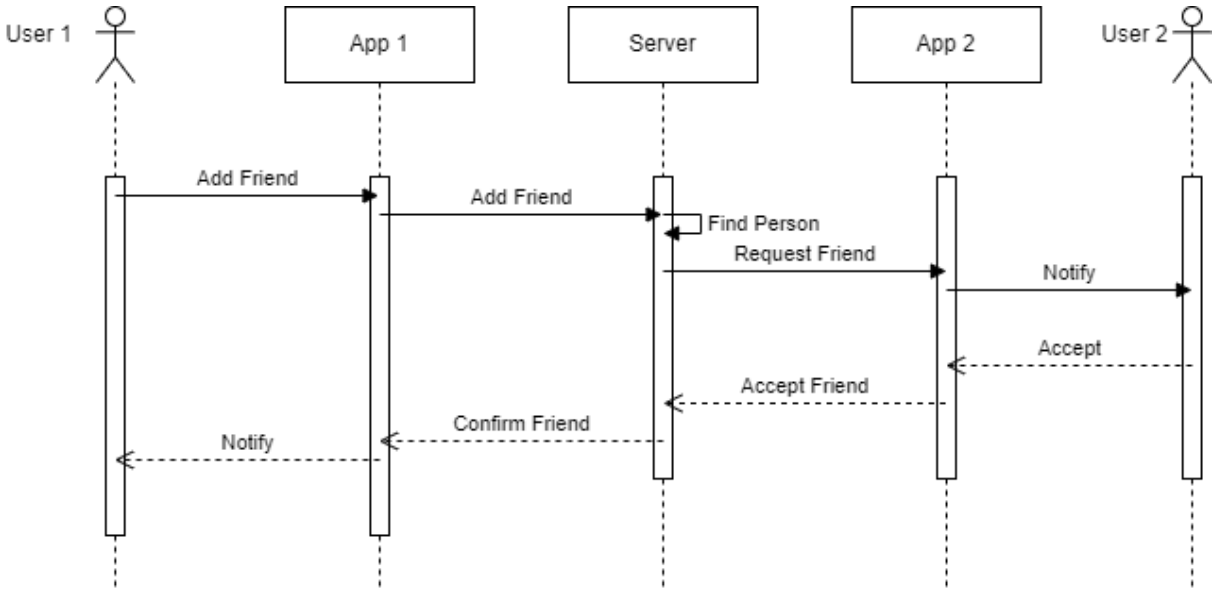

Figure 6: A sequence diagram that shows user adding functionality.

- **Friend Request**:
  - **User initiates request**: App 1 sends a "friend add" request to the server.
  - **Server processes request**: The server finds the person in the system and sends a friend request to App 2.
- **Friend Acceptance**:
  - **User in App 2 accepts**: App 2 sends a confirmation back to the server.
  - **Notification**: Both users are notified of the new connection

4. User Managing Sequence Diagram:

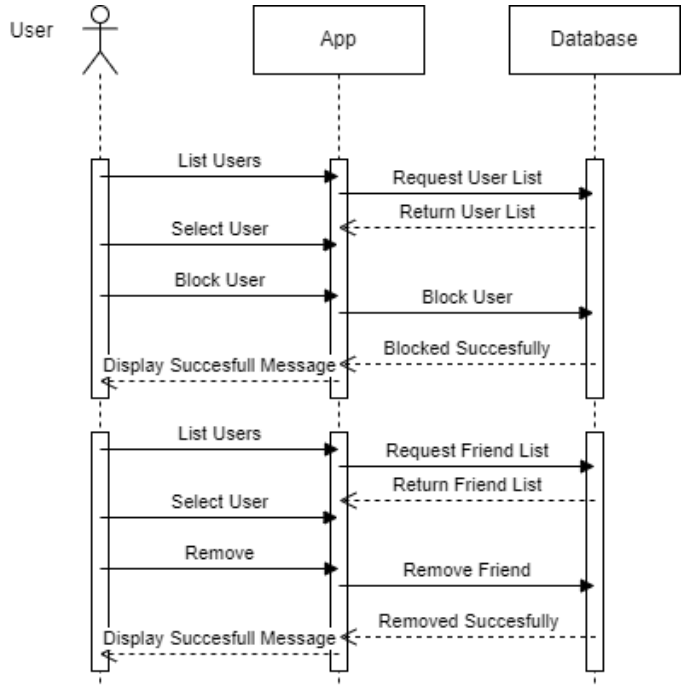

Figure 7: A sequence diagram that operates the user management.

- **Block User:**
    - **User uses block:** The App fetches the user list and allows the user to select another user to block.
    - **Request sent:** The App sends a "block user" request to the database.
    - **Database processes request:** The database processes the block request and confirms the action.
    - **Notification:** The App displays a "Blocked Successfully" message to the user.
- **Remove Friend:**
    - **User uses removal:** The App fetches the user's friend list and allows the user to select a friend to remove.
    - **Request sent:** The App sends a "remove friend" request to the database.
    - **Database processes request:** The database processes the removal request and confirms the action.
    - **Notification:** The App displays a "Removed Successfully" message to the user.

3.5.5 User Interface - Navigational Paths and Screen Mock-ups

Figure 8: Sign-up page

- If you do not have an account, fill in the information and sign up. If you have an account, click the sign in button.
- All information must be filled in for the triangle button to be active.

Figure 10: Sign-up page, security questions

- Select one of the options for security question.

Figure 11: Sign-up page

- All options are filled to finish the registration phase. The triangle color has turned black, so you can click to go to the next page.

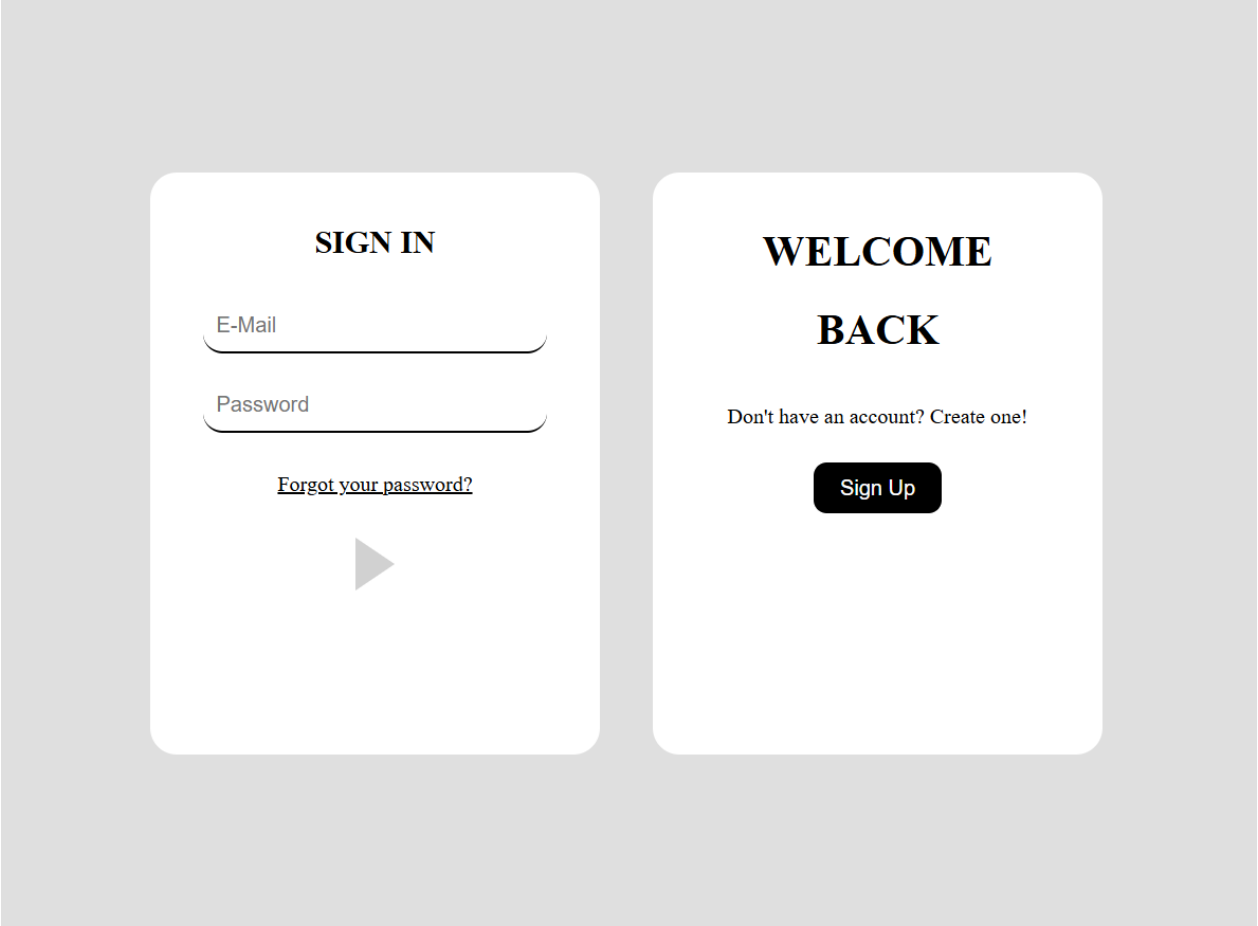

Figure 12: Sign-in page

- To log in to the application, enter the email and password you registered with the system.
- If you do not have an account, click on the sign-up button.

Figure 13: Sign-in page

- The necessary information to log in has been entered, the color of the triangle button has turned black, so you can go to the next page by clicking the button.

Figure 14: Profile setup page

- If you are registered to the system for the first time, you will be directed to this page after the sign-up page.

Figure 15: Profile setup page

- Complete your profile settings by entering your name and surname to proceed to the next page.
- When the triangle button turns black, click on it to go to the next page.

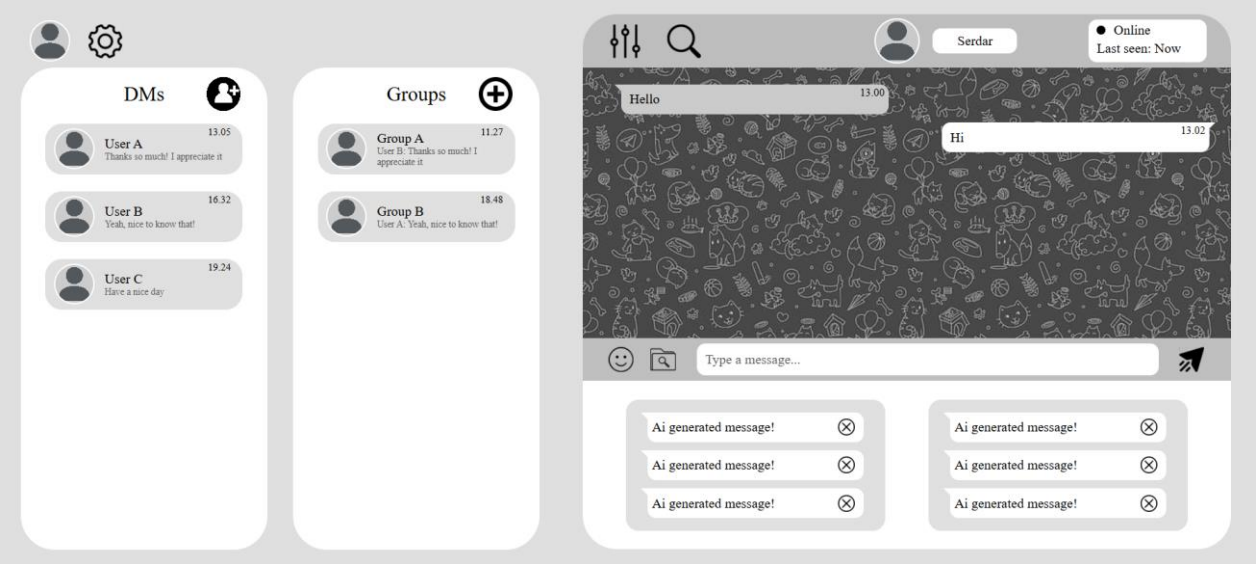

Figure 16: Chat page

- On this page you can talk to your friends, create groups, set up your profile, see and use ai generated message options.

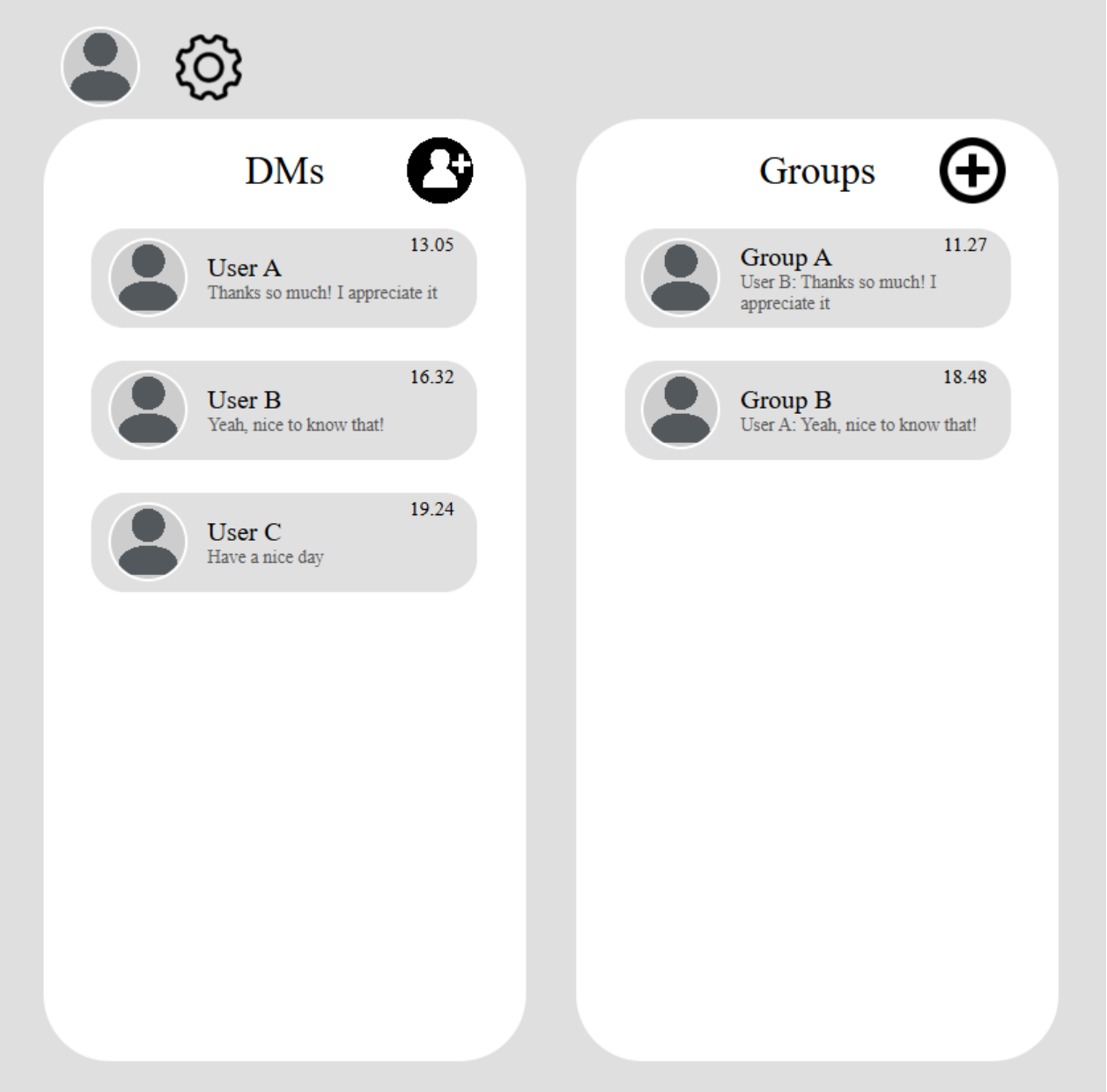

Figure 17: Chat page, DMs and Groups

- You can see the friends you are messaging with, add new friends, and see the content and time of messages sent to you.
- You can see the groups you are in, add new groups, see the name of the person who last wrote a message in the group, content and the time of the last message sent.

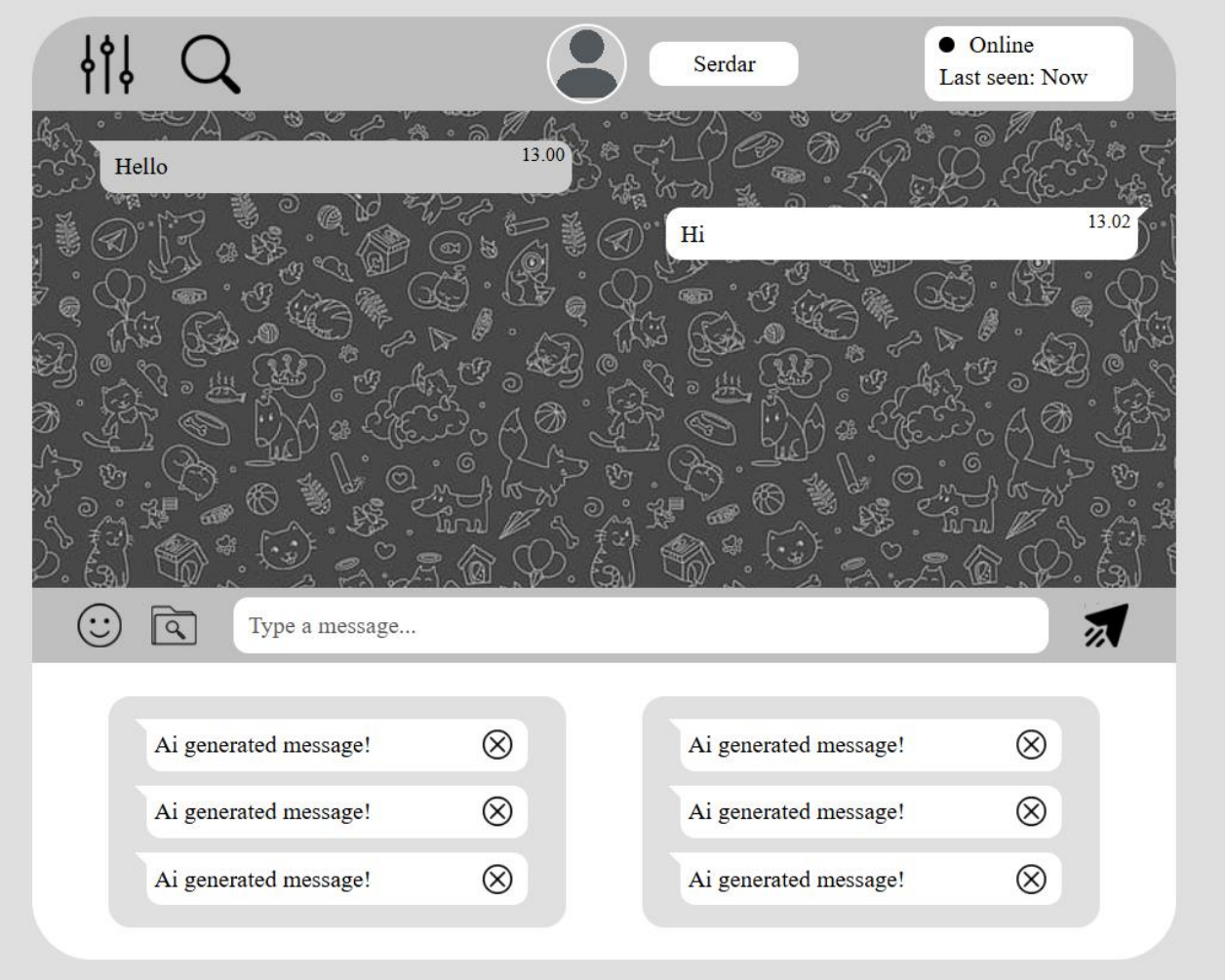

Figure 18: Chat page

- You can see the profile picture, name and online status of the person you are talking to.
- You can choose the answers you will send from the AI generated options. If you don't like an option, you can click on the cross symbol to bring new options.
- You can send emojis and files.

5. Glossary

- GDPR: General data protection regulation
- Ai: Artificial intelligence
- UML: Unified Modelling Language

6. References

General Data Protection Regulation (GDPR). (n.d.). *General Data Protection Regulation (GDPR) – Official Legal Text.* Retrieved November 22, 2024, from https://gdpr-info.eu/

PlantUML. (n.d.). *PlantUML: Open-source UML tool*. Retrieved November 17, 2024, from https://plantuml.com/

Diagrams.net. (n.d.). *Diagrams.net: Online diagramming tool*. Retrieved November 17, 2024, from https://app.diagrams.net/