Index

1. Project Description

Project Title: Chat System

Overview:

Create a user-friendly web-based chat application akin to popular messaging platforms. Users can register accounts, manage profiles, send/receive friend requests, and engage in one-on-one conversations. The system prioritizes simplicity and efficiency in communication.

Key Features:

- User registration and authentication
- Profile management with profile picture support
- Friend request functionality (send, accept, reject, withdraw)
- One-on-one chat interface
- User search by username
- Logout option for secure session management

Timeline: March-2024 to April-2024

Team Members:

1. Aaditya Patel
2. Devarshi Patel
3. Aarsh Patel

Expected Outcome:

A functional and intuitive chat system providing users with a convenient platform for connecting and communicating with friends.

2. Technologies Used

1. Frontend Technologies:

- Technology: React.js
- Where Used: Frontend development
- Purpose/Justification:
    - React.js was chosen for its component-based architecture, which facilitates modular development and reusability of UI components.
    - It provides a virtual DOM for efficient rendering and updating of UI elements, enhancing the overall performance of the application.

2. Backend Technologies:

- Technology: Spring Boot (Java)
- Where Used: Backend development
- Purpose/Justification:
    - Spring Boot was selected for its simplicity and convention-over-configuration approach, which accelerates the development of RESTful APIs and web applications.
    - It offers robust features such as dependency injection, auto-configuration, and built-in security, making it suitable for building scalable and secure backend systems.

3. Database Technology:

- Technology: MySQL
- Where Used: Database management
- Purpose/Justification:
    - MySQL was chosen as the relational database management system (RDBMS) for its reliability, performance, and ease of use.
    - It provides robust support for SQL queries, transactions, and data integrity constraints, making it suitable for storing user profiles, friend relationships, and chat messages in the Simple Chat System.

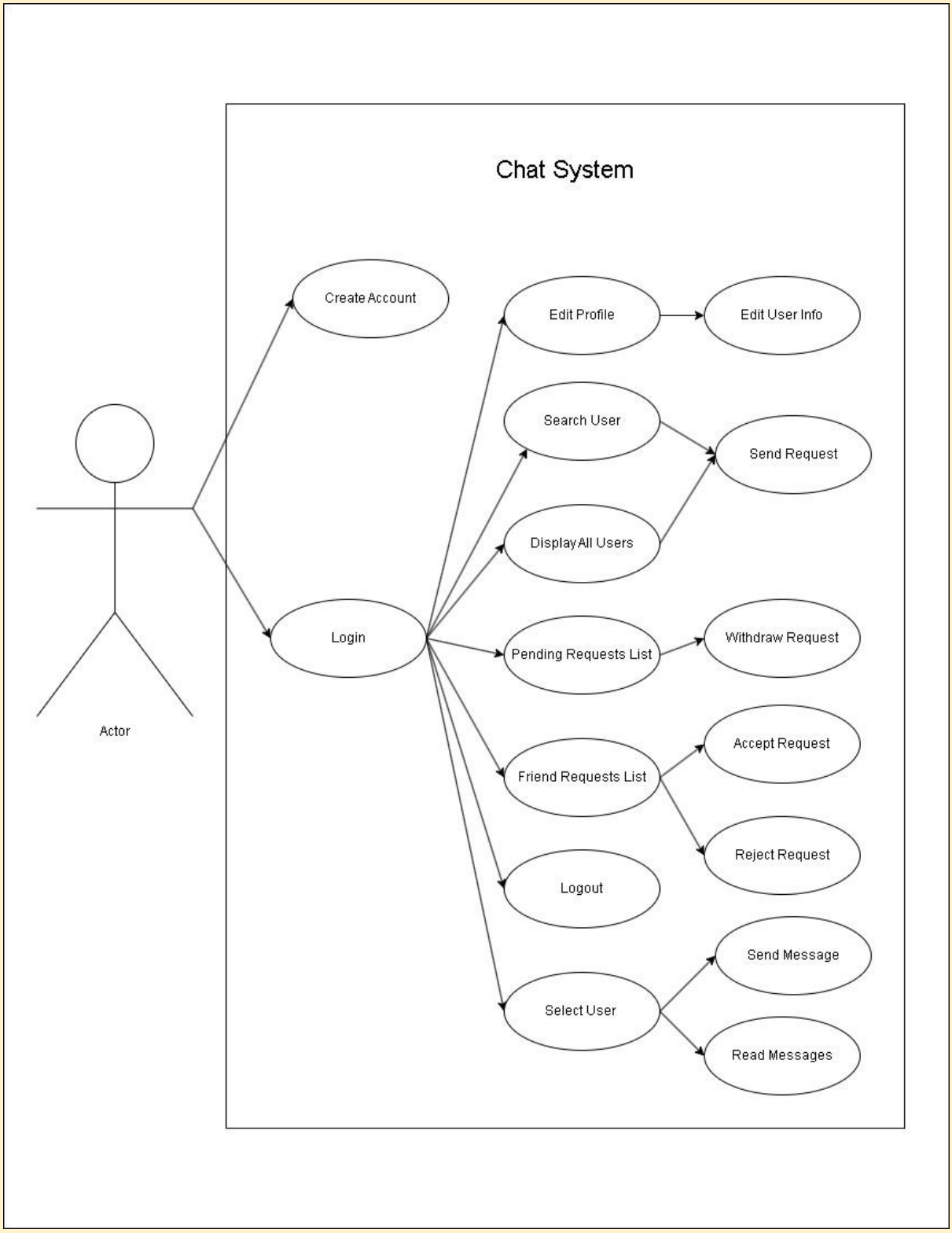
4. Cloud Platform:

- Technology: Amazon Web Services (AWS)
- Where Used: Database hosting
- Purpose/Justification:
    - AWS was selected for hosting the MySQL database on Amazon RDS (Relational Database Service).
    - AWS offers scalable and cost-effective cloud infrastructure, ensuring high availability, durability, and security of the database.

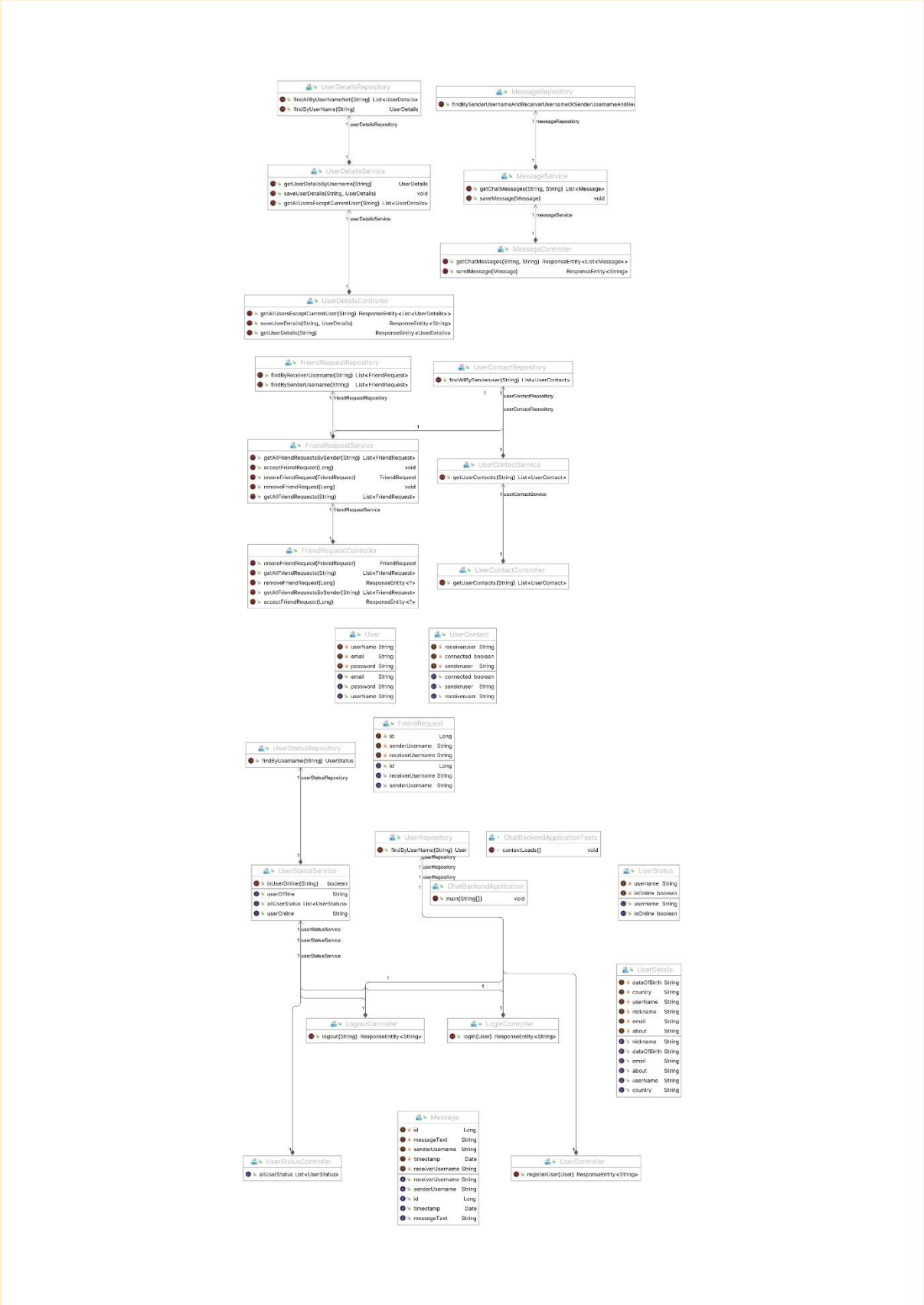
5. Additional Technologies:

- Technology: HTML/CSS/JavaScript, Axios (for HTTP requests), Spring Security (for authentication)
- Where Used: Frontend development (HTML/CSS/JavaScript), Frontend-backend communication (Axios), Security (Spring Security)
- Purpose/Justification:
    - HTML/CSS/JavaScript are fundamental technologies for building the frontend user interface and implementing client-side functionality.
    - Axios is used for making asynchronous HTTP requests from the frontend to the backend API endpoints.
    - Spring Security provides authentication and authorization mechanisms to secure the application's endpoints and protect user data.

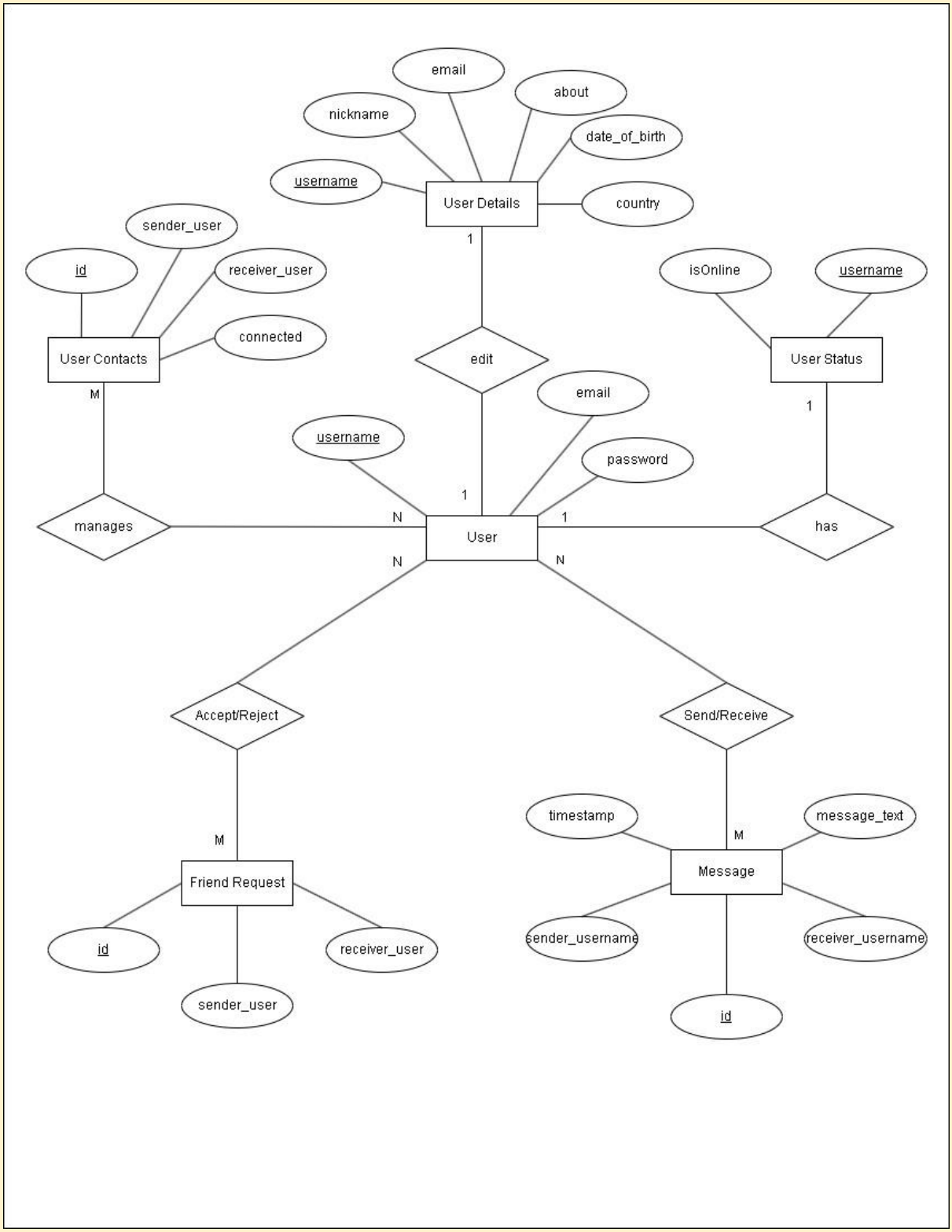
3. Use-Case Diagram



4. Class Diagram



5. Database Design



6. Test Use-cases

1. User Registration:

- Description: Test the user registration process to ensure that users can create accounts successfully.
- Steps:
  1. Navigate to the registration page.
  2. Fill in the required fields (e.g., username, email, password).
  3. Click on the "Sign Up" button.
- Expected Result: User account is created, and the user is redirected to the login page.

2. User Login:

- Description: Test the user login process to ensure that registered users can log in securely.
- Steps:
  1. Navigate to the login page.
  2. Enter valid credentials (username/email and password).
  3. Click on the "Login" button.
- Expected Result: User is authenticated and redirected to the dashboard or home page.

3. Sending Friend Requests:

- Description: Test the functionality of sending friend requests between users.
- Steps:
  1. Navigate to connections component from navbar.
  2. Click on the "Add Friend" button.
- Expected Result: Friend request is sent successfully, and the recipient.

4. Accepting/Rejecting Friend Requests:

- Description: Test the functionality of accepting or rejecting incoming friend requests.
- Steps:
  1. Navigate to the friend requests section.
  2. Choose to accept or reject the request.
- Expected Result: Friend request status is updated accordingly

5. Starting a Chat Conversation:

- Description: Test the functionality of initiating a chat conversation with a friend.
- Steps:
  1. Navigate to the home page.
  2. Click on the profile of the friend you want to chat with.
  3. Start typing a message in the chat interface.
- Expected Result: Chat interface opens with the selected friend, and messages can be sent and received in real-time.

6. Viewing Message History:

- Description: Test the functionality of viewing past messages .
- Steps:
  1. Open an existing chat conversation with a friend.
  2. Scroll through the message history to view previous messages.
- Expected Result: Previous messages are displayed chronologically in the chat interface, allowing users to review past conversations.

7. Logging Out:

- Description: Test the functionality of logging out of the application securely.
- Steps:
  1. Select the "Logout" option from the navbar.

- Expected Result: User session is terminated, and the user is redirected to the login page.

7. Implementation Details

1. Database Design:

1.1 Database Schema:

- Design the database schema to store users , user profiles, friend relationships, and chat messages.
- Define tables for users, user details , friend requests, friendships (connections), and messages, ensuring appropriate normalization and indexing for efficient data retrieval.

2. Backend Development with Spring Boot:

2.1 Data Access Layer:

- Implement data access logic using Spring Data JPA to interact with the MySQL database.
- Define repository interfaces and entity classes to perform CRUD operations on user profiles, friend requests, friendships, and messages.

2.2 API Endpoints:

- Create RESTful API endpoints to handle user authentication, profile management, friend requests, and messaging functionalities.
- Implement controllers to process HTTP requests , validate input data, and invoke service methods for business logic execution.

3. Frontend Development with React:

3.1 UI Components:

- Develop React components to display user profiles, friend lists, chat interfaces, and message history.

- Implement UI features for sending friend requests, accepting/rejecting requests, and initiating chat conversations with contacts.

3.2 Data Fetching and Rendering:

- Fetch user data, friend lists, and chat messages from the backend API endpoints using asynchronous requests (e.g., Axios).
- Update the UI dynamically to reflect changes in user connections, incoming messages, and chat history.

4. Chat Messaging System:

4.1 Message Storage:

- Store chat messages in the database with attributes such as sender ID, recipient ID, timestamp, and message content.
- Implement database queries to retrieve message history between two users for display in the chat interface.

5. Database Hosting:

5.1 Database Hosting:

- Host the MySQL database on Amazon RDS or a similar cloud database service, configuring backup policies and access controls as needed.
- Monitor database performance, storage usage, and query execution times to optimize system efficiency.

8. User Manual

1. Getting Started

1.1 Creating an Account:

- Visit the registration page of the Chat System.
- Fill in the required fields such as username, email, and password.
- Click on the "Sign Up" button to create your account.

1.2 Logging In:

- Enter your username and password on the login page.
- Click on the "Login" button to access your account.

2. Profile Management

2.1 Editing Profile:

- Once logged in, navigate to the profile section.
- Click on the "Edit Profile" option.
- Update your profile information as desired.
- Click on the "Save Changes" button to update your profile.

3. Connecting with Friends

3.1 Sending Friend Requests:

- To send a friend request, search for the user by their username using the search bar or go to the connections page from navbar.
- You can see there user’s details.
- Click on the "Add Friend" button to send a friend request.

3.2 Accepting/Rejecting Friend Requests:

- Navigate to the friend requests section.
- You can see all the available requests there.
- Choose to accept or reject the friend request.

3.3 Withdrawing Friend Requests:

- If you wish to withdraw a friend request, navigate to pending request section.
- You can see all the pending requests there.
- Select the option to withdraw the request.

4. Chatting

4.1 Starting a Conversation:

- Once friends with another user, you can see all the list of contacts on the home page.
- Click on one of the contact to open the chat interface.
- Type your message in the text box at the bottom of the screen.
- Press "Enter" or click on the send button to send your message.

4.2 Viewing Messages:

- Incoming messages will appear in the chat interface.
- Click on the chat to view the conversation history.

5. Logging Out

5.1 Logging Out:

- To log out click on the Logout icon from navbar.