HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR INTERNATIONAL EDUCATION

FINAL REPORT

PROJECT ON SOFTWARE ENGINEERING

BUILDING A SOCIAL NETWORK SYSTEM

INSTRUCTOR: Nguyen Tran Thi Van, MSc.

Group: 04

Students:

21110759 Ho Thanh Duy

21110806 Vu Tien Trinh

Ho Chi Minh City, Dec 2024

HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR INTERNATIONAL EDUCATION

FINAL REPORT

PROJECT ON SOFTWARE ENGINEERING

BUILDING A SOCIAL NETWORK SYSTEM

INSTRUCTOR: Nguyen Tran Thi Van, MSc.

Group: 04

Students:

21110759 Ho Thanh Duy

21110806 Vu Tien Trinh

Ho Chi Minh City, Dec 2024

OBJECTIVES FOR PROJECT ON SOFTWARE ENGINEERING

Full name: Hồ Thanh Duy ….……ID: 21110759

Full name: Vũ Tiến Trình…….……ID: 21110806

Major: Software Engineering

Project : Building a social network system

Contents:

Theory:

- Learn about RESTful API with SpringBoot Framework
- Learn about building website with ReactJs Framework
- Learn about Spring Security (JWT, Oauth)
- Learn about Docker to deploy production
- Learn about Websocket protocol

Practice: Build a social network system with base features:

- Profile Management: Update personal info, manage profile pictures, and view profiles.
- Post Management: Create, edit, delete posts; like, comment, and share content.
- Authentication: Register, log in (JWT), with Oauth and password recovery.
- Relationships: Send friend requests, follow/unfollow, and manage connections.
- Real-Time messaging: WebSocket messaging with basic online chatting only sends text.
- File Management: Upload and manage media in posts and profiles.

Time: 15 weeks (from 19/08/2024 to 30/11/2024)

Sign of student: ……

Sign of student: ……

TP.HCM, day…. Month…. Year …

HEAD OF DEPARTMENT	INSTRUCTOR
(*write, sign full name*)	(*write, sign full name*)

SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom – Happiness

INSTRUCTOR'S EVALUATION SHEET

Student Name: Ho Thanh Duy **Student ID:** 21110759

Student Name: Vu Tien Trinh **Student ID:** 21110806

Project title: Building a social network system.

Instructor: Nguyen Tran Thi Van **, Msc**

EVALUATION

1. Content of the project:

.....

.....

2. Strengths:

.....

.....

3. Weaknesses:

.....

.....

4. Approval for oral defense? (Approved or denied)

.....

5. Overall evaluation: (Excellent, Good, Fair, Poor)

.....

6. Mark: ……. (in words: .....)

Ho Chi Minh City, December 2024

INSTRUCTOR

(Sign with full name)

SOCIALIST REPUBLIC OF VIETNAM

Independence – Freedom – Happiness

REVIEWER'S EVALUATION SHEET

Student Name: Ho Thanh Duy **Student ID:** 21110759

Student Name: Vu Tien Trinh **Student ID:** 21110806

Project title: Building a social network system.

Reviewer: Mai Anh Tho, **PhD**

EVALUATION

1. Content of the project:

.....

.....

2. Strengths:

.....

.....

3. Weaknesses:

.....

.....

4. Approval for oral defense? (Approved or denied)

.....

5. Overall evaluation: (Excellent, Good, Fair, Poor)

.....

6. Mark: ……. (in words: .....)

Ho Chi Minh City, December 2024

REVIEWER

(Sign with full name)

ACKNOWLEDGEMENT

First and foremost, I would like to express our heartfelt gratitude to Mr. Nguyễn Trần Thi Văn for his unwavering support and guidance throughout the process of researching and implementing this topic. I'd want to express our gratitude to the teacher for his passionate instruction and guidance in selecting themes, techniques, and rectifying flaws in the implementation process.

I'd also want to express our gratitude to the professors at Ho Chi Minh City's University of Technology and Education, particularly the Faculty Lecturers, who energetically taught and shared a wealth of information and expertise with us during our schooling. To assist me in effectively completing the topic.

However, faults made during the implementation of the topic may be overlooked by teachers. I eagerly await instructor feedback so that I may get more experience and increase my expertise in order to better serve the learning and working process in the future.

Thank you sincerely!

Hồ Thanh Duy

Vũ Tiến Trình

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1. Reasons for choosing the topic

The development of technology in the 4.0 era today has been expanding with the emergence of social networking applications that have been a trend that brings many utilities to help people connect quickly, anytime, anywhere in the world through the Internet. Social networking applications have had many influences on people's lives such as Facebook, Zalo, Instagram, Reddit and many related applications. However, these social networks have not really targeted a specific research object. This is a topic that the group aims to bring a virtual social environment to users.

2. Purpose of the topic

The purpose of the topic is to target the rapid development of the need to share and find information quickly from specific websites instead of using Google to search for information, which takes a lot of time. With Smartphones, users can completely search for information quickly in the most convenient way.

3. Research subjects and scope

The target audience includes 2 groups of subjects: the group of subjects who have knowledge, experience and experience through some prominent products on the market such as TikTok, Facebook and the remaining group is low-tech users who do not have much experience with advanced social networking applications today.

In which the group of subjects on technology knowledge includes in-depth research on: Open source Java Spring boot, Library ReactJs, MongoDB database, Postgresql and libraries related to the research issue. In which, in-depth research on the topic of real-time with technologies such as WebSocket to send notifications, chat quickly and conveniently for users, specialized technologies such as Json Web Token, ...

4. Expected results

The anticipated outcomes of the project include:

- Gaining a clear understanding of popular forms of online exams and the technologies used to build web platforms, specifically ReactJs, Spring boot and RESTful API.
- Successfully developing a web application that supports test creation and test-taking with a simple and user-friendly interface. Ensure the implementation of essential functionalities for test takers, test creators, and administrators as initially intended.

5. Link Github Source code and Website

Front-End: https://github.com/SOCIAL-NETWORK-2425/social-network-frontend

Back-End: https://github.com/SOCIAL-NETWORK-2425/social-network-backend

CHAPTER 2. THEORETICAL FOUNDATIONS

2.1. NodeJS

2.1.1. General NodeJS

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project!

Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.

A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

2.1.2. Principles of operation

Event-Driven Architecture

Node.js operates on an event-driven architecture, centered around an event loop. This core mechanism continuously monitors the call stack and handles tasks asynchronously, ensuring efficient resource utilization. Key components of the event loop include:

- Timers: Execute callbacks after a specified delay (e.g., setTimeout).
- I/O Callbacks: Handle the completion of non-blocking I/O operations.
- Polling: Retrieves new I/O events.
- Check: Executes setImmediate callbacks.

Single-Threaded with Non-blocking I/O

Despite its single-threaded nature, Node.js excels in handling concurrent connections. It achieves this by offloading I/O and intensive operations to worker threads (via the libuv thread pool) or external systems. This non-blocking approach prevents the main thread from being blocked, allowing it to efficiently process multiple requests simultaneously.

Module and Package

Node.js leverages a modular architecture, providing both built-in and third-party modules.

- **Built-in Modules:** Essential modules like http (for creating servers), fs (file system operations), and path (handling file paths) are included by default.
- **NPM (Node Package Manager):** The world's largest software registry, offering millions of packages for various functionalities. Developers can easily install, manage, and share these packages to extend the capabilities of their Node.js applications.

2.1.3. Advantages

- **Speed**: Node.js is fast, leveraging an event loop and non-blocking I/O for handling multiple concurrent requests efficiently.
- **Productivity**: Node.js is ideal for real-time, single-threaded applications like chat and video streaming.
- **Error Handling**: Built-in mechanisms simplify catching and managing runtime errors using event-based error objects.
- **Cost-Effectiveness**: Node.js reduces IT costs through minimal hardware requirements and low maintenance expenses.
- **Faster Development**: Allows building high-performance apps quickly, ideal for microservices and rapid product launches.
- **Performance in Slow Networks**: Its single-threaded, non-blocking I/O model ensures fast response in environments with slow connections.
- **Scalability**: Highly scalable, it adapts quickly to changing data needs, and cloud providers enhance performance.

2.1.4. Disadvantages

- **Asynchronous Programming Model:** Asynchronous programming allows program elements to run independently in parallel or sequentially when dependent.

- **Unstable API:** An unstable API may change frequently, causing issues for developers who rely on consistency for their software.
- **Dealing with Relational Databases:** Relational databases are widely used but can be more complex to work with compared to non-relational ones, even with Node.js tools like MySQL modules or ORMs.
- **Memory Leaks:** Poorly coded Node.js applications can lead to memory leaks, but tools like heapdump and careful coding can prevent this.

2.2. ReactJS

2.2.1. General ReactJS

React is a versatile JavaScript library designed to construct user interfaces (UIs) for web applications. It empowers developers to create dynamic and interactive user experiences by breaking down complex UIs into reusable components.

2.2.2. Core concept of React

Components: These are self-contained building blocks that encapsulate UI logic and data. They can be either functional or class-based, offering flexibility in their implementation.

JSX: This syntax extension for JavaScript allows developers to write HTML-like structures within JavaScript code, making it easier to define the structure of UI components.

Virtual DOM: React employs a virtual DOM, a lightweight representation of the actual DOM. This optimization technique minimizes the number of DOM manipulations required, resulting in significant performance improvements.

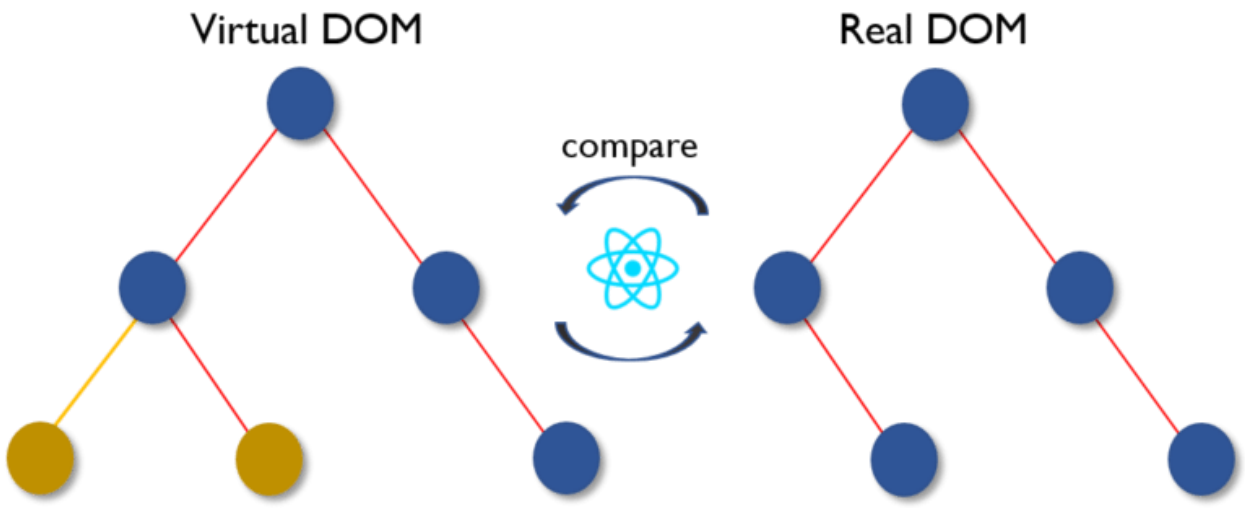


Figure 2-1. Virtual DOM (Document Object Model)

One-way data binding: Props are used to convey data from parent to child. It's straightforward to control and remedy problems using a simple data flow. With the characteristics listed above, ReactJS is used to create huge apps with data that changes over time. When the interface changes, the majority of the data changes as well.

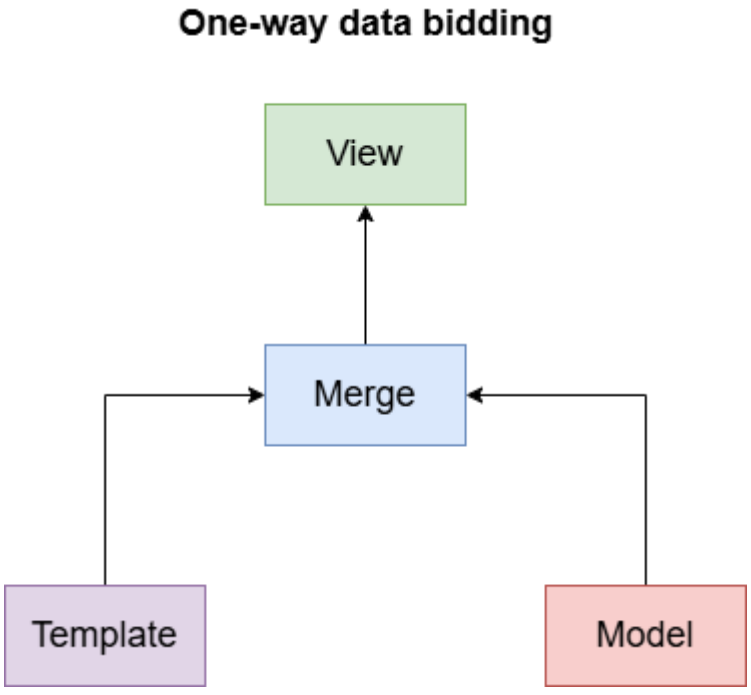


Figure 2-2. One-way data bidding model in ReactJS project

2.3. PostgresSQL

2.3.1. General PostgresSQL

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the POSTGRES project at the University of California at Berkeley and has more than 35 years of active development on the core platform.

PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on all major operating systems, has been ACID-compliant since 2001, and has powerful add-ons such as the popular PostGIS geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organizations.

2.3.2. Core principles of Postgres

This architecture enables a PostgreSQL system to cater to multiple clients, connecting either locally or through network. When the master process receives a client connection, it forks (ie, the creation of an independent process that will consume CPU and RAM and executes on its own) a new process dedicated to that specific connection. If multiple clients are connecting, each client gets a forked process.

Since each forked process consumes CPU cores and RAM, the number of clients connecting to the server simultaneously are limited by the available cores and RAM. Any new client requests after the server resources are exhausted will be declined. In such cases, clients have to retry.

2.3.3. Advantages

- PostgreSQL is an open-source database that provides affordable solutions without a license charge.

- PostgreSQL maintains data integrity by enforcing constraints, verifying data types, and enabling referential integrity through foreign vital regulations.
- PostgreSQL maintains data integrity by enforcing constraints, verifying data types, and enabling referential integrity through foreign vital regulations.
- PostgreSQL is built to manage demanding workloads and large-scale applications. Sharding facilitates horizontal scalability and offers performance-enhancing tools like query planning, caching, and parallel query execution.

2.3.4. Disadvantages

- Postgres is not owned by one organization. So, it has had trouble getting its name out there despite being fully featured and comparable to other DBMS systems
- Changes made for speed improvement requires more work than MySQL as PostgreSQL focuses on compatibility
- Many open source apps support MySQL, but may not support PostgreSQL
- On performance metrics, it is slower than MySQL.

2.4. Spring Boot Framework

2.4.1. General Spring boot

Spring Boot is an open-source Java-based framework that simplifies the development of standalone, production-grade Spring applications. It provides rapid application development experience by automating configuration and setup tasks, reducing boilerplate code, and offering a wide range of features out of the box.

Figure 2-3. Java Spring boot Framework Logo

2.4.2. Advantages

- Time-efficient and effortless development of Spring-based apps.
- Autoconfiguration of all components for a production-grade Spring application.
- Prebuilt embedded servers (Tomcat, Jetty, and Undertow), leading to accelerated and more productive app deployment.
- HTTP end-points, letting enter inner app features such as metrics, health status, etc and don't have XML configuration.
- A vast choice of plugins, facilitating developers' work with embedded and in-memory databases.
- Easy access to databases and queue services such as MySQL, Oracle, MongoDB, Redis, ActiveMQ, and others.
- Large community and a great variety of tutorials, facilitating the getting-to-know period.

2.4.3. Disadvantages

- Lack of control:
    - Due to the opinionated style, Spring Boot creates many unused dependencies that result in large deployment file size.
    - The challenging and time-consuming process of converting the legacy or existing Spring project into the Spring Boot apps.
- Not suitable for large-scale projects: Working seamlessly with microservices, according to many developers, Spring Boot doesn't fit well for building monolithic applications.

2.5. WebSocket

2.5.1. General Websocket

WebSocket is used as a communication protocol that provides full-duplex communication channels over a single, long-lived connection between a client and a server. In this protocol, there are no restrictions like HTTP that for a response you have to make a request first. The

server can send the message without getting any request and we use it in the chat feature and send notifications.

2.5.2. Advantages over traditional HTTP

- Persistent connections reduce bandwidth and latency compared to HTTP methods like long polling.
- Supported by major browsers and numerous libraries/frameworks across programming languages.
- Allows custom application-level protocols and extensions (e.g., pub/sub messaging).
- Instantly pushes data, ideal for unpredictable events like alerts.
- Enables simultaneous bidirectional messaging, perfect for multi-user applications like chat rooms.
- Outperforms one-way protocols like Server-Sent Events (SSE).

2.5.3. Disadvantages of WebSocket

- Complex Setup: More effort compared to simpler protocols like HTTP.
- Security Risks: Requires careful handling to avoid vulnerabilities like CSWSH.
- Scalability Issues: Persistent connections increase server resource use.
- Limited Support: Older or specialized browsers may lack full support.
- Overhead for Small Apps: May be excessive for low-frequency communication.
- State Management: Requires explicit handling of persistent sessions.

2.6. Library

2.6.1. Material UI

Definition

Material UI is an open-source React component library that implements Google's Material Design.

It includes a comprehensive collection of prebuilt components that are ready for use in production right out of the box and features a suite of customization options that make it easy to implement your own custom design system on top of our components.

Installation

- Using npm

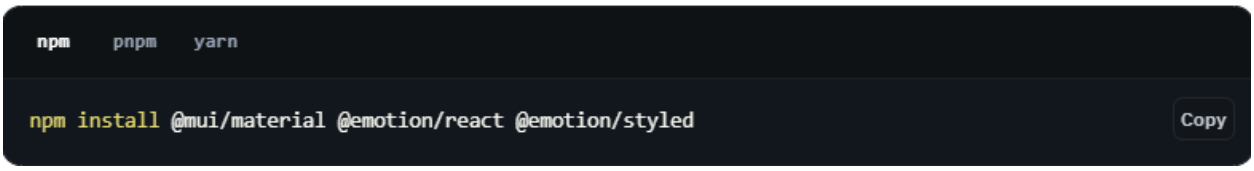


Figure 2-4. Installation Material UI with NPM

- Using yarn

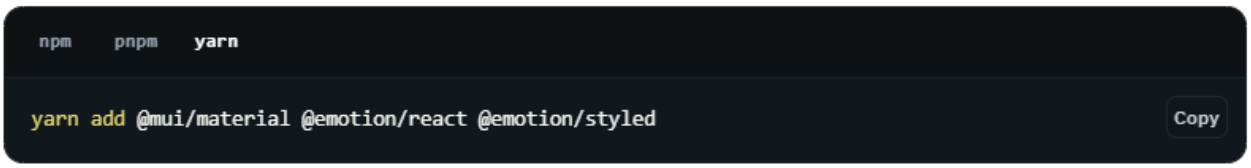


Figure 2-5. Installation Material UI with Yarn

2.6.2. Axios

Definition

Axios is a promise-based HTTP Client for node.js and the browser. It is isomorphic (= it can run in the browser and nodejs with the same codebase). On the server-side it uses the native node.js http module, while on the client (browser) it uses XMLHttpRequests.

Installation

- Using npm

Figure 2-6. Installation Axios with NPM

2.6.3. Redux

Definition

Redux is an open source JavaScript library for centralizing and managing application state. It is most commonly used with libraries like React or Angular to build user interfaces. Similar to Facebook's Flux architecture, it was created by Dan Abramov and Andrew Clark**.**

Redux is a pattern and library for managing and updating application state, using events called "actions". It serves as a centralized store for state that needs to be used across your entire application, with rules ensuring that the state can only be updated in a predictable fashion.

Core concept of Redux

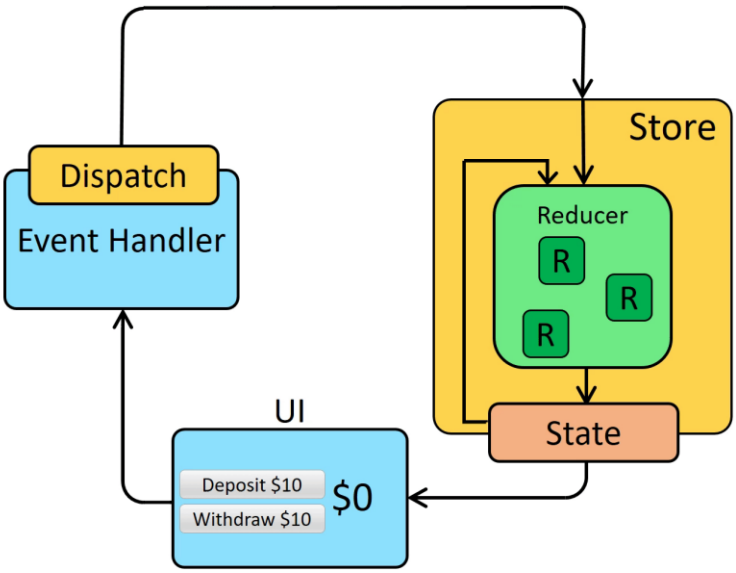


Figure 2-7. Operation of State Management Redux

Redux has three main components: Actions, Store, and Reducers:

- **Actions**: are events, the method we use to send data from the application to the Redux store, this data can be user interactions with the application, API calls.

- **Reducers**: are functions that take the current state of the application, perform an action and return a response about a new state. States are stored as objects and are defined how they change in response to an action sent to the store.
- **Store**: is the only one in an application that uses Redux to store state, which can be access stored state, change state, and register or unregister listeners via helper methods.

2.6.4. React router DOM

Definition

React Router DOM is an npm package that enables you to implement dynamic routing in a web app. It allows you to display pages and allow users to navigate them. It is a fully-featured client and server-side routing library for React.

React Router Dom is used to build single-page applications i.e. applications that have many pages or components but the page is never refreshed instead the content is dynamically fetched based on the URL. This process is called Routing and it is made possible with the help of React Router Dom.

Installation

- Using npm

```
> npm i react-router-dom
```

Figure 2-7. Installation React router DOM with NPM

CHAPTER 3. REQUIREMENTS CAPTURING AND MODELING

3.1. Current status survey

Social media applications have continued to grow and evolve in 2024, reflecting both regional and global shifts in user engagement. The number of social media users has surged to over 5 billion globally, marking an 8% annual growth. This trend illustrates not only increasing accessibility to technology but also a deepening integration of social media into daily life. Platforms like TikTok, Instagram, and YouTube have experienced notable growth in users, with TikTok seeing a dramatic rise in its ad audience, particularly among younger demographics. Instagram has also strengthened its position, attracting a significant portion of the U.S. population through its focus on visual storytelling and brand discovery.

The behavior of users across different platforms highlights evolving interaction patterns, with many using social media for exploring brands, engaging with video content, and connecting through shared interests. Video formats, especially longer ones, continue to dominate engagement metrics, showcasing their role in capturing attention and driving conversations. Meanwhile, regional penetration varies widely, with Europe maintaining some of the highest usage rates, while areas in Africa show opportunities for growth.

An analysis of the current landscape and performance of various social media application:

3.1.1. FACEBOOK

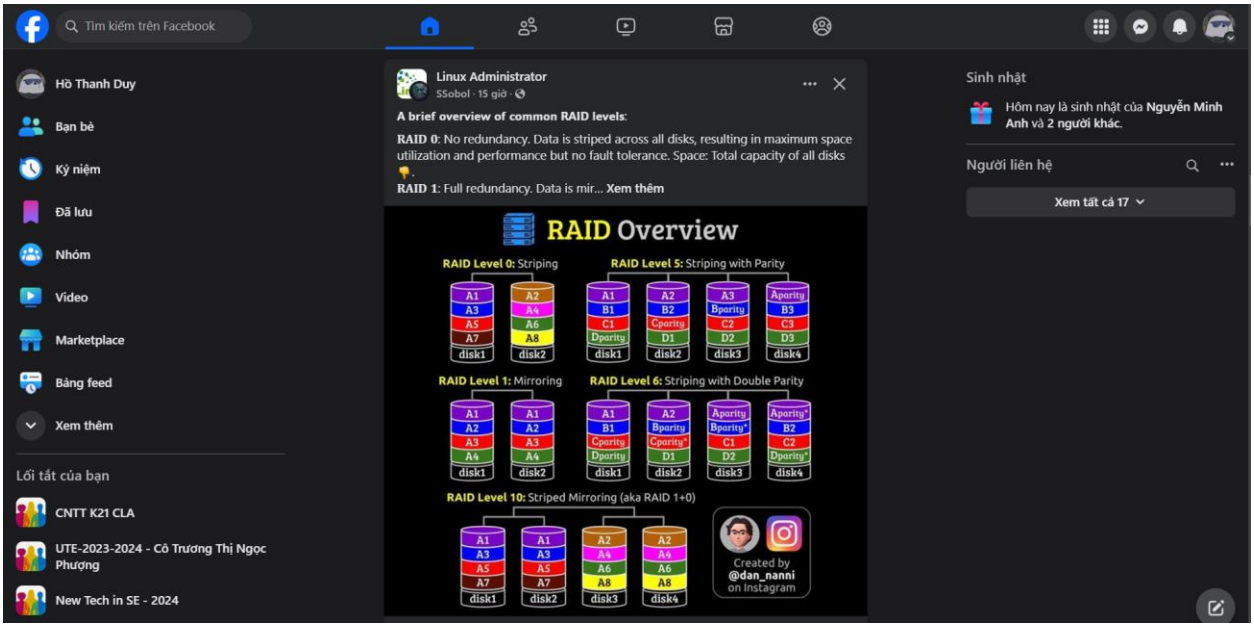


Figure 3-1. Survey Faccebook - Social Media Application

Advantages	- Large user base enables broad social and business networking. - Offers versatile features like groups, marketplace, and events for diverse purposes. - Robust advertising platform with advanced targeting options for businesses. - Easy to use, connecting people across the globe with intuitive tools.
Disadvantages	- Concerns over privacy and data security due to past breaches. - Increasing spread of misinformation and unregulated content. - Declining popularity among younger users, who prefer other platforms. - Algorithmic feeds can encourage excessive screen time and decrease productivity.

Table 3-1. Advantages and Disadvantages of Facebook Social Media Application

3.1.2. REDDIT

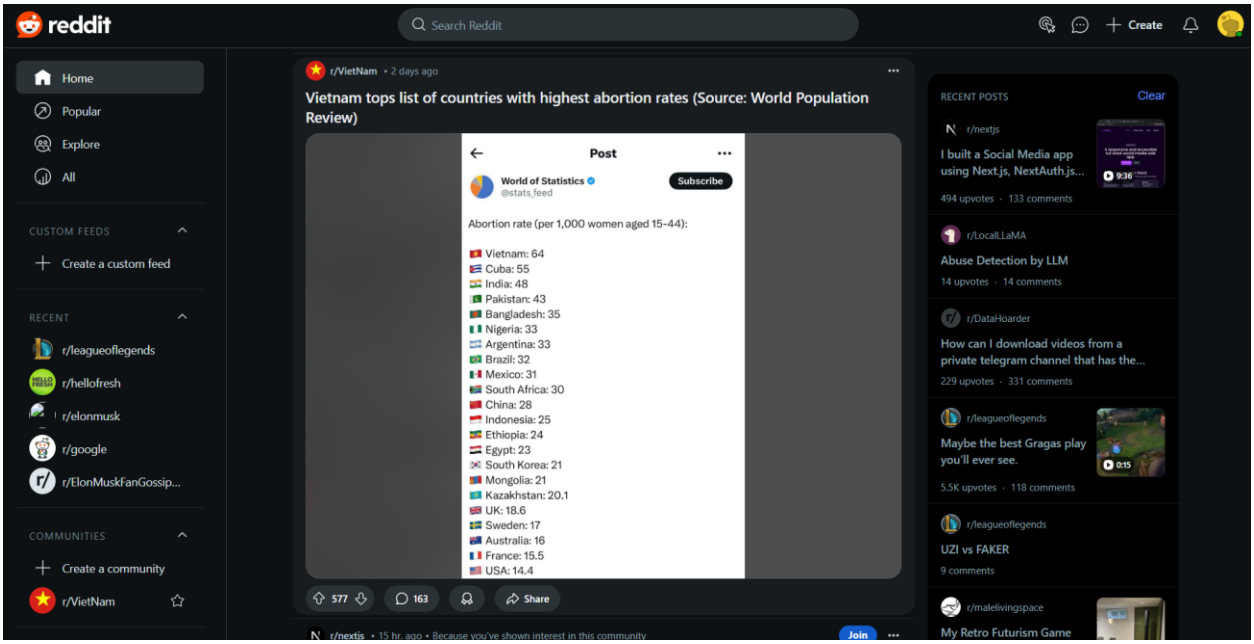


Figure 3-2. Survey Reddit - Social Media Application

Advantages	- Community-driven platform allows for in-depth discussions on niche topics through subreddits. - Users can remain anonymous, providing privacy and freedom of expression. - Wide variety of content, from entertainment to educational resources, curated by users. - Transparent, user-driven moderation fosters a sense of community and inclusiveness.
Disadvantages	- Decentralized moderation can lead to inconsistency in quality and handling of inappropriate content. - Some subreddits may foster toxic environments or polarizing debates. - Interface and culture can be challenging for new users to adapt to.

	- Quality of content is inconsistent, as it heavily depends on user contributions.

Table 3-2. Advantages and Disadvantages of Reddit Social Media Application

3.1.3. LINKEDIN

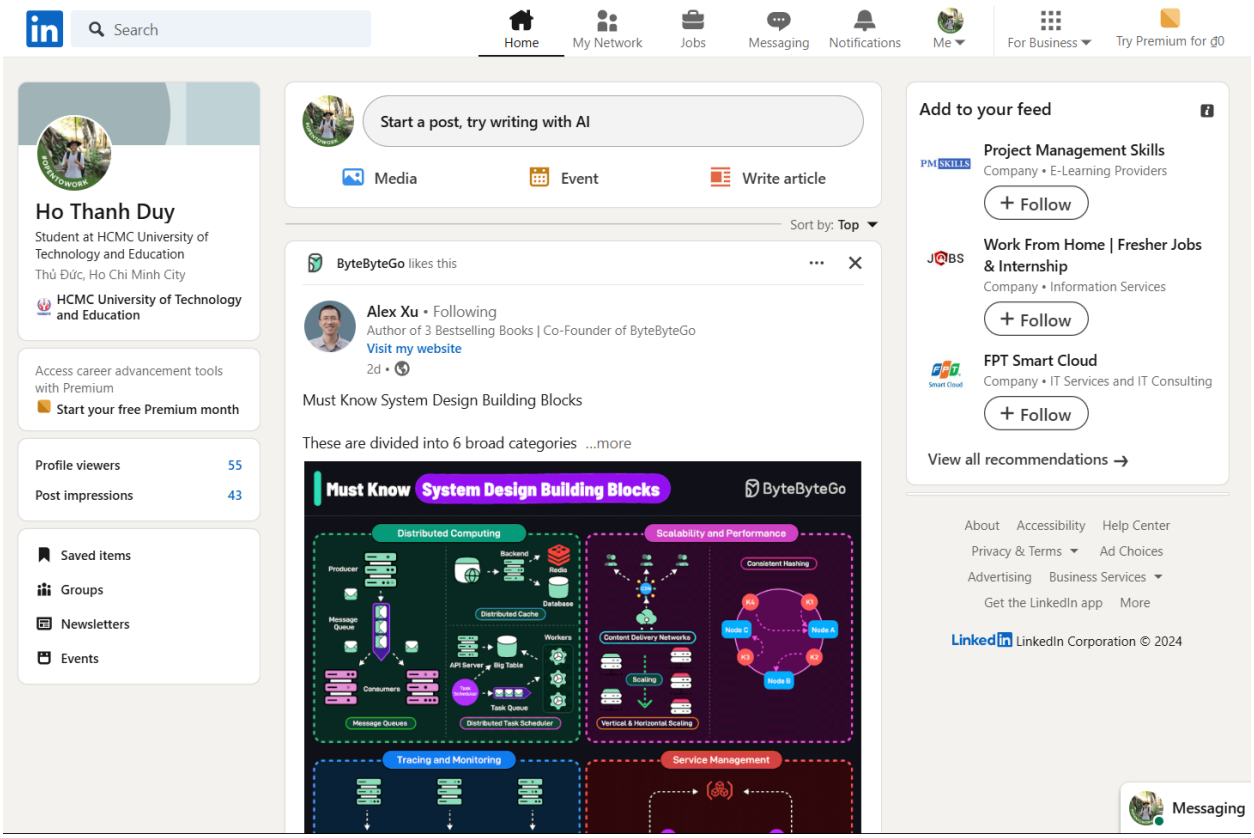


Figure 3-3. Survey Linkedin - Social Media Application

Advantages	- Focused on professional networking, making it ideal for career growth and job hunting. - Provides a platform to showcase professional skills, achievements, and portfolios. - Valuable resource for companies to recruit talent and generate leads. - Analytics tools help users measure their reach and engagement effectively.

Disadvantages	- Many advanced features, such as InMail and analytics, require costly premium subscriptions. - Can feel overly formal and lacks casual interaction compared to other platforms. - High competition on job postings can make it harder for applicants to stand out. - Users may encounter spam or excessive solicitation from recruiters or salespeople.

Table 3-3. Advantages and Disadvantages of Linkedin Social Media Application

3.2. Requirements capturing

3.2.1. List of actions

Management	Action
Account	- Login with authentication
Post	- Create post - Attach files - Edit post - Set post's status - Tag Friends - Delete post - Reaction/ Comment/ Share post
Media Resources	- Upload Media - View/ Download Media
Reporting	- View User Activity Reports - Manage User Accounts - Monitor Content - Approve/Reject Posts - Ban/Unban Users

	- View Admin Actions History - Assign Admin Roles
Relationship	- Send Friend Requests - Manage Friends List
Conversation	- Send Messages - File Sharing in Chat - Online Status
Profile	- View Profile - Edit Profile Information - Upload Profile Picture - Update Contact Information - View Friends/Followers List - Set Profile Visibility - Profile Activity Log - Deactivate Account
Notifications	- Post Notifications - Message Notifications
Feed and search	- View Posts in News Feed - Filter News Feed Content - Search Users

Table 3-4. List of actions from requirements capturing

Type	Detail Requirement's Type
Storage	Storing or persisting data in a database or other storage system.
Lookup	Fetching data from a predefined set of values or reference data (e.g., drop-downs, lists).
Search	Querying data from the system with filters or search criteria.

Output	Displaying data, such as reports or data views, to users.

Table 3-5. Detail Requirement's Type of Requirements

3.2.2. Business functional

Division: Manage Account **ID:**

No.	Function	Type	Constrains/ Formula code	Form code	Description
1	Login	Search	Only username & password with authentication from system can login		Check authentication user account and login application

Table 3-6. Manage Account Functional

Division: Manage Post **ID:**

No.	Function	Type	Constrains/ Formula code	Form code	Description
1	Create post	Storage			Users can create posts with text, images, videos, or links.
2	Attach files	Storage			Users can attach files (images, videos, documents) to posts.
3	Edit post	Storage			Users can edit their posts after publishing.
4	Set post's status	Storage			Adjusting post's status after publishing

5	Tag Friends	Output			Users can tag friends in posts and photos.
6	Delete post	Storage			Users can delete their own posts.
7	Reaction/ Comment/ Share post	Output			Users can interact with posts through likes, comments, etc.

Table 3-7. Manage Post Functional

Division: Media Resources **ID:**

No.	Function	Type	Constrains/ Formula code	Form code	Description
1	Upload Media	Storage			Users can upload photos and videos to their profiles.
2	View/ Download Media	Output			Users can view and download uploaded media content.

Table 3-8. Media Resources Functional

Division: Manage Reporting **ID:**

No.	Function	Type	Constrains/ Formula code	Form code	Description
1	View User Activity Reports	Output	Admin		Admins can view detailed reports on user activities (posts, messages, interactions).

2	Manage User Accounts	Output	Admin		Admins can deactivate, delete, or reset user accounts.
3	Monitor Content	Output	Admin		Admins can monitor posts, comments, and media for violations.
4	Approve/ Reject Posts	Output	Admin		Admins can approve or reject posts pending moderation.
5	Ban/Unban Users	Output	Admin		Admins can ban or unban users for violating platform policies.
6	View Admin Actions History	Output	Admin		Admins can review the history of actions taken by other admins.
7	Assign Admin Roles	Output	Admin		Admins can assign or remove admin roles to/from other users.

Table 3-9. Manage Reporting Functional

Division: Manage Relationship **ID:**

No.	Function	Type	Constrains/ Formula code	Form code	Description
1	Send Friend Requests	Storage			Users can send requests to other users to become friends.

2	Manage Friends List	Lookup			Users can organize, view, or remove friends from their list.

Table 3-10. Manage Relationship Functional

Division: Manage Conversation **ID:**

No.	Function	Type	Constrains/ Formula code	Form code	Description
1	Send Messages	Storage			Users can send messages with text, images, or videos.
2	File Sharing in Chat	Storage			Users can share files within direct chats.
3	Online Status	Output			Displays the online/offline status of users in real-time.

Table 3-11. Manage Conversation Functional

Division: Manage Profile **ID:**

No.	Function	Type	Constrains/ Formula code	Form code	Description
1	View Profile	Lookup			Users can view their own and others' profiles.
2	Edit Profile Information	Output			Users can update their name, bio, contact info, etc.

3	Upload Profile Picture	Output			Users can upload or change their profile picture.
4	Update Contact Information	Output			Users can update their email address, phone number, etc.
5	View Friends/Followers List	Lookup			Users can view a list of their friends or followers.
6	Set Profile Visibility	Output			Users can set their profile visibility to public or private.
7	Profile Activity Log	Lookup			Users can view a history of their actions and profile changes.
8	Deactivate Account	Output			Users can temporarily deactivate their account.

Table 3-12. Manage Profile Functional

Division: Notifications **ID:**

No.	Function	Type	Constrains/ Formula code	Form code	Description
1	Post Notifications	Output			Notifications for posts being liked, shared, or comment.

2	Message Notifications	Output			Alerts for new messages received in direct chats.

Table 3-13. Notifications Functional

Division: Feed and Search **ID:**

No.	Function	Type	Constrains/ Formula code	Form code	Description
1	View Posts in News Feed	Output			Displays posts from friends in a user's feed.
2	Filter News Feed Content	Lookup			Users can filter posts by type (images, videos, text).
3	Search Users	Search			Users can search for others by name or email.

Table 3-14. Feed and Search Functional

3.2.3. System functional

No.	Content	Criteria	Description
1	User Authentication	Authentication	Only account has authentication with system can login and use various function of Social Network application
2	User permission	Permission	User: manage profile, sending media resource through post (upload and download from personal's storage), post operation like reaction, new post, … Admin: retrieve all features and function like user as well as manage all report from

			application, manage user account, application's resouces .
3	Local Area Network	Environment	System has to build in website and mobile application (multi platform)

Table 3-15. System Functional

3.2.4. Non-functional

No.	Content	Criteria	Description
1	The software interface is user-friendly, easy to operate	Usability	Interface is easy to use and can be used in all devices.
2	The software is across various machines, allowing for information sharing	Compatibility	The information can be shared for others in other devices at the same time
3	The database is securely stored and easily retrievable	Efficiency	Securing the information and easy to retrieve in the database

Table 3-16. Non-functional

3.3. Requirements modeling

3.3.1. Use case diagrams

3.3.1.1. User management use case diagram

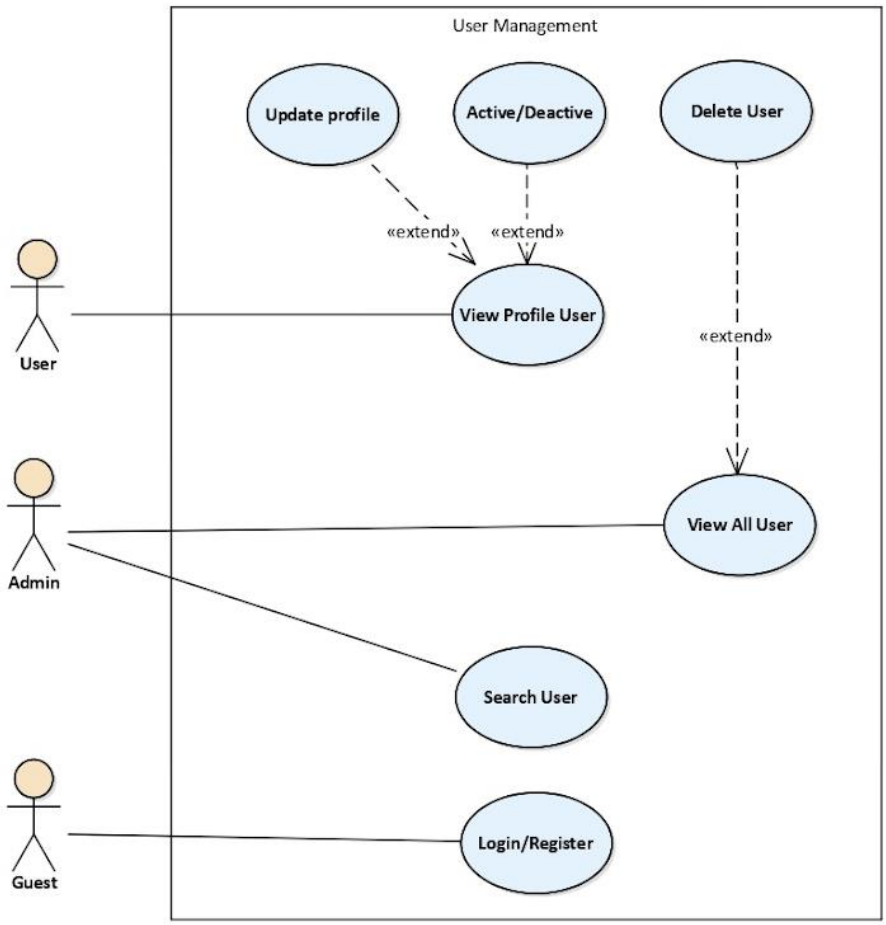


Figure 3-4. User management use case diagram

3.3.1.2. Media management use case diagram

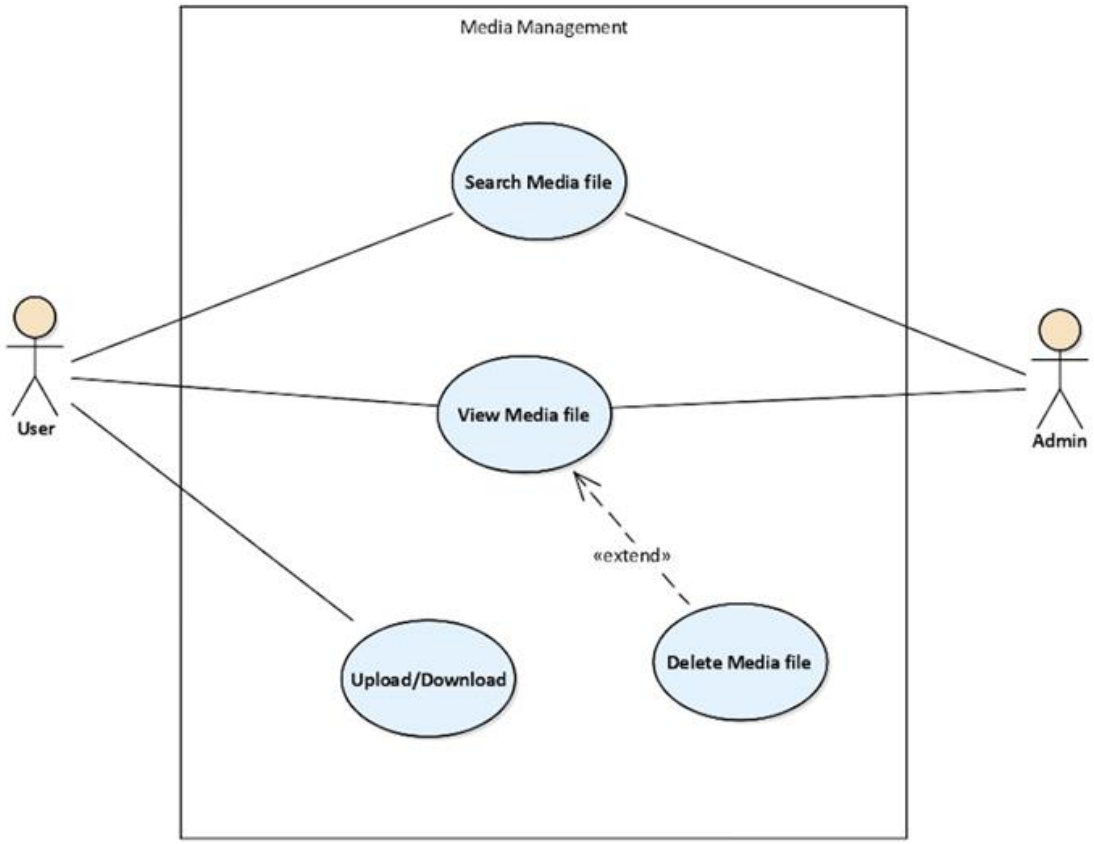


Figure 3-5. Media management use case diagram

3.3.1.3. Post management use case diagram

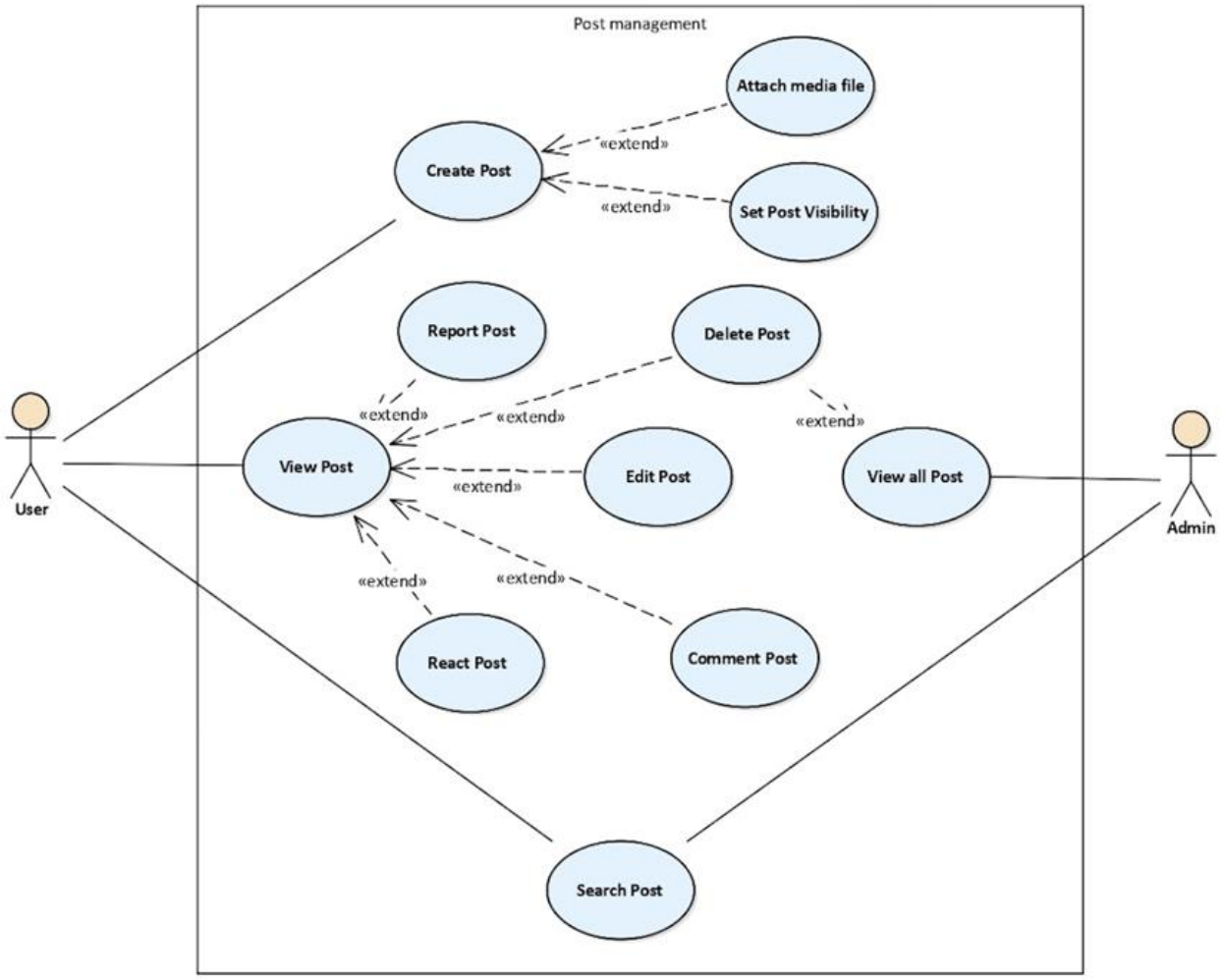


Figure 3-6. Post management use case diagram

3.3.1.4. Friend management use case diagram

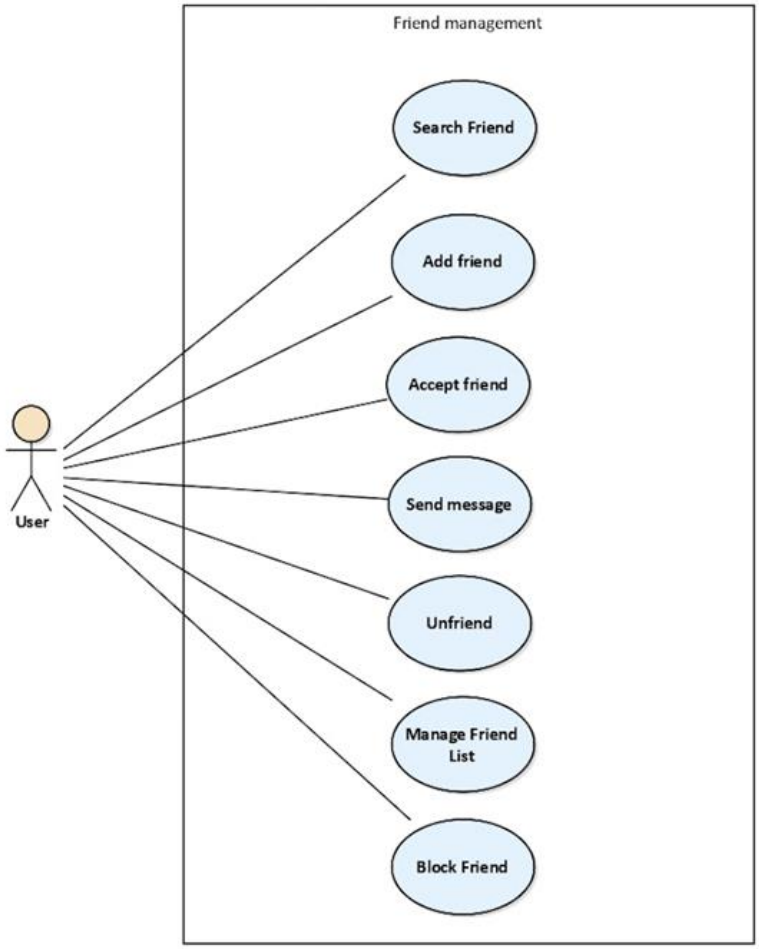


Figure 3-7. Friend management use case diagram

3.3.2. Use case specifications

3.3.2.1. Use case Register (UC-1)

Use case ID	UC-1
Use Case Name	Register
Short Description	Allows a user or guest to register a new account on the platform.
Actor	User, Guest
Pre-conditions	None
Post-Conditions	Actor registers account successfully

Main Flow	1. Actor click 'Register' button 2. The system returns register page. 3. Actor fills all input box (account register & profile) 4. The system authenticates email, password and notifies "Registration Success" and returns login page.

Table 3-17. Use case Register (UC-1)

3.3.2.2. Use case Login (UC-2)

Use case ID	UC-2
Use Case Name	Login
Short Description	Allows a user to log in to their existing account.
Actor	User
Pre-conditions	The user must have an existing account.
Post-Conditions	The user is successfully logged in and directed to the dashboard or home page.
Main Flow	1. Actor clicks the 'Login' button. 2. The system displays the login form. 3. Actor enters the email and password. 4. The system validates the credentials. 5. If valid, the system authenticates the user and redirects to the dashboard or home page. 6. If invalid, the system displays an error message.

Table 3-18. Use case Login (UC-2)

3.3.2.3. Use case Change password (UC-3)

Use case ID	UC-3
Use Case Name	Change Password
Short Description	Allows a user to change their password after logging in.
Actor	User

Pre-conditions	The user must be logged in and have access to the account settings.
Post-Conditions	The user's password is successfully updated.
Main Flow	1. Actor clicks on the 'Change Password' option. 2. The system prompts the user to enter the current password, new password, and confirm the new password. 3. Actor enters the required information. 4. The system validates the current password and ensures the new password meets the required criteria. 5. If valid, the system updates the password and confirms the change. 6. If invalid, the system displays an error message (e.g., incorrect current password, password mismatch,…)

Table 3-19. Use case Change password (UC-3)

3.3.2.4. Use case Log out (UC-4)

Use case ID	UC-4
Use Case Name	Log Out
Short Description	Allows a logged-in user to log out of their account.
Actor	User
Pre-conditions	The user must be logged in to their account.
Post-Conditions	The user is logged out and redirected to the login page.
Main Flow	1. Actor clicks the 'Log Out' button or selects 'Log Out' from the account menu. 2. The system logs out the user, clearing the session or authentication token. 3. The system redirects the user to the login page.

Table 3-20. Use case Log out (UC-4)

3.3.2.5. Use case Change profile's user (UC-5)

Use case ID	UC-5

Use Case Name	Change Profile's User
Short Description	Allows a user to update their profile information (e.g., name, profile picture, bio).
Actor	User
Pre-conditions	The user must be logged in and have access to their profile settings.
Post-Conditions	The user's profile is successfully updated with the new information.
Main Flow	1. Actor navigates to the 'Admin page' and choice tab 'User' 2. Actor clicks on the 'Edit' button to modify profile details. 3. The system displays editable fields (e.g., name, profile picture, bio). 4. Actor updates the required information. 5. The system validates the input (e.g., ensures profile picture is a valid file format, name is not empty). 6. If valid, the system updates the profile and shows a confirmation message. 7. If invalid, the system displays an error message (e.g., invalid file format or empty fields).

Table 3-21. Use case Change profile's user (UC-5)

3.3.2.6. Use case Create/Edit/Delete Post (UC-6)

Use case ID	UC-6
Use Case Name	Create Post
Short Description	Allows a user to post a new post or delete an existing post.
Actor	User
Pre-conditions	The user must be logged in and have access to the posting or their own posts.
Post-Conditions	The post is successfully created or deleted.

Main Flow	1. Actor click on box create new post. 2. Actor fills in the content for the post (e.g., text, image, or link). 3. System upload all images to cloudinary 4. If valid, the system creates the post and displays a success message. 5. The post is now visible in the user's timeline or feed. 7. If invalid, the system displays an error message (e.g., invalid file format or empty fields).

Table 3-22. Use case Create Post (UC-6)

Use case ID	UC-6
Use Case Name	Edit Post
Short Description	Allows a user to edit an existing post.
Actor	User
Pre-conditions	The user must be logged in and have access to the posting or their own posts.
Post-Conditions	The post is successfully edited.
Main Flow	1. Actor click on post that user want to edit. 2. Dalog appear and allowing user edit. 3. If valid, the system edit content of the post and displays a success message. 4. The post is now appears in the user's timeline or feed with new content. 5. If invalid, the system displays an error message.

Table 3-23. Use case Edit Post (UC-6)

Use case ID	UC-6
Use Case Name	Delete Post
Short Description	Allows a user to delete an existing post.

Actor	User
Pre-conditions	The user must be logged in and have access to the posting or their own posts.
Post-Conditions	The post is successfully deleted.
Main Flow	1. Actor click on post that user want to remove. 2. Dialog confirm that user delete post 3. If valid, the system delete the post and displays a success message. 4. The post is now disappear in the user's timeline or feed. 5. If invalid, the system displays an error message.

Table 3-24. Use case Delete Post (UC-6)

3.3.2.7. Use case Comment Post (UC-7)

Use case ID	UC-7
Use Case Name	Comment Post
Short Description	Allows a user to comment on a post.
Actor	User
Pre-conditions	The user must be logged in and the post must exist.
Post-Conditions	The comment is successfully added to the post.
Main Flow	1. Actor navigates to the post they wish to comment on. 2. Actor clicks the ‘Comment’ button. 3. The system displays a comment input box. 4. Actor enters the comment and submits it. 5. If valid, the system adds the comment to the post and displays it. 6. If invalid, the system displays an error message.

Table 3-25. Use case Comment Post (UC-7)

3.3.2.8. Use case Reaction Post (UC-8)

Use case ID	UC-8
Use Case Name	Reaction Post

Short Description	Allows a user to react (like, love, etc.) to a post.
Actor	User
Pre-conditions	The post receives a new reaction from the user.
Post-Conditions	The post receives a new reaction from the user.
Main Flow	1. Actor navigates to the post they wish to react to. 2. Actor selects a reaction (e.g., like, love, etc.). 3. The system validates the reaction. 4. If valid, the system records the reaction and updates the post's reaction count. 5. If invalid, the system displays an error message.

Table 3-26. Use case Reaction Post (UC-8)

3.3.2.9. Use case In-Active account user (UC-9)

Use case ID	UC-9
Use Case Name	In-Active Account User
Short Description	Allows the system to mark an account as inactive if the user has not logged in for a period of time.
Actor	Admin
Pre-conditions	The user has not logged in for a specified period of time.
Post-Conditions	The account is marked as inactive and the user may not be able to perform actions.
Main Flow	1. The system checks the last login time of the user. 2. If the user has been active for the specified period, the system marks the account as inactive. 3. The system notifies the user of the account status change.

Table 3-27. Use case In-Active account user (UC-9)

3.3.2.10. Use case Delete account user (UC-10)

Use case ID	UC-10
Use Case Name	Delete Account User
Short Description	Allows a user to permanently delete their account.

Actor	Admin
Pre-conditions	The user must be logged in and have access to admin settings.
Post-Conditions	The user's account is deleted from the system.
Main Flow	1. Actor navigates to the 'Admin Settings' page and choice 'User' tab. 2. Actor selects the 'Delete Account' icon option. 3. The system prompts the user to confirm the deletion. 4. Actor confirms the deletion. 5. The system permanently deletes the user's account and all associated data.

Table 3-28. Use case Delete account user (UC-10)

3.3.2.11. Use case Send Message (Chatting) (UC-11)

Use case ID	UC-11
Use Case Name	Send Message
Short Description	Allows a user to send a message to another user.
Actor	User
Pre-conditions	The user must be logged in and have access to the messaging feature.
Post-Conditions	The message is successfully sent to the recipient.
Main Flow	1. Actor navigates to the messaging interface. 2. Actor selects the user recipient. 3. Actor types the message. 4. Actor sends the message. 5. If valid, the message is sent and appears in the recipient's inbox. 6. If invalid, the system displays an error message.

Table 3-29. Use case Send Message (Chatting) (UC-11)

3.3.2.12. Use case Follow/Unfollow user (UC-12)

Use case ID	UC-12

Use Case Name	Follow/Unfollow User
Short Description	Allows a user to follow or unfollow another user's profile.
Actor	User
Pre-conditions	The user must be logged in and the other user's profile must exist.
Post-Conditions	The user is now following or no longer following the other user.
Main Flow	1. Actor navigates to the friend page they wish to follow/unfollow. 2. Actor clicks the 'Follow' or 'Unfollow' button. 3. The system validates the action. 4. If valid, the system updates the following status and displays a success message. 5. If invalid, the system displays an error message.

Table 3-30. Use case Follow/Unfollow user (UC-12)

3.3.2.13. Use case Save Post Items (UC-13)

Use case ID	UC-13
Use Case Name	Save Post Items
Short Description	Allows a user to save a post for future reference (bookmark or save to a collection).
Actor	User
Pre-conditions	The user must be logged in and the post must exist.
Post-Conditions	The post is saved to the default folder of user
Main Flow	1. Actor navigates to the post they wish to save. 2. Actor clicks on the 'Save items' button. 3. The system saves the post to default folder of user. 4. The system shows a confirmation message indicating the post has been saved.

	5. The post can be accessed from the user's saved items in the future.

Table 3-31. Use case Save Post Items (UC-13)

3.3.2.14. Use case Change folder for post (UC-14)

Use case ID	UC-14
Use Case Name	Change Folder for Post
Short Description	Allows a user to move a post to a different folder or collection for organization.
Actor	User
Pre-conditions	The user must be logged in, and the post must be in a folder.
Post-Conditions	The post is successfully moved to the new folder.
Main Flow	1. Navigate to any existing folder of user 2. Actor selects the post and clicks on the 'Move' button. 3. The system validates the action. 4. If valid, the system moves the post to the selected folder and shows a confirmation message. 5. If invalid, the system displays an error message (e.g., folder not available).

Table 3-32. Use case Change folder for post (UC-14)

3.3.2.15. Use case Create folder for post (UC-15)

Use case ID	UC-15
Use Case Name	Create Folder for Post
Short Description	Allows a user to create a new folder to organize posts.
Actor	User
Pre-conditions	The user must be logged in and have access to the post management feature.
Post-Conditions	A new folder is created and available to store posts.
Main Flow	1. Actor navigates to the post management or folder organization page.

	2. Actor clicks on the ‘Create New Folder’ button. 3. The system prompts the user to enter a name for the new folder. 4. Actor enters the folder name and clicks ‘Create’. 5. The system validates the folder name (e.g., checks for duplicates or invalid characters). 6. If valid, the system creates the new folder and displays it in the list of folders. 7. If invalid, the system displays an error message and prompts the user to re-enter the folder name.

Table 3-33. Use case Create folder for post (UC-15)

CHAPTER 4. SYSTEM DESIGN

4.1. Sequence Diagrams

4.1.1. Login

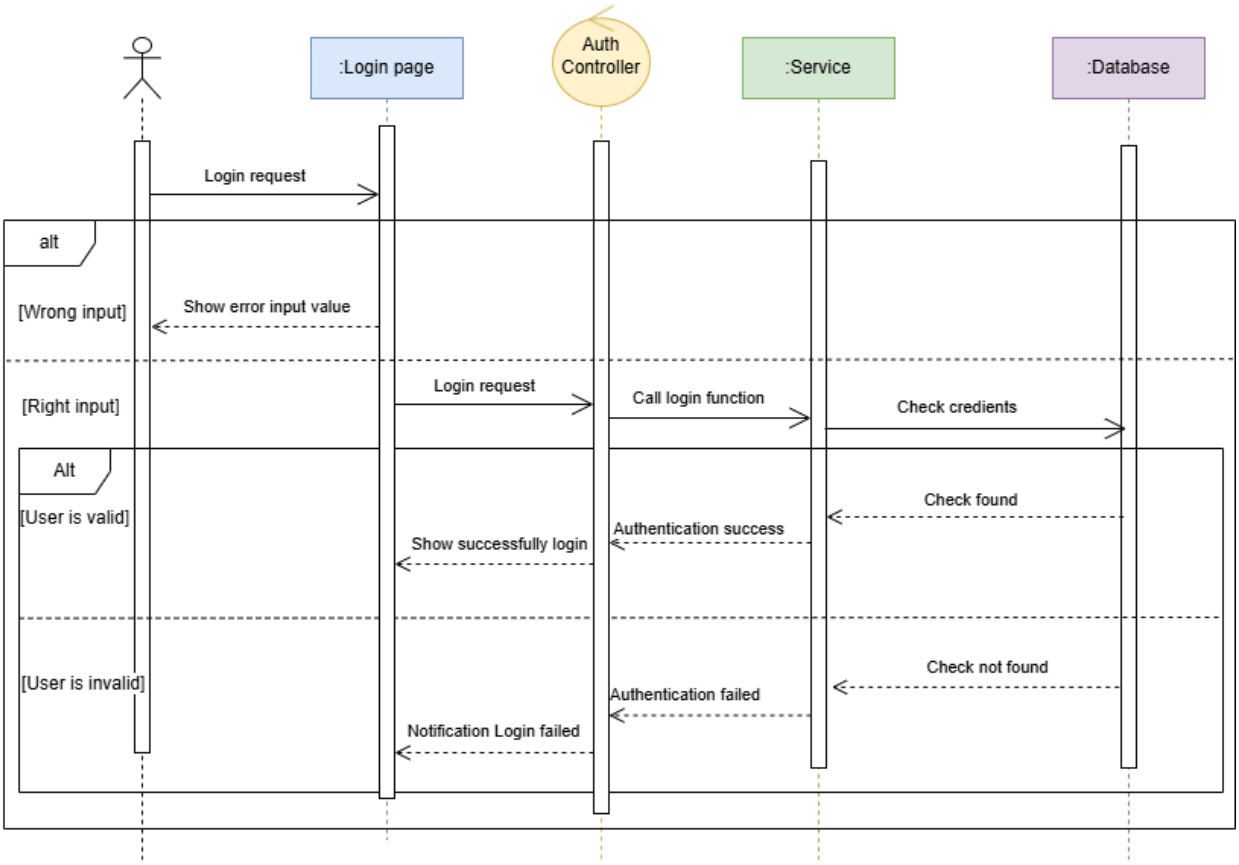


Figure 4-1. Login Sequence Diagram

4.1.2. Register

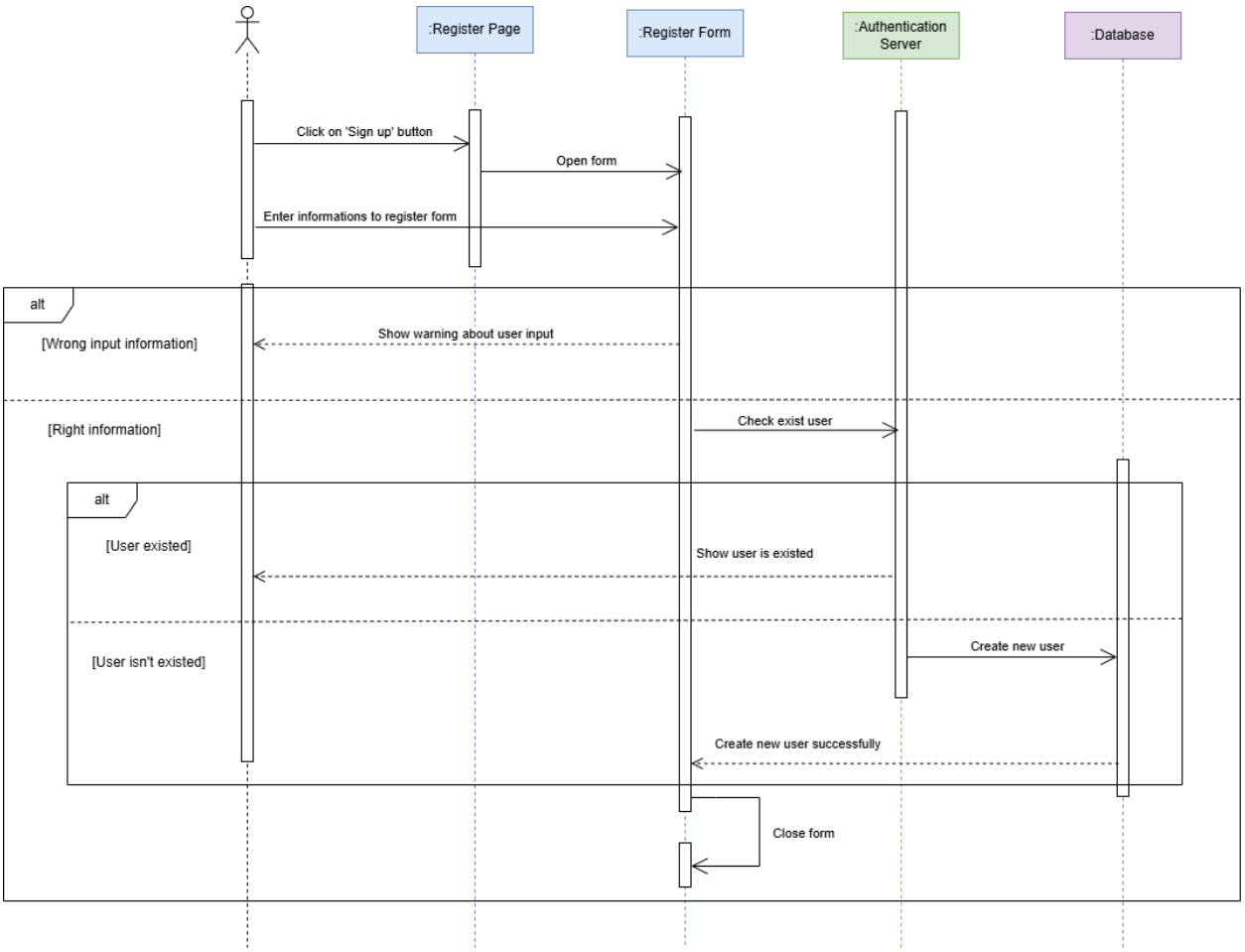


Figure 4-2. Register Sequence Diagram

4.1.3. Create New Profile

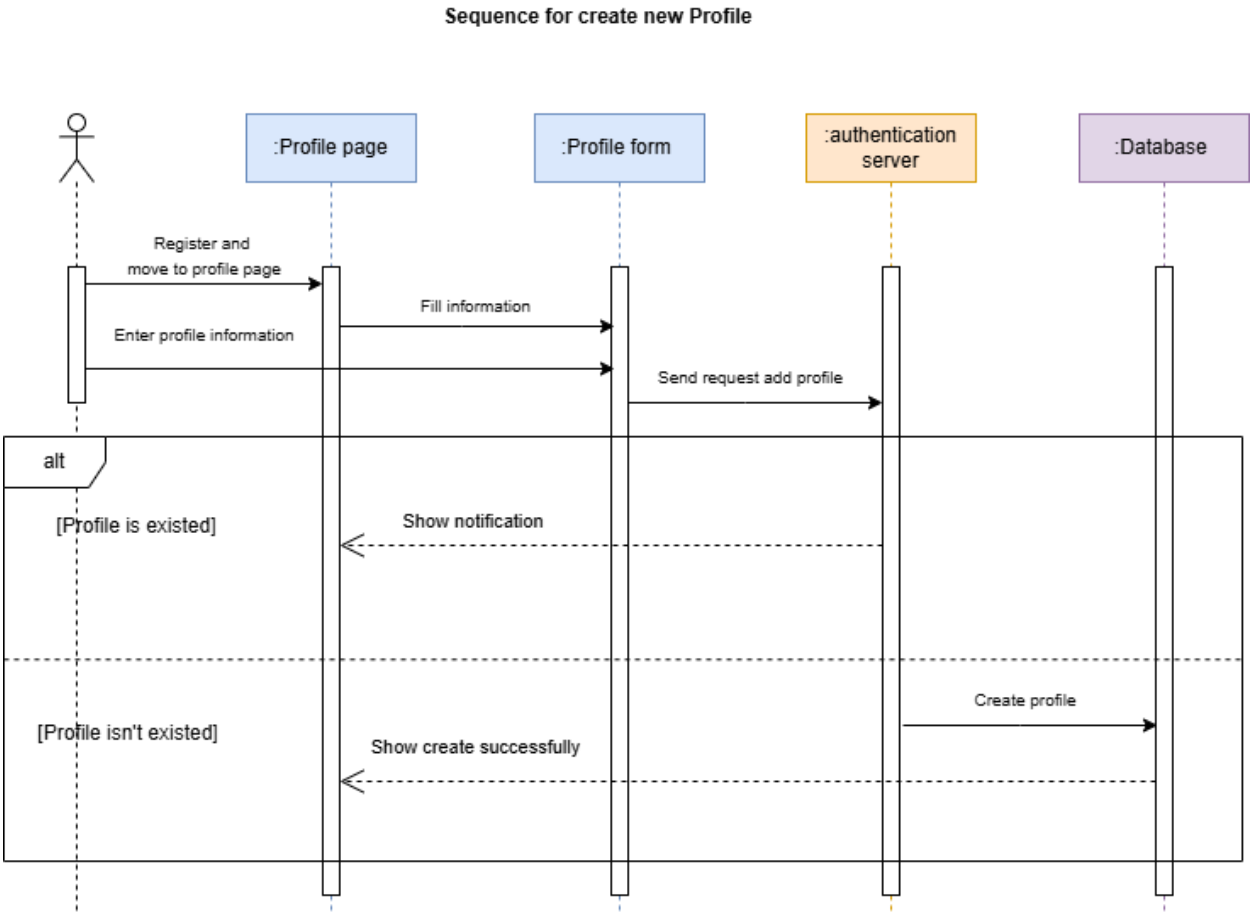


Figure 4-3. Create New Profile Sequence Diagram

4.1.4. Change Profile's Information

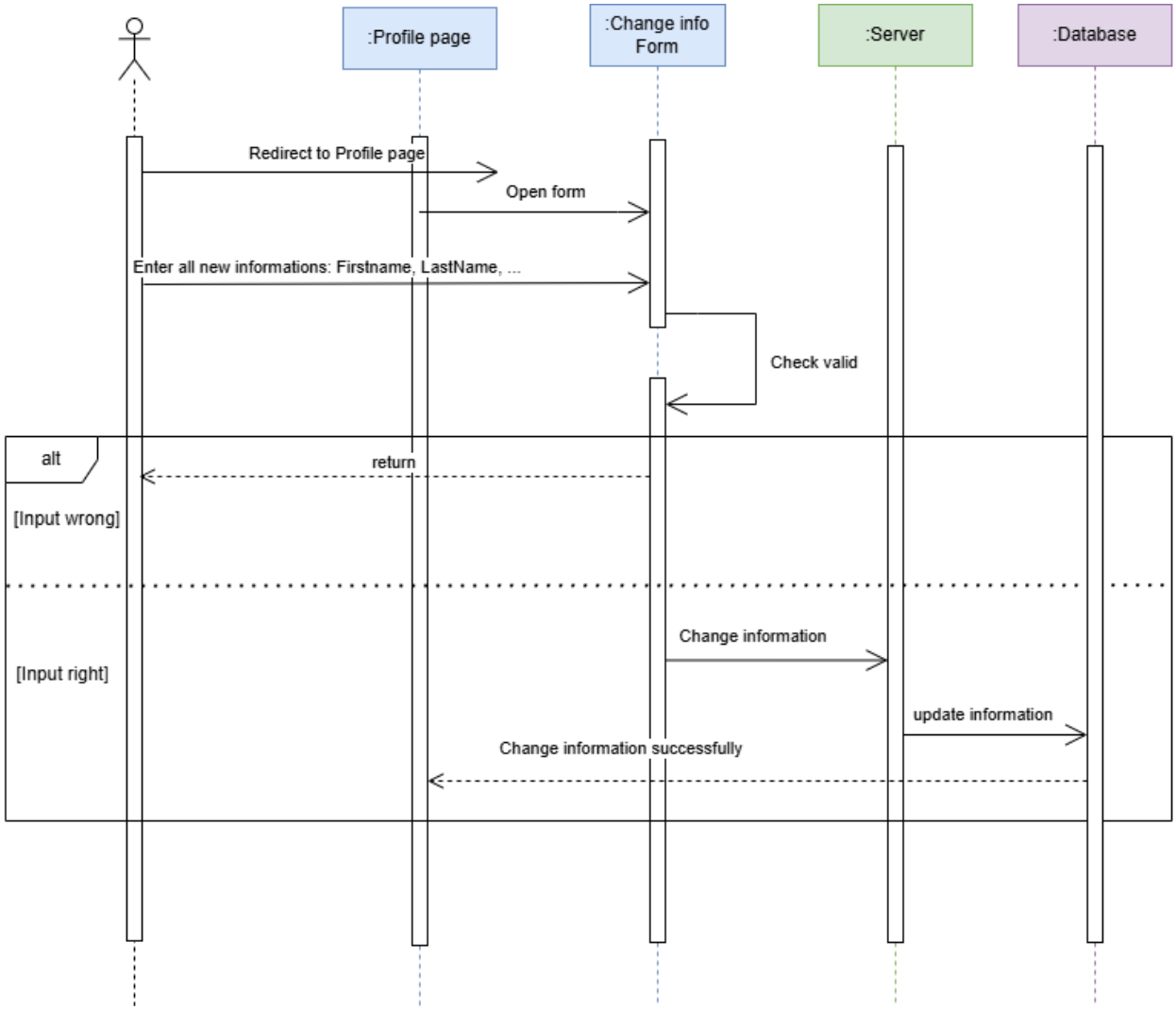


Figure 4-4. Change Profile Information Sequence Diagram

4.1.5. Change Password

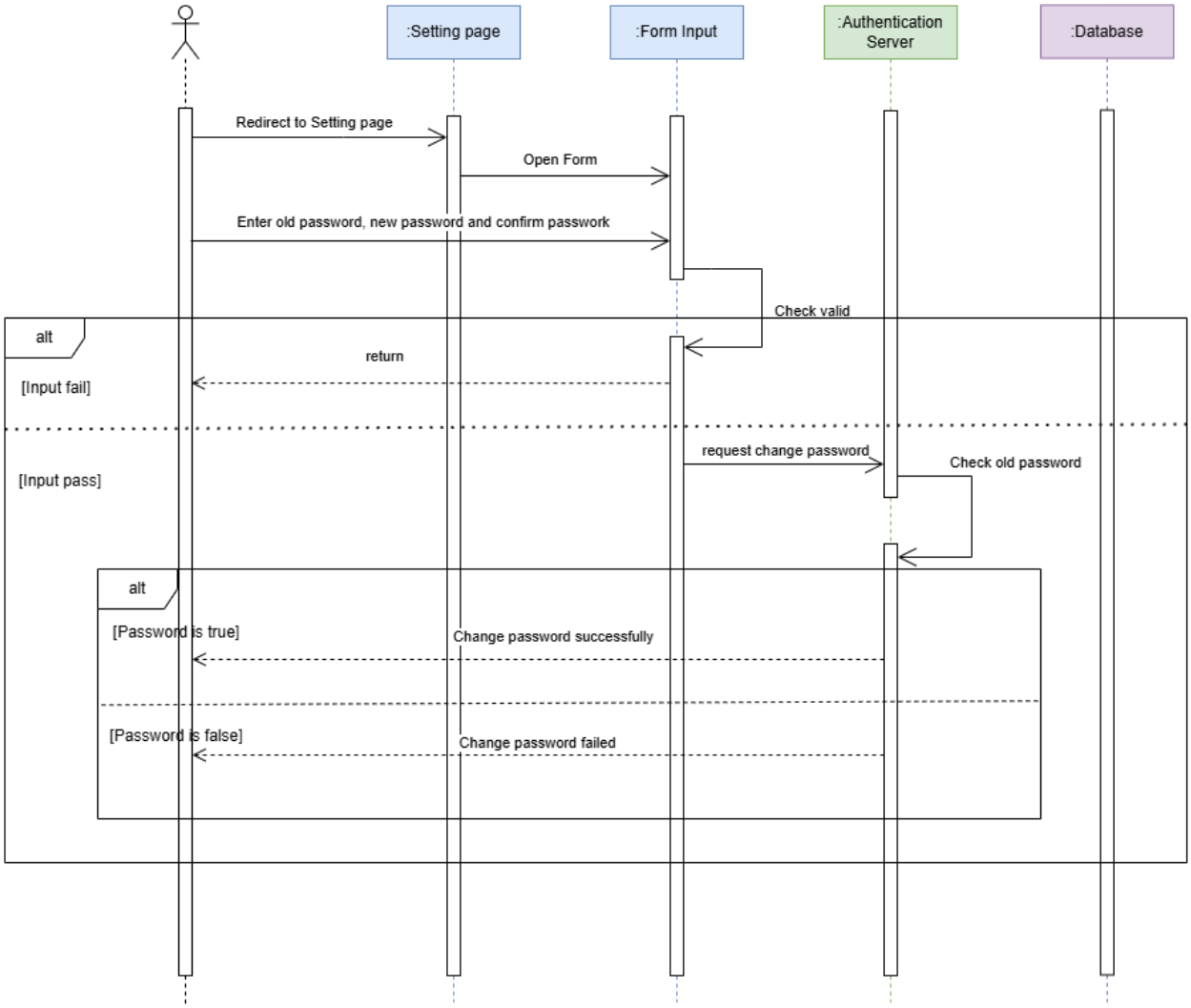


Figure 4-5. Change Password Sequence Diagram

4.1.6. Follow User

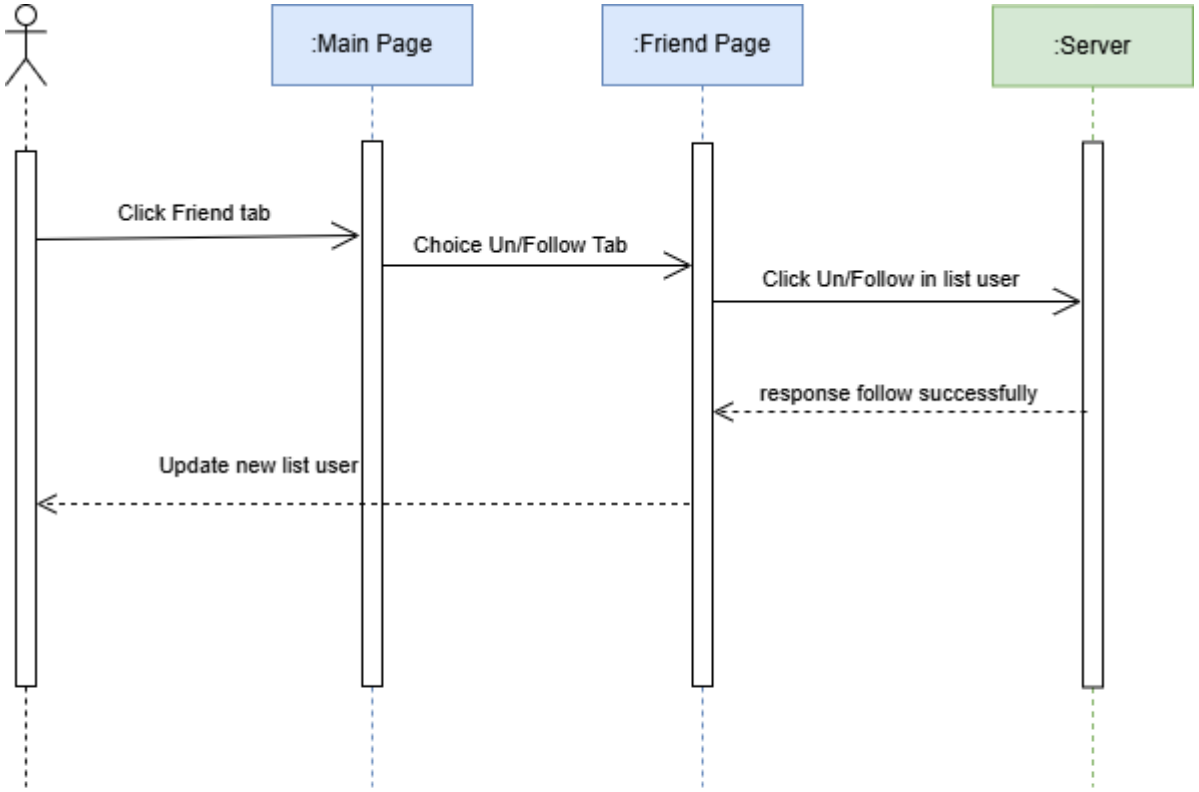


Figure 4-6. Follow User Sequence Diagram

4.1.7. Create New Post

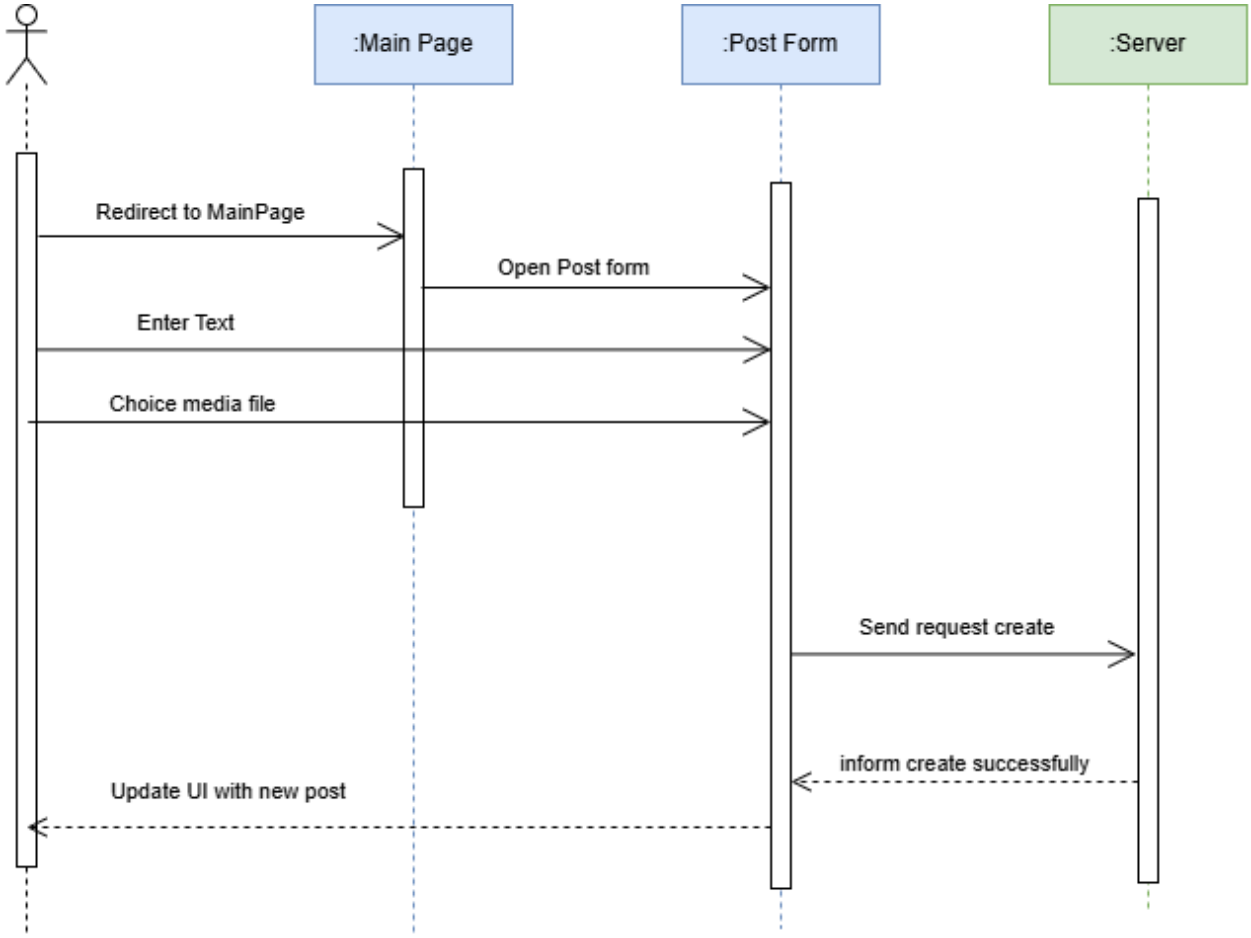


Figure 4-7. Create New Post Sequence Diagram

4.1.8. Edit Post

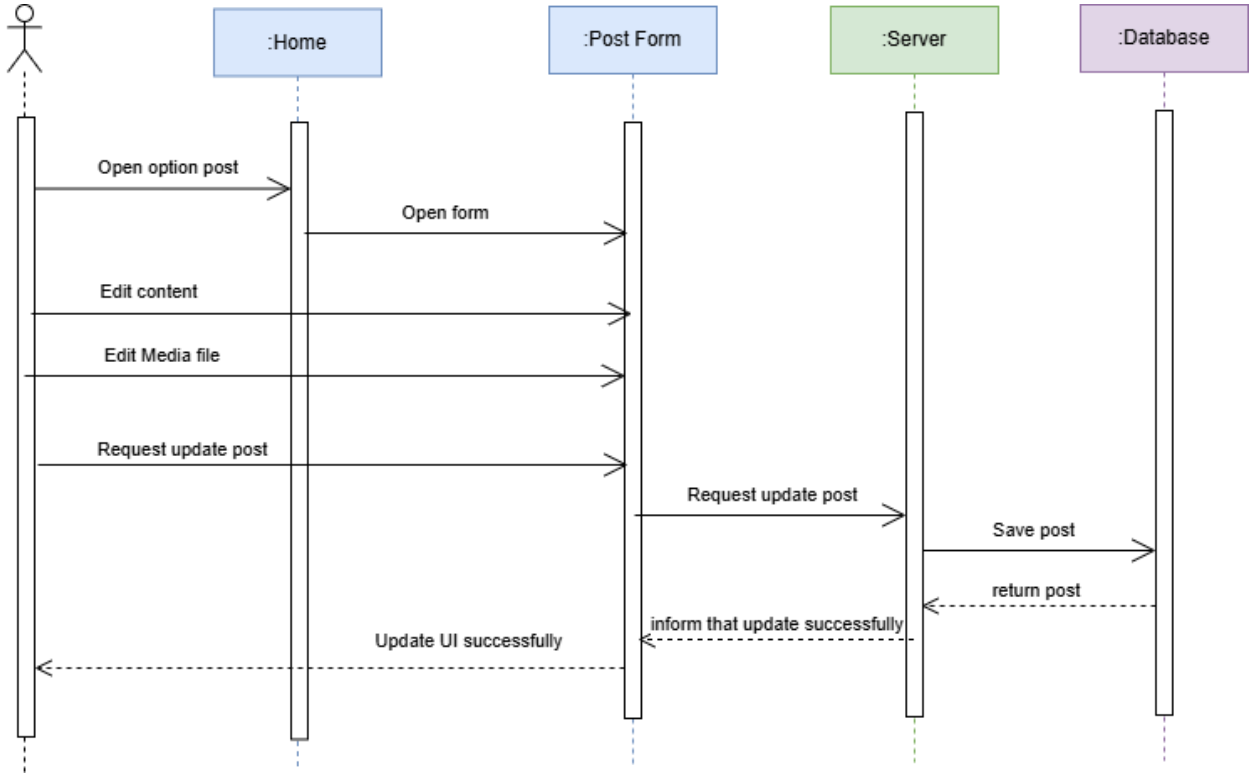


Figure 4-8. Edit Post Sequence Diagram

4.1.9. Reaction

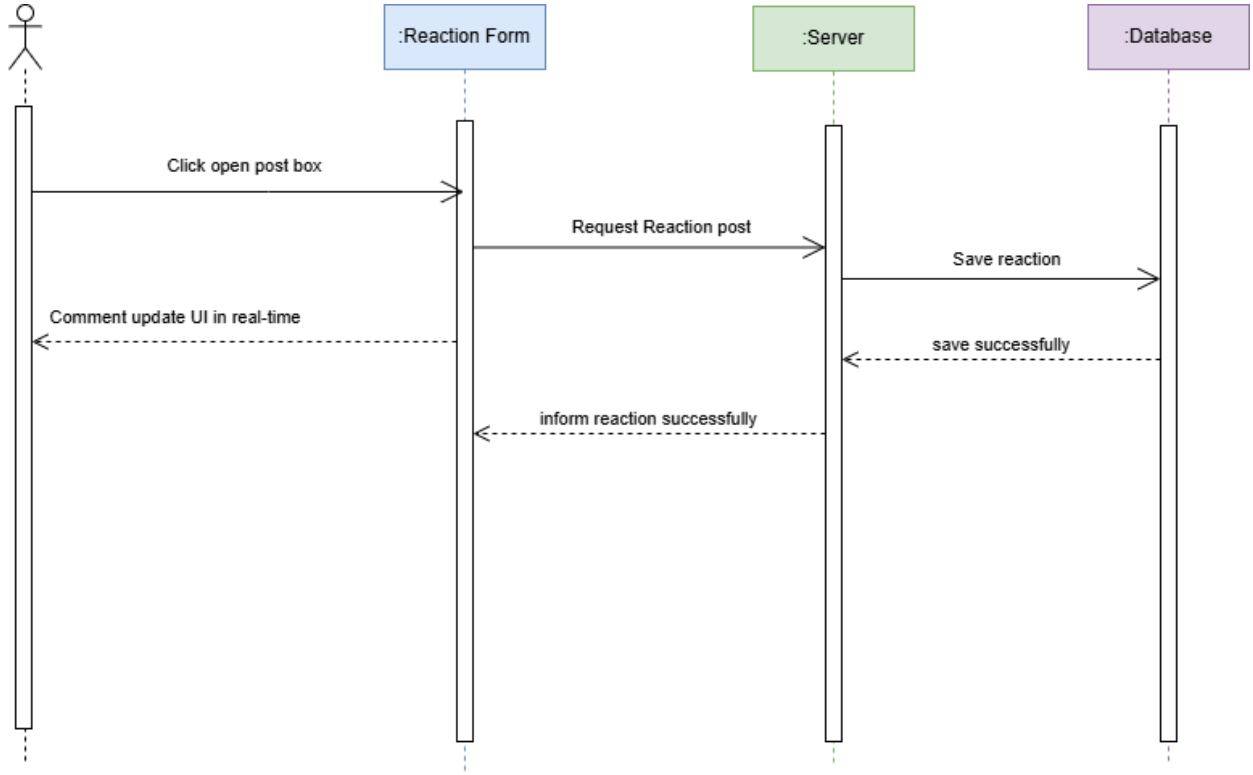


Figure 4-9. Reaction Sequence Diagram

4.1.10. Comment

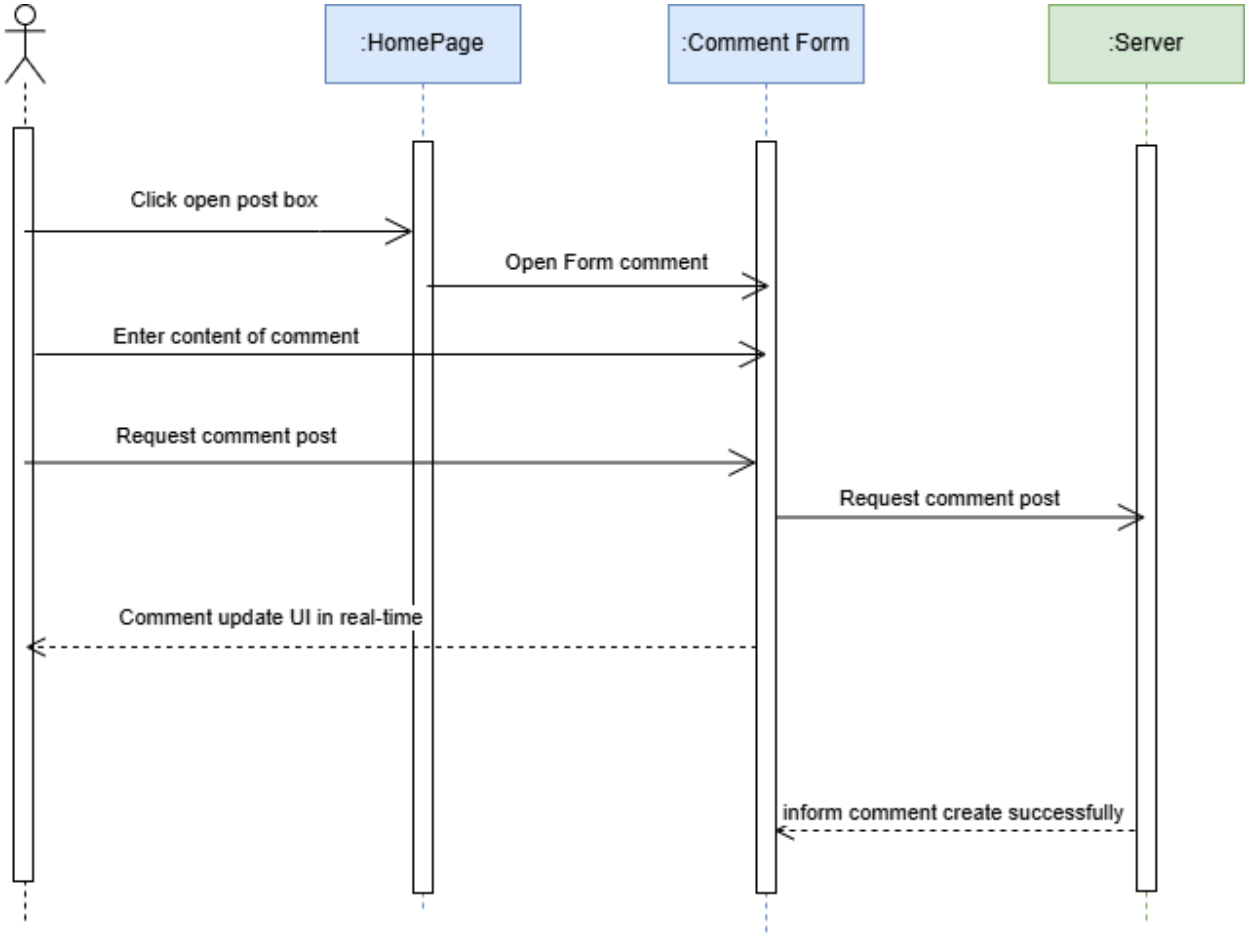


Figure 4-10. Comment Sequence Diagram

4.1.11. Delete Existed Post

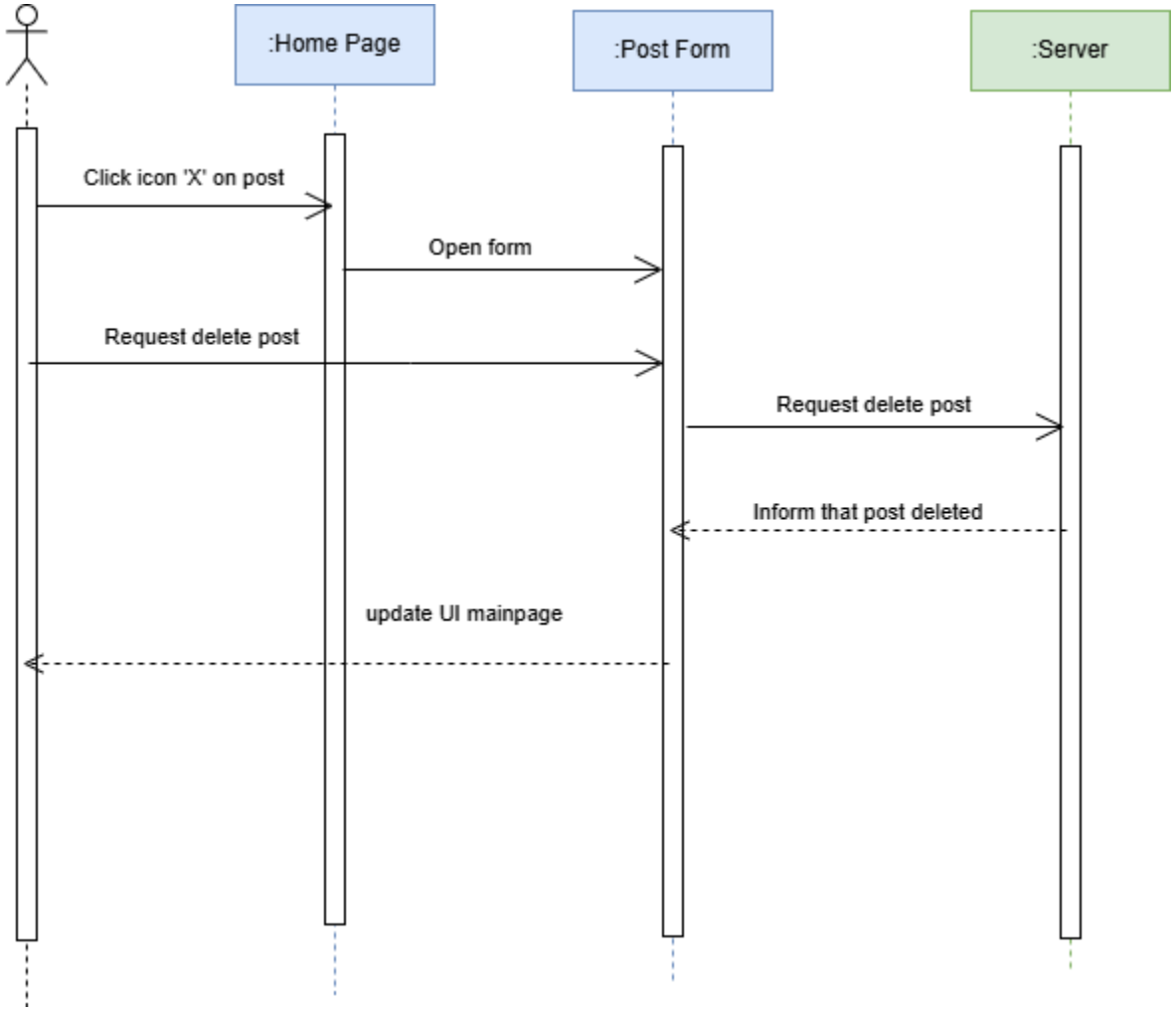


Figure 4-11. Delete Existed Post Sequence Diagram

4.1.12. Add Post To Folder

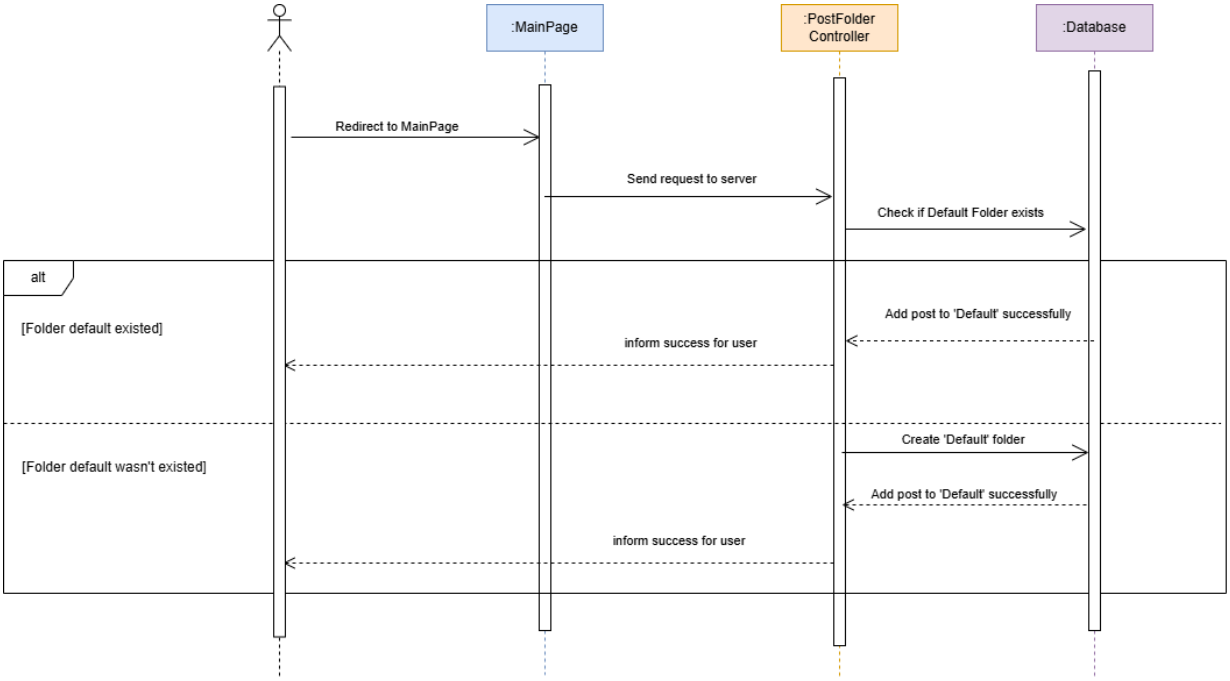


Figure 4-12. Add Post To Folder Sequence Diagram

4.1.13. Create Folder

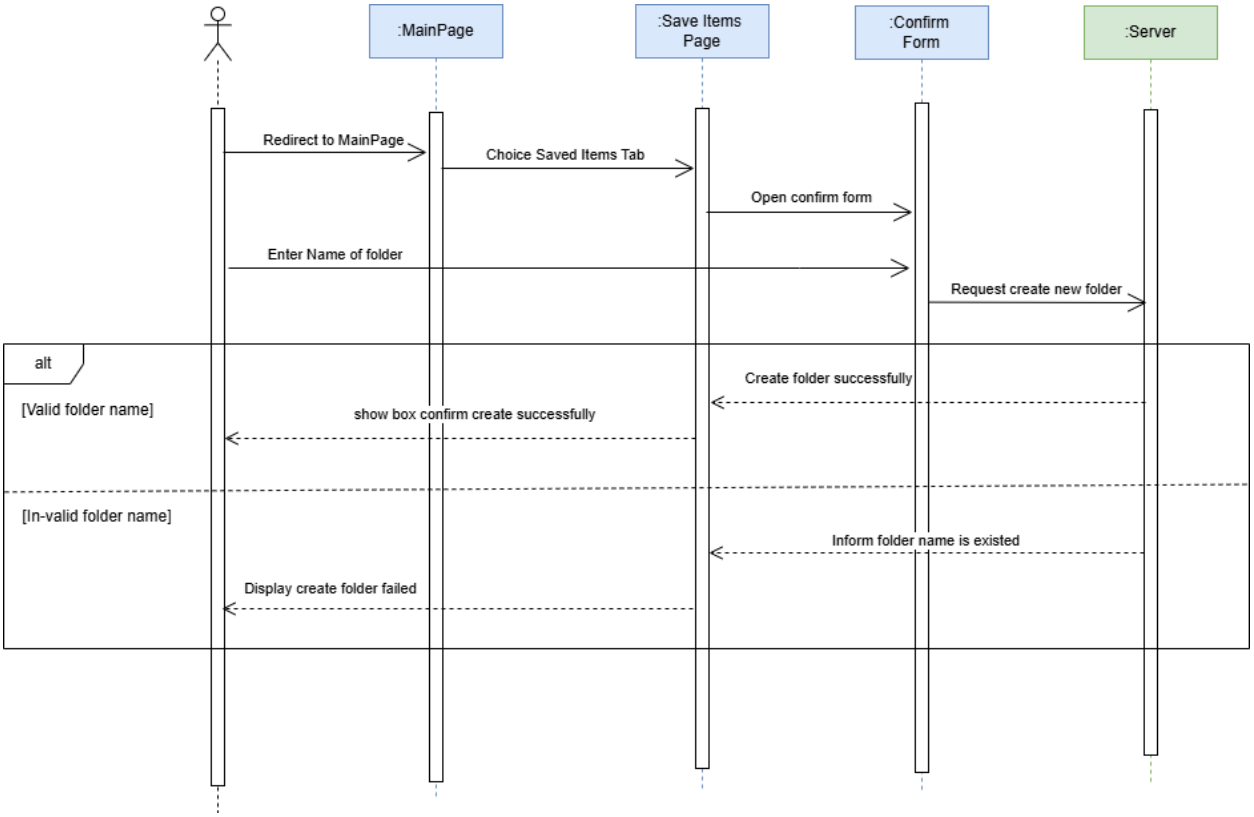


Figure 4-13. Create Folder Sequence Diagram

4.1.14. In-active Account

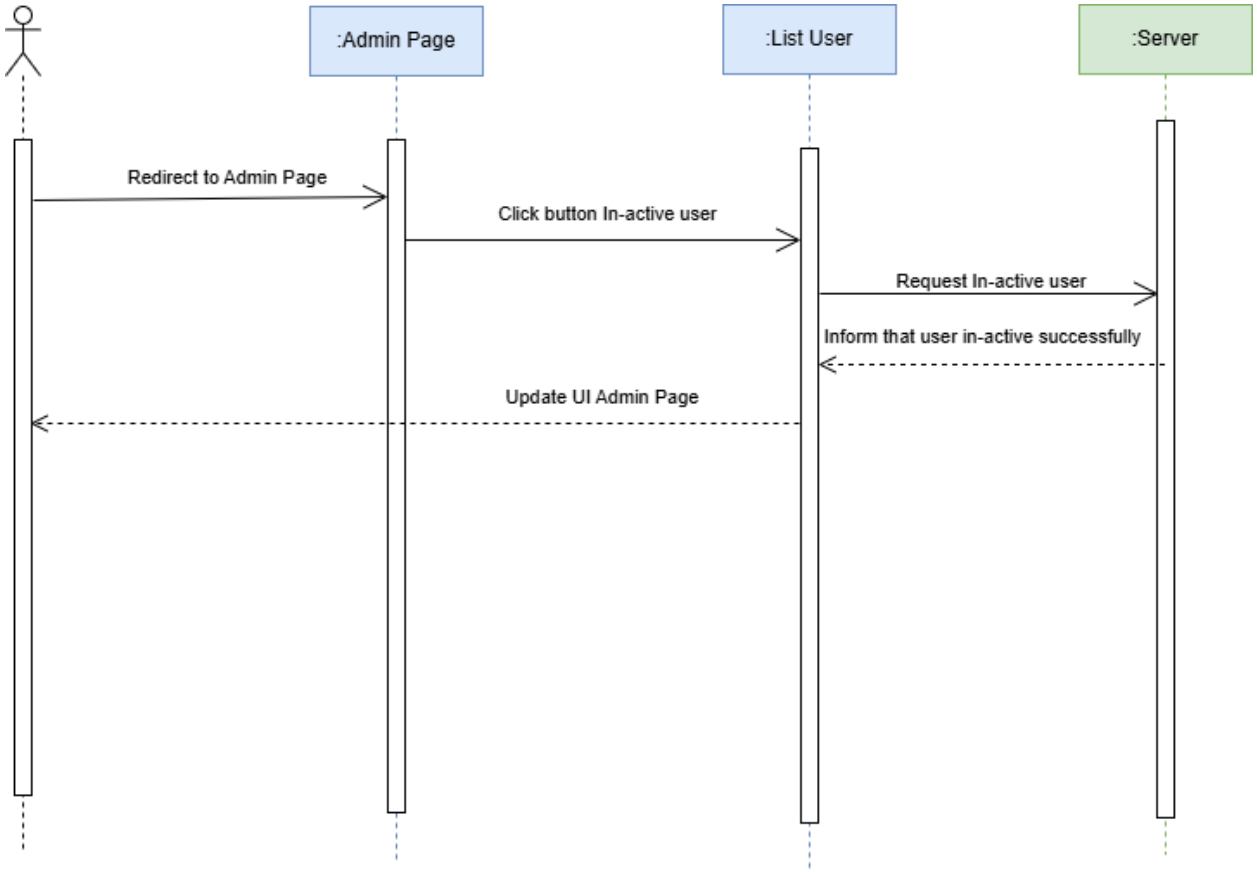


Figure 4-14. In-Active Account Sequence Diagram

4.1.15. Delete Account

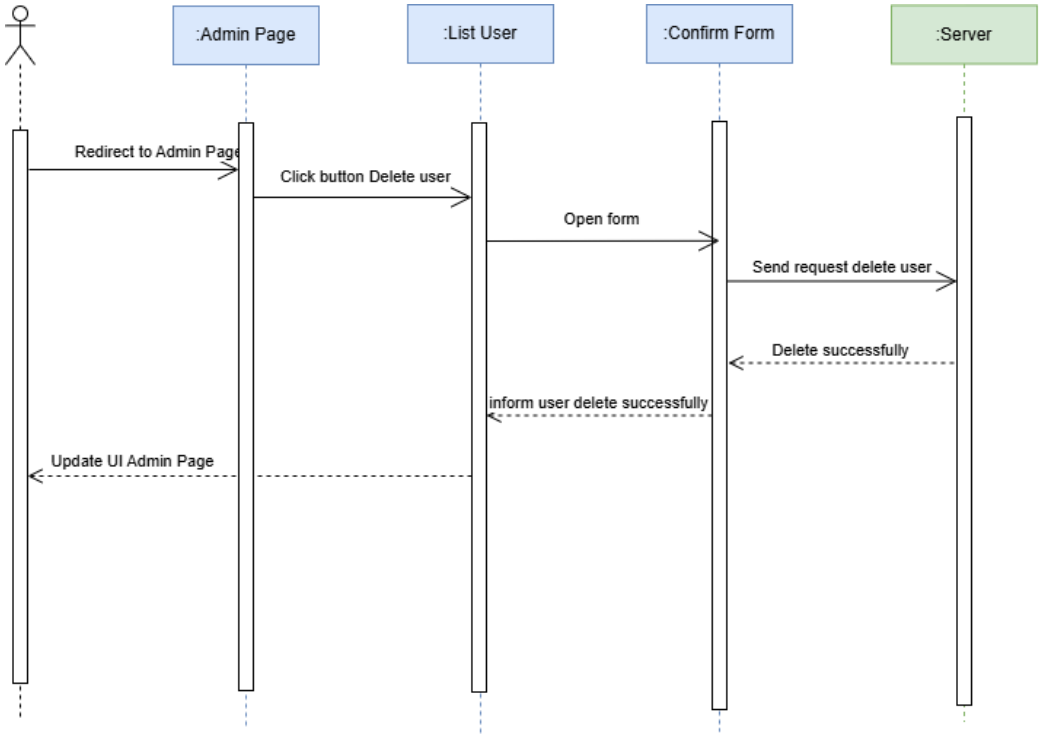


Figure 4-15. Delete Account Sequence Diagram

4.1.16. Chatting

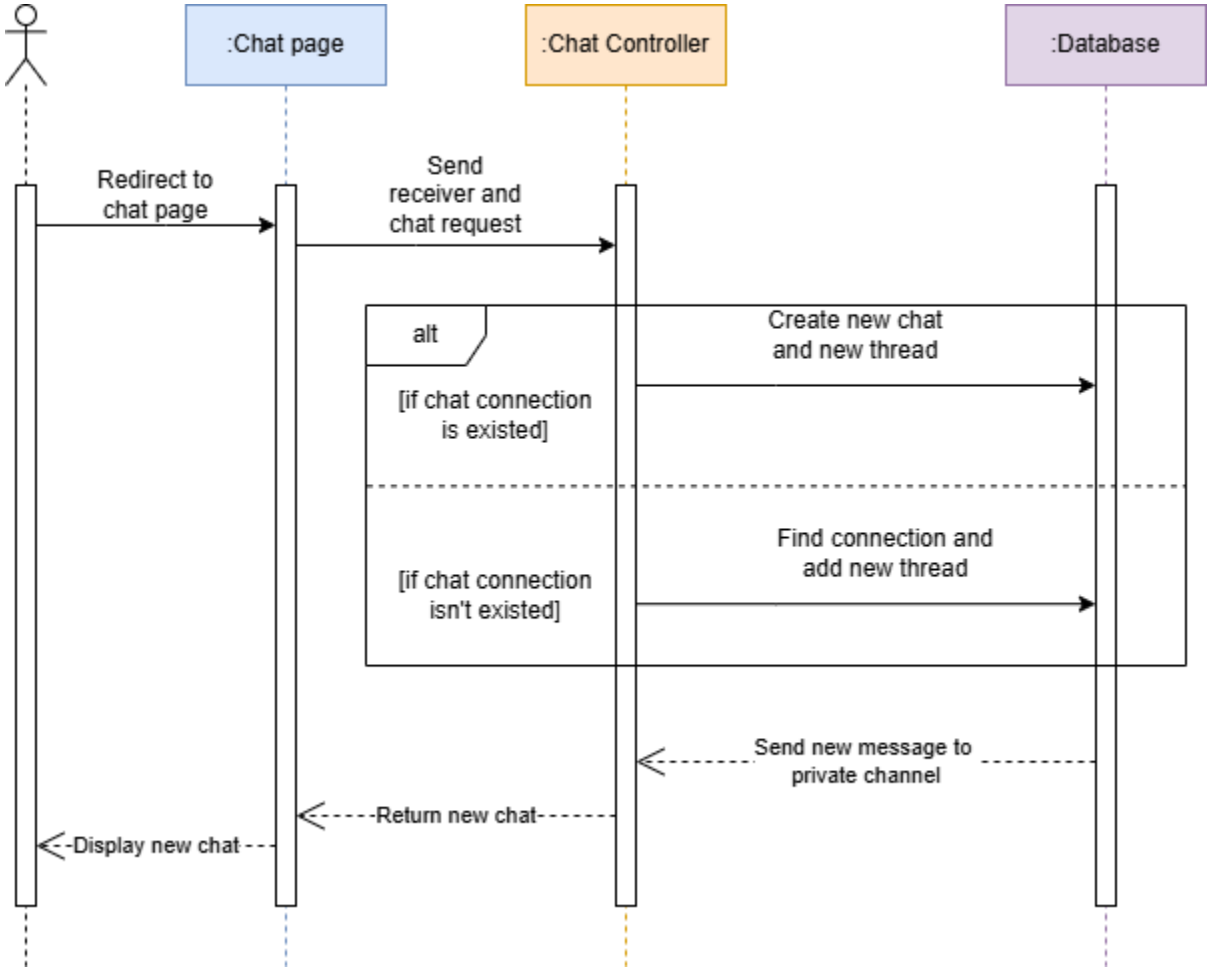


Figure 4-16. Chatting Sequence Diagram

4.2. Database design

4.2.1. Class Diagram

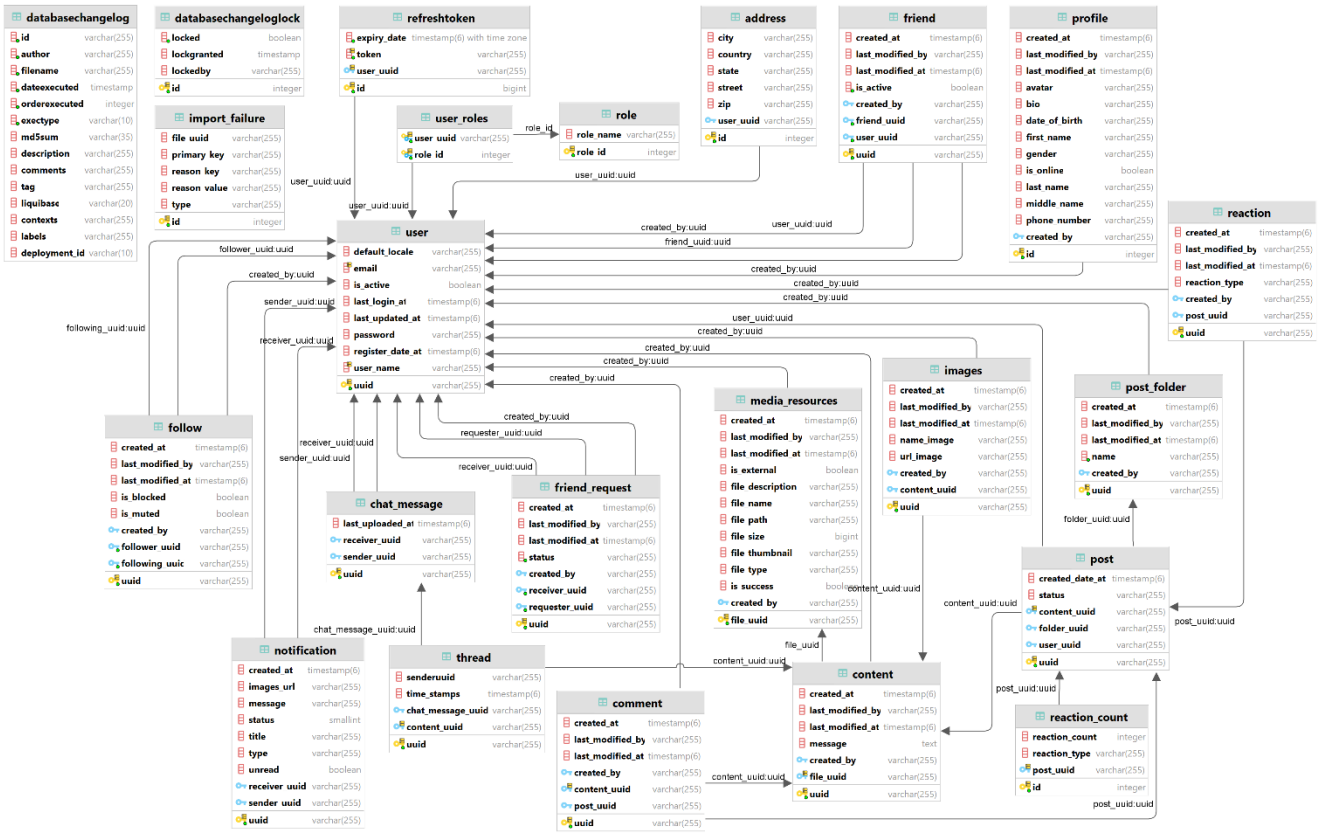


Figure 4-17. Class Diagram of Social Media System

4.2.2. Entity-Relational Database

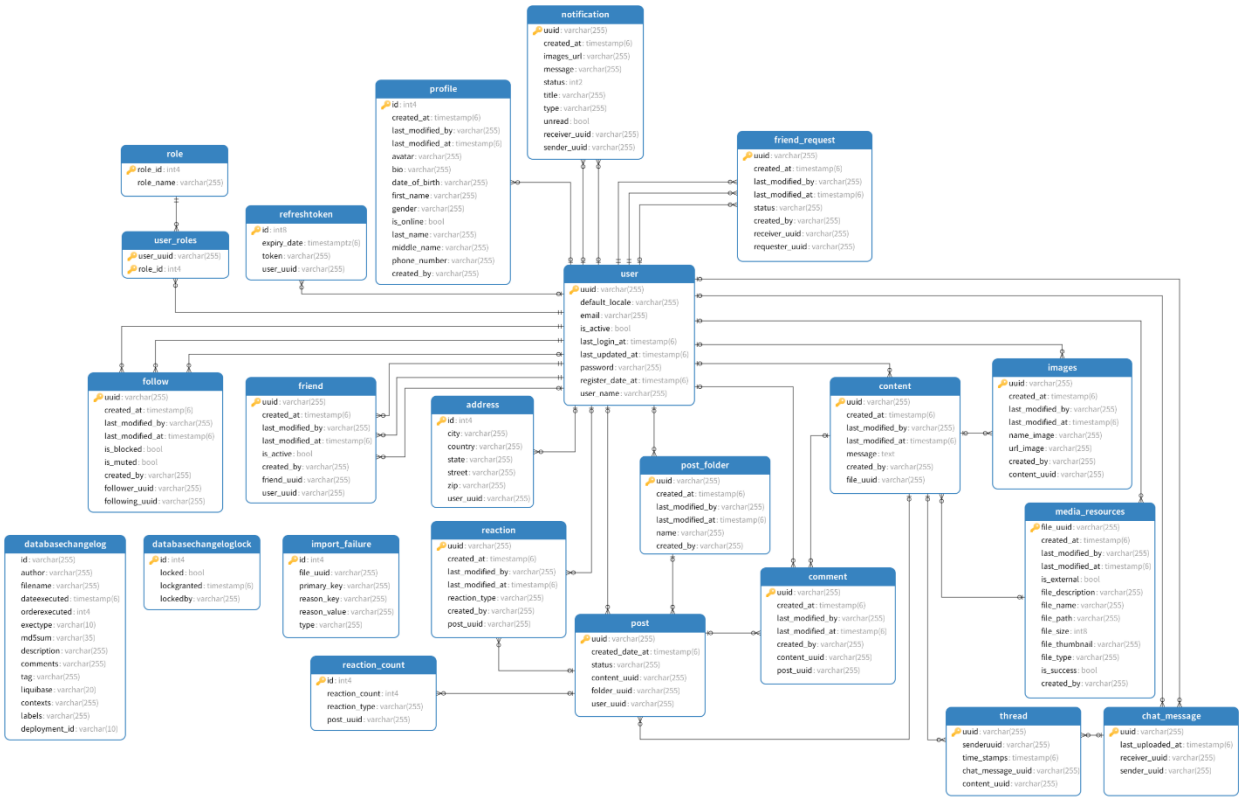


Figure 4-18. Entity-Relational Database

4.2.3. Database Description

4.2.3.1. User Entity

No	Attribute	Type	Range/ Contraints	Meaning	Notes
1	uuid	VARCHAR(36)	UUID format	Unique identifier for the user.	
2	user_name	VARCHAR	Minimum 2 characters	The username chosen by the user.	
3	password	VARCHAR	Minimum 6 characters	The user's account password.	
4	email	VARCHAR	Valid email format	Email address of the user.	

5	default_locale	VARCHAR	Language/region code (e.g., en)	Default locale for the user, used for localization.	
6	is_active	BOOLEAN	True or False	Indicates if the user account is active.	
7	register_date_at	TIMESTAMP	Timestamp	Date and time when the user registered.	
8	last_login_at	TIMESTAMP	Timestamp	Date and time of the user's last login.	
9	last_updated_at	TIMESTAMP	Timestamp	Date and time when the user's details were last updated.	

Table 4-1. User Entity

4.2.3.2. Profile Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	id	INTEGER	Auto-incremented positive value	Unique identifier for the profile	
2	first_name	VARCHAR		User's first name	
3	middle_name	VARCHAR		User's middle name	
4	last_name	VARCHAR		User's last name	
5	gender	VARCHAR	Enum values: *MALE, FEMALE, OTHER*	Gender of the user.	
6	date_of_birth	VARCHAR		User's date of birth.	
7	bio	VARCHAR		User's biography or description.	

8	phone_number	VARCHAR		User's phone number.	
9	avatar	VARCHAR		URL or path to the user's avatar.	
10	is_online	BOOLEAN	true or false	Indicates if the user is online.	

Table 4-2. Profile Entity

4.2.3.3. Address Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	id	INTEGER	Auto-incremented positive value	Unique identifier for the address.	
2	street	VARCHAR		Street address of the user.	
3	city	VARCHAR		City of the user.	
4	state	VARCHAR		State or region of the user.	
5	zip	VARCHAR		ZIP or postal code of the user.	
6	country	VARCHAR		Country of the user.	

Table 4-3. Address Entity

4.2.3.4. Role Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	role_id	INTEGER		Unique identifier for the role.	
2	role_name	VARCHAR	ERole values: *ROLE_USER*, *ROLE_ADMIN*	Name of role	

Table 4-4. Role Entity

4.2.3.5. Post Entity

No	Attribute	Type	Range/	Meaning	Notes

			Contrains		
1	uuid	VARCHAR(36)		Unique identifier for the post.	
2	status		EstatusPost: *PUBLIC*, *PRIVATE*, *FRIENDS*	Status of the post.	
3	created_date_at	TIMESTAMP	Timestamp	Timestamp when the post was created.	
4	content_uuid	VARCHAR(36)	Matches UUID in Content table	References the associated content.	
5	folder_uuid	VARCHAR(36)	Matches UUID in PostFolder table		
6	user_uuid	VARCHAR(36)	Matches UUID in User table	References the author of the post.	

Table 4-5. Post Entity

4.2.3.5. Content Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the content.	
2	file_uuid	VARCHAR(36)	Matches file_UUID in MediaResources table	References the associated media resource.	
3	message	TEXT		Text message or description of the content.	

Table 4-6. Content Entity

4.2.3.6. Reaction Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes

1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the reaction.	
2	post_UUID	VARCHAR(36)	Matches UUID in Post table	References the post being reacted to.	
3	reaction_type	VARCHAR	Ereaction: *LIKE*, *HAHA*, *WOW*, *SAD*, *ANGRY*, *LOVE*	Type of reaction (e.g., like, love).	

Table 4-7. Reaction Entity

4.2.3.7. Comment Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the comment.	
2	post_UUID	VARCHAR(36)	Matches UUID in Post table	References the post the comment belongs to.	
3	content_uuid	VARCHAR(36)	Matches UUID in Content table	References the content of the comment.	

Table 4-8. Comment Entity

4.2.3.8. Image Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the Image.	
2	name_image	VARCHAR		The name of the image.	
3	url_image	VARCHAR		URL or path to the image.	

4	content_uuid	VARCHAR		References the content this image belongs to.	

Table 4-9. Image Entity

4.2.3.9. Follow Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the follow record.	
2	is_blocked	BOOLEAN	true or false	Indicates if the follower has blocked the following user.	
3	is_muted	BOOLEAN	true or false	Indicates if the follower has muted the following user.	
4	follower_uuid	VARCHAR(36)	Matches UUID in User table	References the user who is following another user.	
5	following_uuid	VARCHAR(36)	Matches UUID in User table	References the user being followed.	

Table 4-10. Follow Entity

4.2.3.10. Friend Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the friendship record.	
2	user_uuid	VARCHAR(36)	Matches UUID in User table	References the current user in the friendship.	

3	friend_uuid	VARCHAR(36)	Matches UUID in User table	References the friend of the current user.	
4	is_active	BOOLEAN	true or false	Indicates if the friendship is active.	

Table 4-11. Friend Entity

4.2.3.11. FriendRequest Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the friend request.	
2	requester_uuid	VARCHAR(36)	Matches UUID in User table	References the user who sent the friend request.	
3	receiver_uuid	VARCHAR(36)	Matches UUID in User table	References the user who received the friend request.	
4	status	VARCHAR(36)	RequestStatus: *PENDING, ACCEPTED, REJECTED*	Represents the current status of the friend request.	

Table 4-12. FriendRequest Entity

4.2.3.12. MediaResources Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	file_UUID	VARCHAR(36)	Auto-generated UUID	Unique identifier for the media resource.	
2	file_name	VARCHAR(255)		The name of the media file.	
3	file_type	VARCHAR(255)	EfileType values	The type of the file (e.g., image, video).	

4	is_external	BOOLEAN	True/False	Indicates whether the media is external (e.g., hosted outside the application).	
5	file_path	VARCHAR(255)		The path to the file within the application or storage.	
6	file_size	BIGINT	File size in bytes	Size of the file in bytes.	
7	file_description	TEXT		A brief description of the file content.	
8	file_thumbnail	VARCHAR(255)		Path or URL to the file's thumbnail image.	
9	is_success	BOOLEAN	True/False	Indicates whether the file was uploaded successfully.	

Table 4-13. MediaResources Entity

4.2.3.13. Notification Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)		Unique identifier for notification.	
2	status	VARCHAR		The current status of the notification.	
3	title	VARCHAR(255)		The title of the notification.	
4	message	TEXT		The message of the notification.	

5	imagesUrl	VARCHAR(255)		URL of any image(s) with the notification.	
6	type	VARCHAR(255)		The type of notification.	
7	createdAt	TIMESTAMP	Not null	The timestamp when it was created.	
8	unread	BOOLEAN		Flag indicating whether the notification is unread.	
9	sender	VARCHAR(36)	Foreign key referencing User(UUID)	The user who sent the notification.	
10	receiver	VARCHAR(36)	Foreign key referencing User(UUID)	The user who receives the notification.	

Table 4-14. Notification Entity

4.2.3.14. PostFolder Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the post folder.	
2	name	VARCHAR(255)	Not null	The name of the post folder.	

Table 4-15. PostFolder Entity

4.2.3.15. ChatMessage Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the chat message.	

2	sender_uuid	VARCHAR(36)	Foreign key referencing User.UUID	The sender of the chat message	
3	receiver_uuid	VARCHAR(36)	Foreign key referencing User.UUID	The receiver of the chat message	
4	last_uploaded_at	TIMESTAMP	Timestamp	The timestamp when the chat message was last uploaded.	

Table 4-16. ChatMessage Entity

4.2.3.16. Thread Entity

No	Attribute	Type	Range/ Contrains	Meaning	Notes
1	uuid	VARCHAR(36)	Auto-generated UUID	Unique identifier for the thread.	
2	chat_message_uuid	VARCHAR(36)	Foreign key referencing ChatMessage.uuid	The associated chat message for this thread.	
3	Sender_uuid	VARCHAR(36)		The sender of the thread message.	
4	content_uuid	VARCHAR(36)	Foreign key referencing Content.uuid	The content associated with the thread message.	
5	timeStamps	TIMESTAMP		The timestamp when the thread message was created or modified.	

Table 4-17. Thread Entity

4.3. User Interface Design

No.	Screen Name	Description

SCU01	Register screen	User will be forced to fill all of the fields for registering an account
SCU02	Login Screen	Login page for registered users
SCU03	Profile Screen	New user must be filled all of the fields for make user profile
SCU04	Main screen	Show social information, including new posts, user information, notification, follow request recommendations, change languages
SCU05	Following Screen	User can view all users and each user profile and request the following.
SCU06	Feed Screen	User can view all post
SCU07	SaveItem Screen	User can view all save items, create or delete collections
SCU08	Resource Screen	User can view all media files, delete, view detailing and download them
SCU09	Chat Screen	User can view all users, including online and offline users, select and send messages realtime for them.
SCU10	Main friend Management Screen	User can view all users that are not friends and can view the request to add friends, and can send request add friends for others.
SCU11	All Friend Screen	User will view all friends and their profiles.
SCU12	Recommend Friend screen	User will view users that are not friends and their detail profiles

SCU13	Follower Screen	User will view their followers and the detailed profile of each follower.
SCU14	Main Admin Screen	Admin will manager users, posts, edit user profile and delete use

Table 4-18. List of User Interface Design

4.3.1. Admin Interface

4.3.1.1. Main Admin Screen (SCU14)

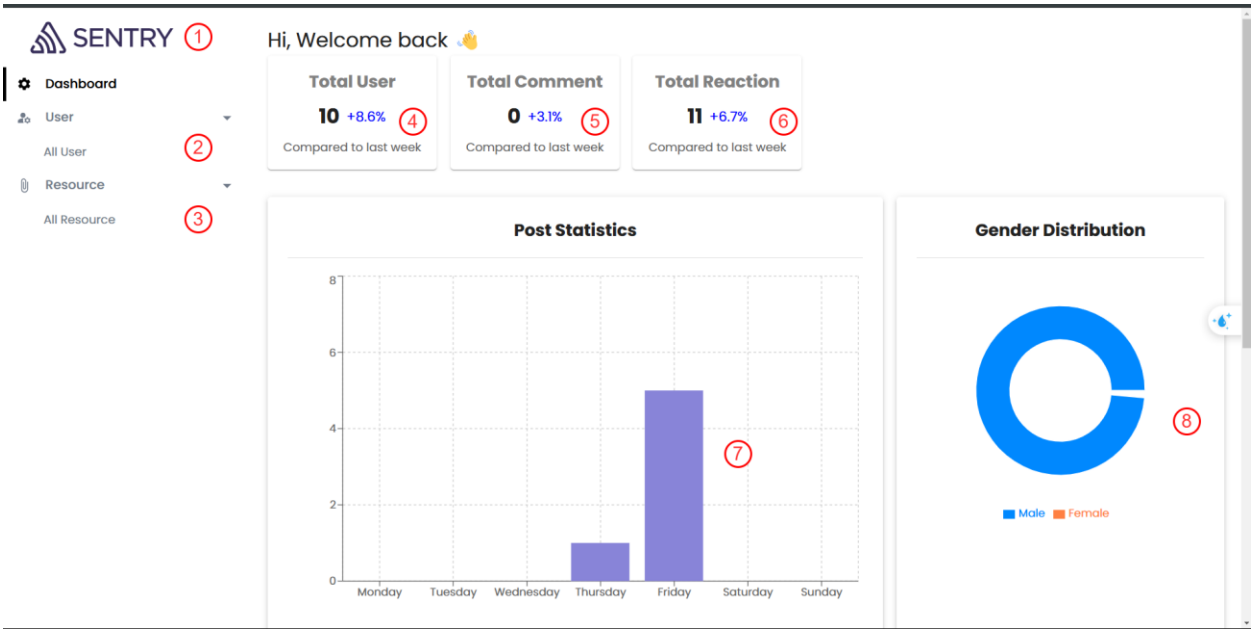


Figure 4-19. Dashboard Admin Screen

No.	Name	Type	Event
1	Logo	Button	Back to main screen
2	AllUser	Button	After clicking, the page goes to user management screen
3	All Resource	Button	After clicking, the page goes to resource management screen
4	Total user	TextView	Show total user for admin
5	Total Comment	TextView	Show total comment
6	Total reaction	TextView	Show total reaction

7	Post statistics	Chart	Show statistics all the posts in system
8	Gender Distribution	Chart	Show distribution of gender of user account

Table 4-19. Dashboard Admin Screen

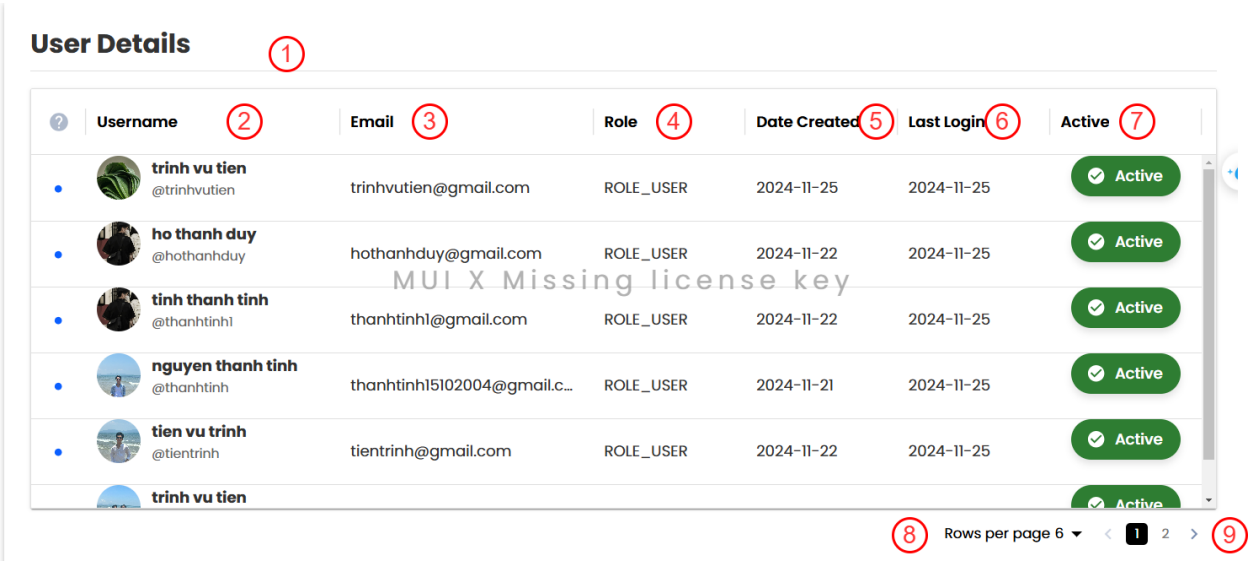


Figure 4-20. Table of List Active User

No.	Name	Type	Event
1	User detail	Table	Show all user in system
2	UserName	Button	Sort by user name
3	Email	Button	Sort by email
4	Role	Button	Sort by role
5	Date Created	Button	Sort by date
6	Last login	Button	Sort by date
7	Active	Button	Sort by status
8	Rows per page	Button	For use change number of users can display in table
9	narrow	Button	For user can next and back the table user information

Table 4-20. Table of List Active User

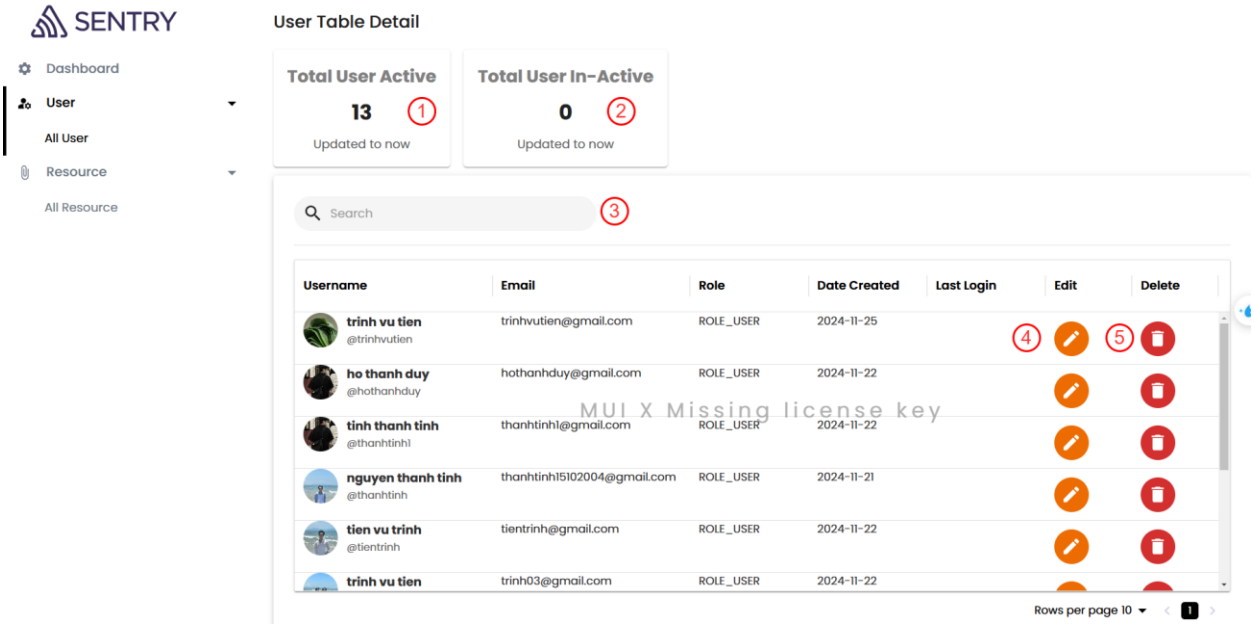


Figure 4-21. User Management Page

No.	Name	Type	Event
1	User detail	Table	Show all user in system
2	UserName	Button	Sort by user name
3	Email	Button	Sort by email
4	Role	Button	Sort by role
5	Date Created	Button	Sort by date
6	Last login	Button	Sort by date
7	Active	Button	Sort by status
8	Rows per page	Button	For use change number of users can display in table
9	Narrow	Button	For user can next and back the table user information

Table 4-21. User Management Page

4.3.2. User Interface

4.3.2.1. Register screen (SCU01)

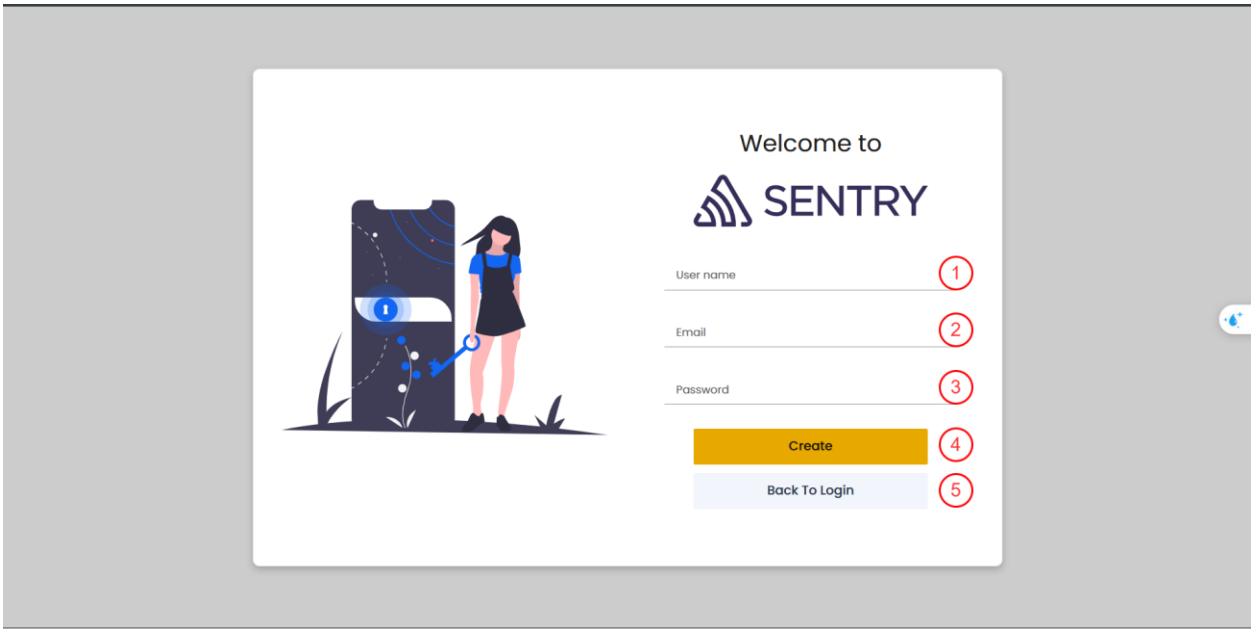


Figure 4-22. Register screen

No.	Name	Type	Event
1	UserName	TextBox	For user input username
2	Email	TextBox	For user input email
3	Password	TextBox	For user input password
4	Create	Button	After clicking, new user account will be created and the page goes to profile screen

Table 4-22. Register screen

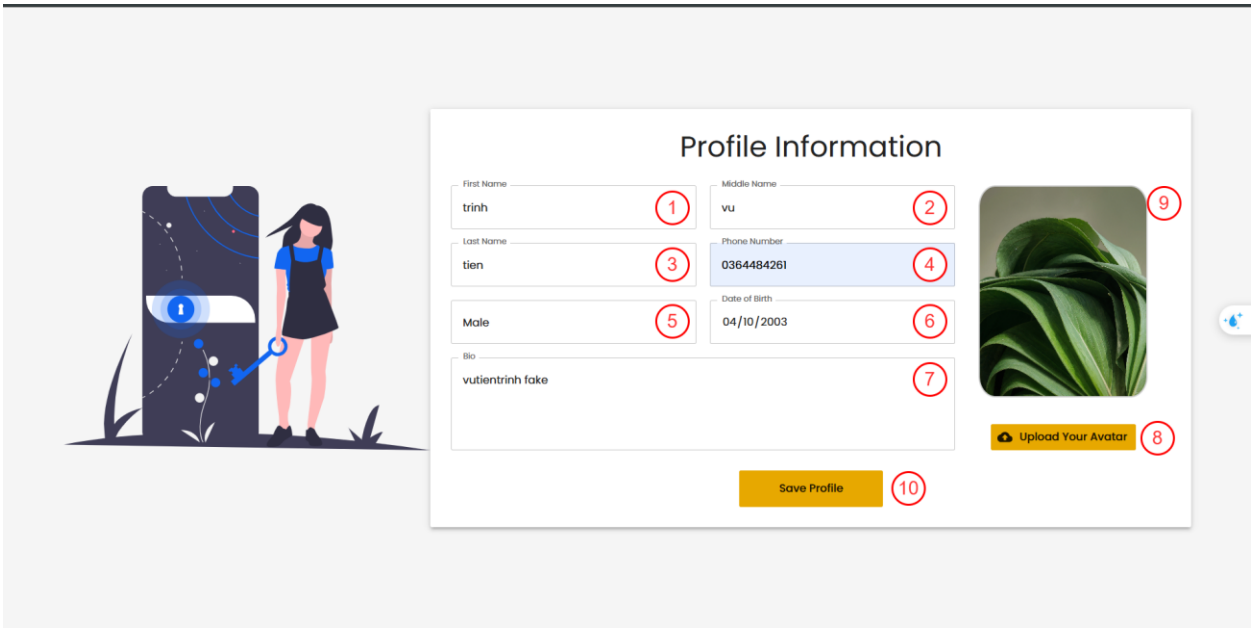


Figure 4-23. Register Form - Profile Information Page

No.	Name	Type	Event
1	First name	TexBox	For user input first name
2	Middle name	TextBox	For user input middle name
3	Last Name	TextBox	For user input lastname
4	Phone Number	TextBox	For user input phone number
5	Male	Selected Box	For user select gender
6	Date of Birth	Date	For user input date of birth
7	Bio	TextBox	For user input bio
8	Upload your Avatar	Button	After clicking, file explorer will be open and user choose your image
9	Image	Image	Avatar will be display here
10	Save profile	Button	After clicking, new profile will be saved and the page goes to main screen

Table 4-23. Register Form - Profile Information Page

4.3.2.2. Login Screen (SCU02)

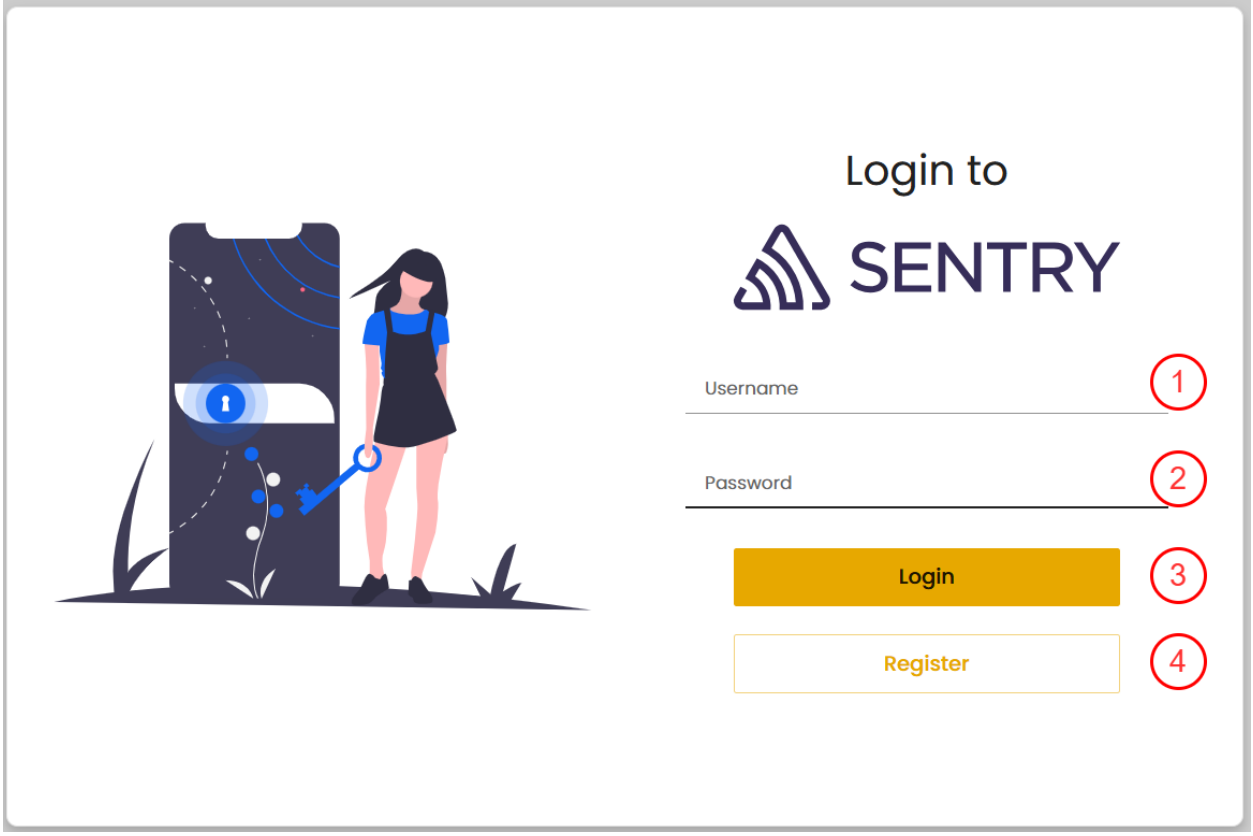


Figure 4-24. Login Screen

No.	Name	Type	Event
1	User Name	TextBox	For user to input username
2	Password	TextBox	For user to input password
3	Login	Button	After clicking, the page goes to main screen
4	Register	Button	After clicking, the page goes to Register Screen

Table 4-24. Login Screen

4.3.2.4. Main screen (SCU04)

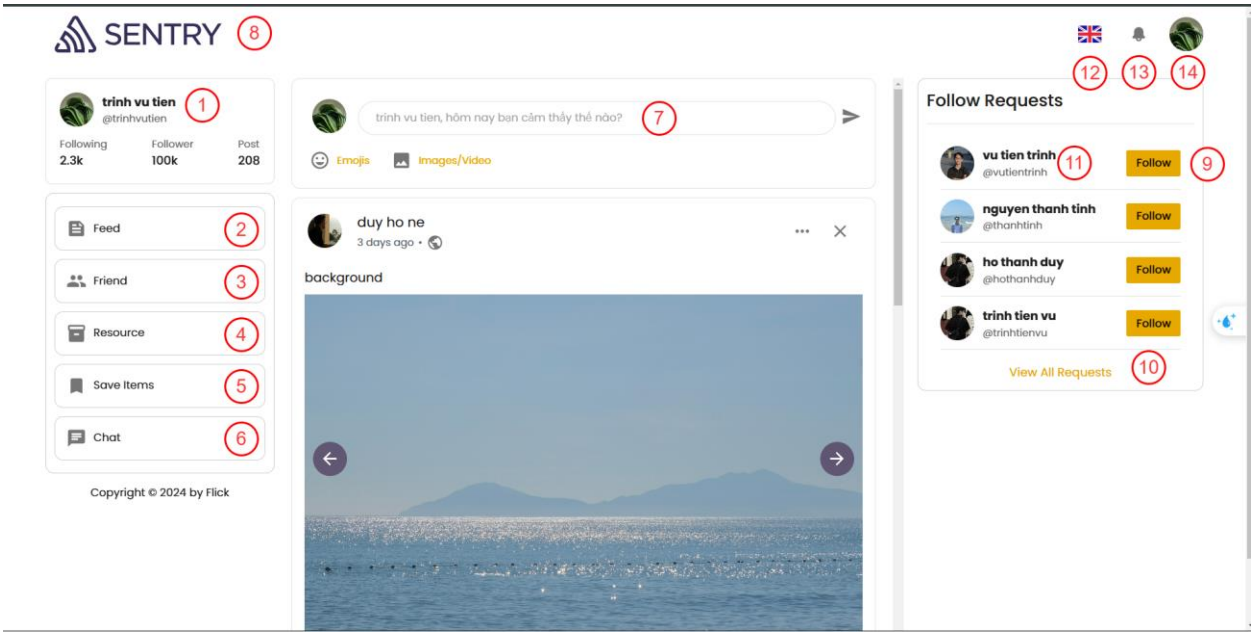


Figure 5-25. Main Screen

No.	Name	Type	Event
1	User information	Text	Show name and email of user
2	Feed	Button	After clicking, the page goes to feed screen
3	Friend	Button	After clicking, the page goes to friend screen
4	Resource	Button	After clicking, the page goes to resource screen
5	Save Items	Button	After clicking, the page goes to Save Items screen
6	Chat	Button	After clicking, the page goes to chat screen
7	New post	Box	After clicking, the new post model will be popped up
8	Logo	Link	After clicking, the page will goes to main screen

9	Follow	Button	After clicking, user send follow request to another user
10	View all requests	Link	After clicking, the page goes to following screen
11	Another user information	Link	after clicking, the page goes to detail profile user screen

Table 4-25. Main Screen

4.3.2.6. Feed screen (SCU06)

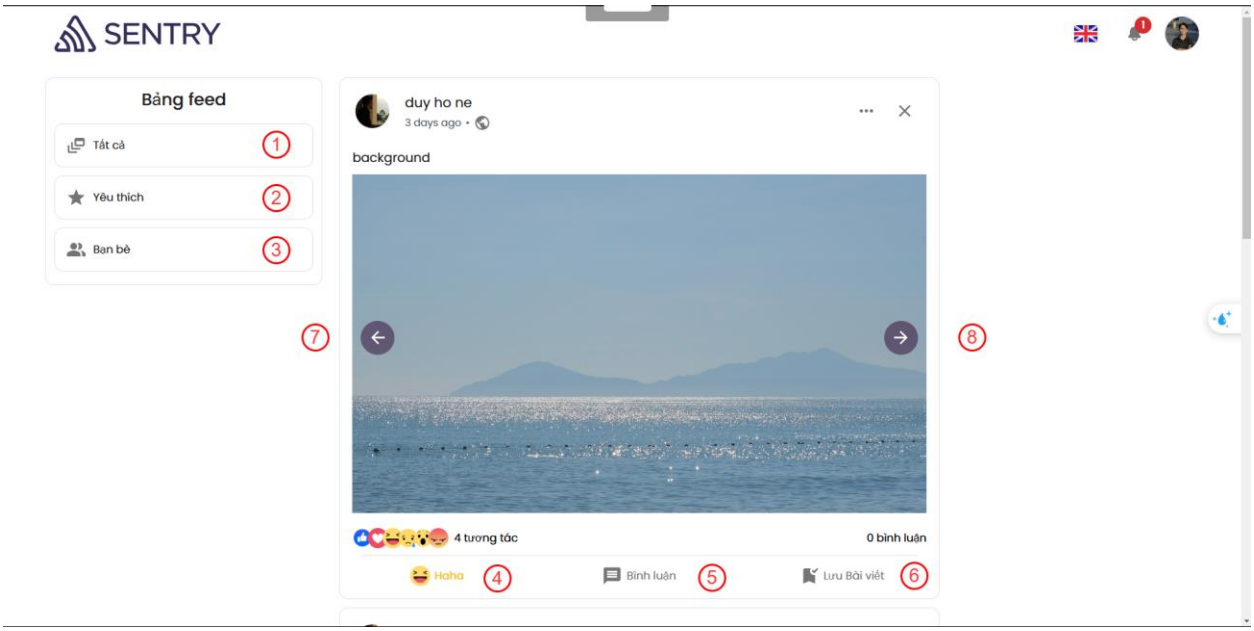


Figure 4-26. Feed Screen

No.	Name	Type	Event
1	Tatca	Button	The page display all of posts
2	Yeu thich	Button	The page display posts that in favorites of user
3	Ban be	Button	The page displays the posts of the user's friends.
4	Tuong tac	Selected Box	User can react the post
5	Binh luan	Button	User can comment to post

6	Luu bai viet	Button	User can save the post to Save Items

Table 4-26. Feed Screen

4.3.2.7. Saved Items screen (SCU07)

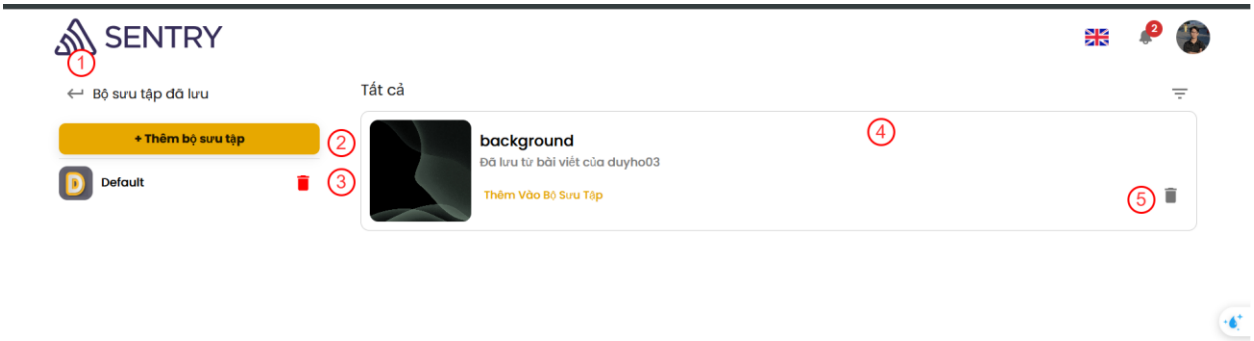


Figure 4-27. Saved Items screen

No.	Name	Type	Event
1	Back	Button	Back to main page
2	Them bo suu tap	Button	For user create new Collection
3	Delete	Button	Delete collection
4	Collection's Item	TextView	Display collection's items
5	Delete item	Button	For user can be deleted items in collection

Table 4-27. Saved Items screen

4.3.2.8. Resource screen (SCU08)

Figure 4-28. Resource screen

No.	Name	Type	Event
1	Edit	Button	Open edit mode.
2	View	Button	User can view image larger.
3	Download	Button	User can download image to their devices.
4	Delete	Button	User can delete media item.

Table 4-28. Resource screen

4.3.2.9. Chat screen (SCU09)

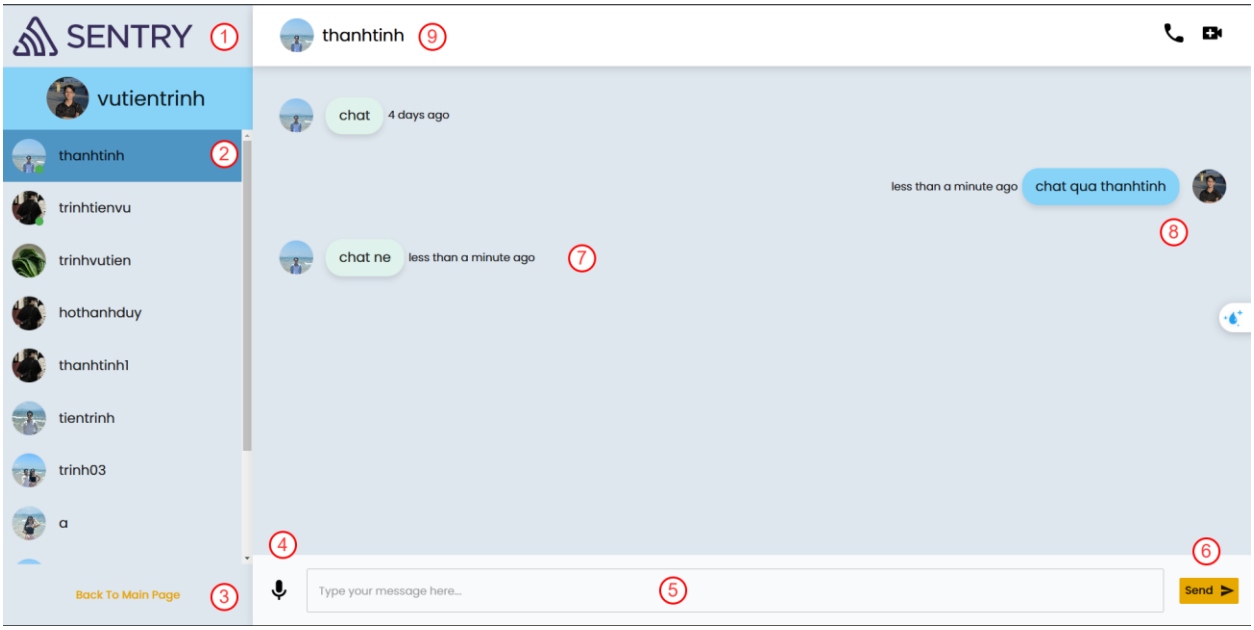


Figure 4-29. Chatting screen

No.	Name	Type	Event
1	Logo	Link	Back to main page
2	Other user	Button	For user can click to choose receiver message
3	Back to main page	Link	Back to main page
4	Voice recorder	Button	For user to record their voice and then it changes to text.
5	Type your message here	TextBox	For user input messages
6	Send	Button	After clicking, new message will send to receiver
7	Receiver's message	TextView	Display Receiver' messages
8	Sender's messages	TextView	Display Sender's messages
9	Receiver Information	TextView	Display name and avatar of receiver.

Table 4-29. Chatting screen

4.3.2.10. Main friend Management Screen (SCU10)

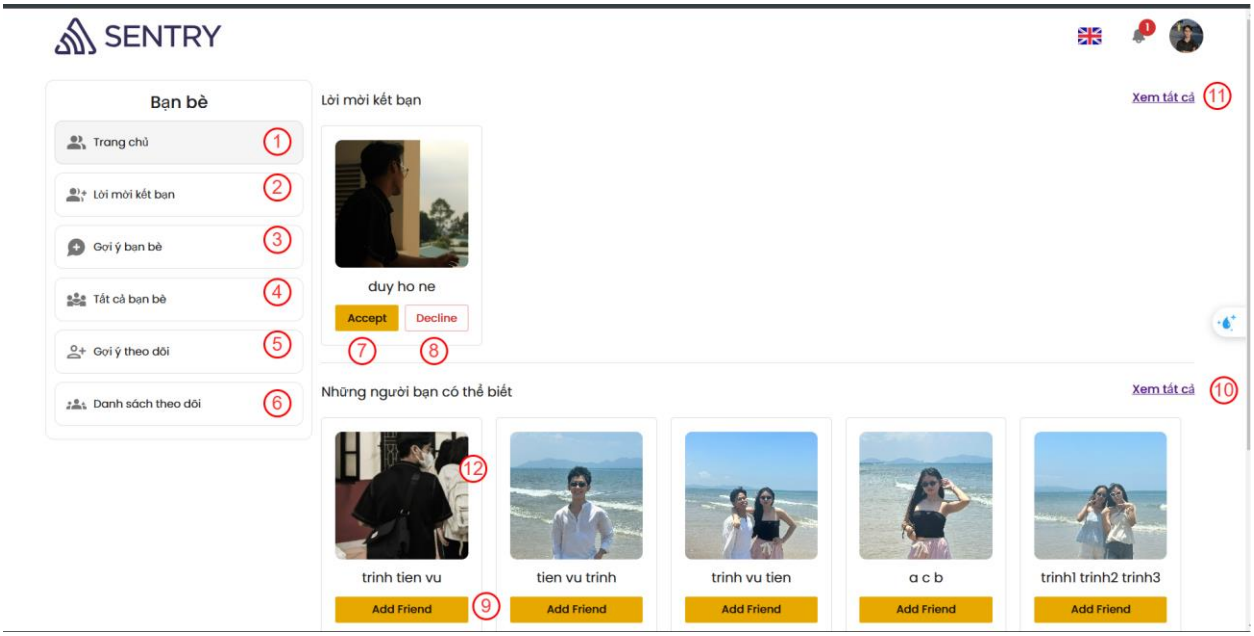


Figure 4-30. Main friend Management Screen

No.	Name	Type	Event
1	Trang chu	Button	After clicking, the page goes to main friend page
2	Loi moi ket ban	Button	After clicking, the page goes to new friend request screen
3	Goi y ban be	Button	After clicking, the page goes to “goi y ket ban “ screen
4	Tat ca ban be	Button	After clicking, the page goes to all friend screen
5	Goi y theo doi	Button	After clicking, the page goes to ‘goi y theo doi “ screen
6	Danh sach theo doi	Button	After clicking, the page goes to all follow screen
7	Accept	Button	For user can accept the request add new friend of other user

8	Decline	Button	For user can reject the request add new friend of other user
9	Add Friend	Button	For user send requests add new friends.
10	Xem tat ca	Link	The page goes to “goi y ban be” screen
11	Xem tat ca	Link	The page goes to new friend request screen
12	Avatar	Image	Display User avatar

Table 4-30. Main friend Management Screen

4.3.2.11. All Friend Screen (SCU11)

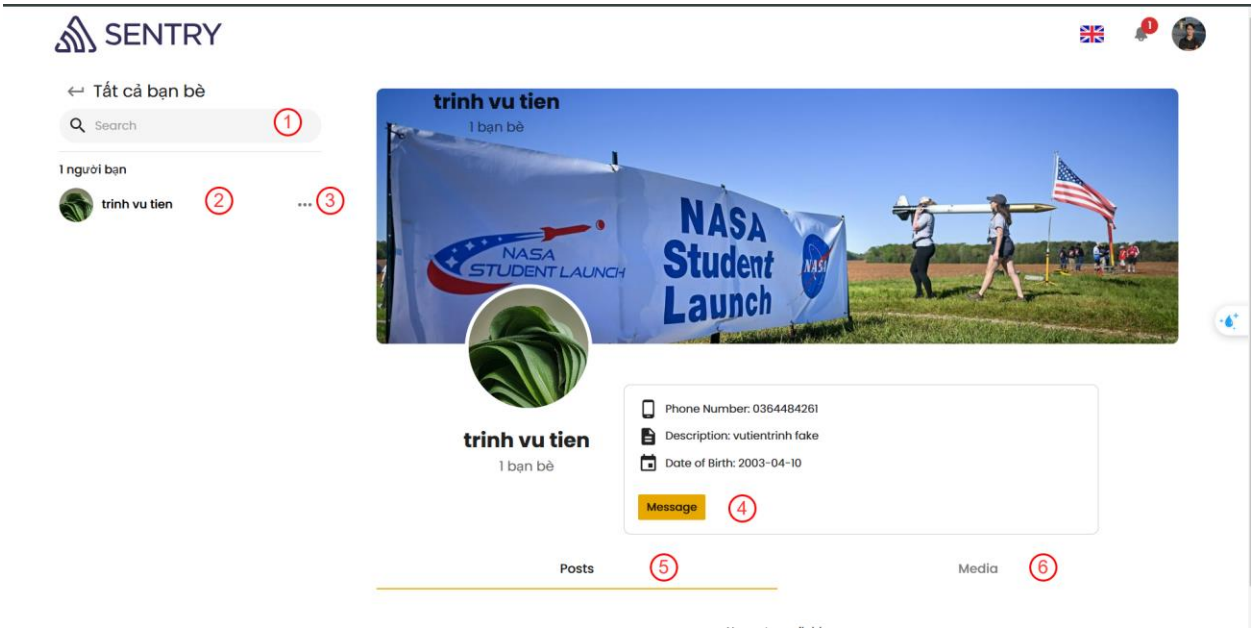


Figure 4-31. Show All Friend Screen

No.	Name	Type	Event
1	Search	TextBox	For user can search friend
2	Use name and avatar	TextView + Image	Display use name and avatar
3	More Option	Button	Display more option component for user can delete friend
4	Message	Button	For user can chat with friend

5	Post	Tab	Display all post of friend
6	Media	Tab	Display all media of friend

Table 4-31. All Friend Screen

4.3.2.12. Recommend Friend screen (SCU12)

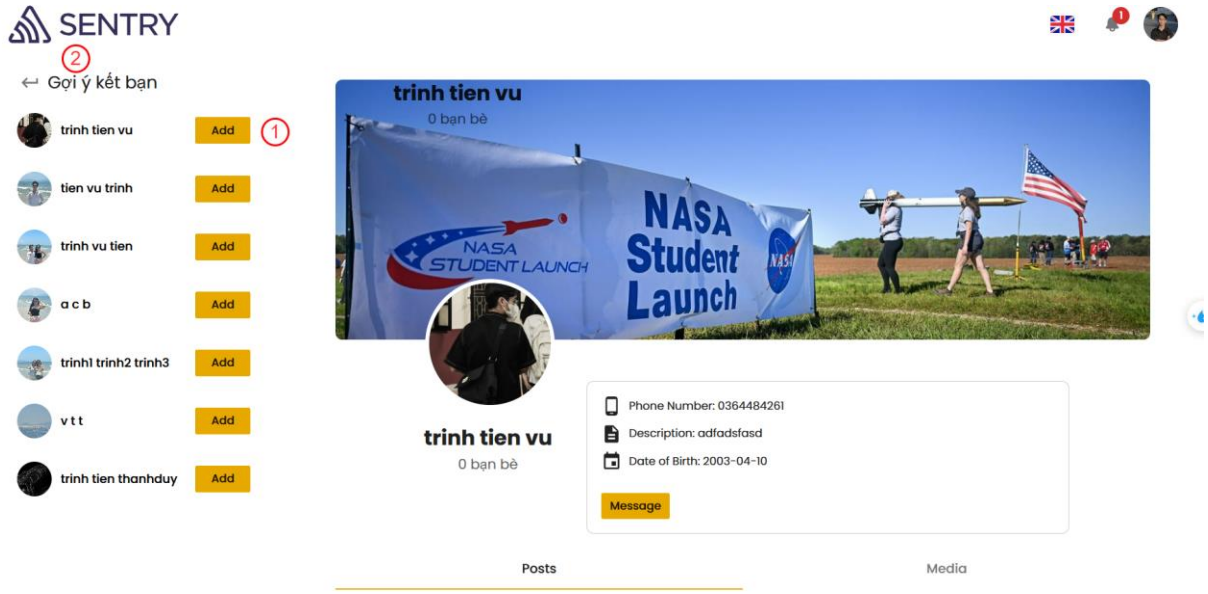


Figure 4-32. Recommend Friend screen

No.	Name	Type	Event
1	Add	Button	For user add new friend
2	Back	Button	Back to friend management screen

Table 4-32. Recommend Friend screen

4.3.2.13. Follower Screen (SCU13)

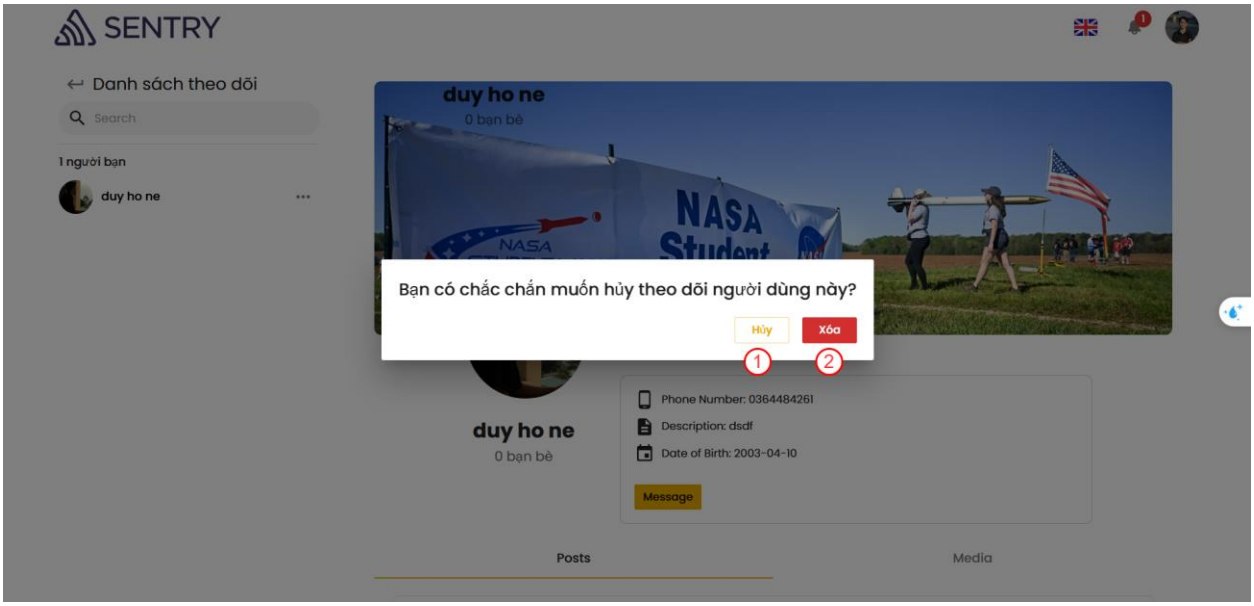


Figure 4-33. Follow Screen

No.	Name	Type	Event
1	Huy	Button	After clicking, user can cancel
2	Xoa	Button	After clicking, the follower is removed

Table 4-33. Follow Screen

CHAPTER 5. IMPLEMENTATION AND TESTING

5.1. Implementation

5.1.1. Application Implementation

No.	Application	Description
1	NodeJS	- Version: v20.7.0 - Link Download:https://nodejs.org/en/blog/release/v20.7.0
2	Git	-Version: v2.42.0 - Link Download: https://git-scm.com/download

Table 5-1. List of application need for Social Application

5.1.1.1. Back-End Implement

Installation Instruction

Step 1. Clone the Repository (SSH - Setup your github account)

```
git clone git@github.com:SOCIAL-NETWORK-2425/social-network-backend.git
```

Step 2. Set up your database

- Download PostgreSQL version 16 from the official website
- Create the ***SocialMedia*** Database by psql shell:

```
CREATE DATABASE SocialMedia;
```

Step 3. Configure the *.env* file

Set up your .env file according to the env-sample provided in the repository.

Step 4. Configure Folders

Ensure you have the necessary directories (uploadFiles, uploadImages) inside the SocialNetwork folder. If these folders are missing, navigate to the scripts directory and run the following script:

- For ubuntu run: ./init_folder.sh
- For windows run: ./init_folder.bat

Step 5. Run the Application: java -jar target/ socialnetwork-0.0.1-SNAPSHOT.war

Variables	Description
DATABASE_URL	The JDBC connection string for connecting to the PostgreSQL database. It includes the host, port, and database name.
DATABASE_USERNAME	The username used to authenticate with the PostgreSQL database.
DATABASE_PASSWORD	The password used to authenticate with the PostgreSQL database.
PORT	The port on which the application server is running.
PUBLIC_FOLDER	The directory path for storing public assets like uploaded files or shared resources.
IMPORTED_FILES	The directory path for storing files that have been imported into the application.
BASE_FOLDER-UPLOAD	The base directory path for uploaded files within the application.
BASE_FOLDER	The directory path for storing locale-specific data files, such as translations or region-specific content.

BASE_FOLDER_LOCALE	The directory path for storing locale-specific data files, such as translations or region-specific content.
BASE_IMPORT_FILE	The base directory path for importing files; currently left undefined.
IMAGE_FOLDER	The directory path for storing uploaded images.
CLOUDINARY_CLOUD_NAME	The Cloudinary account identifier used to organize and manage assets.
CLOUDINARY_API_KEY	The API key used to authenticate with Cloudinary for uploading and managing assets.
CLOUDINARY_API_SECRET	The secret key used for securely authenticating requests to Cloudinary.
CLOUDINARY_URL	The URL string for connecting to the Cloudinary API using the provided credentials.
JWT_SECRET	The secret key used to sign and verify JSON Web Tokens (JWT) for secure authentication and authorization.

Table 5-2. Back-End env variables

5.1.1.2. Front-End Implement

Installation Instruction

Step 1. Clone the Repository (SSH - Setup your github account)

```
git clone git@github.com:SOCIAL-NETWORK-2425/social-network-frontend.git
```

Step 2. Install Dependencies

Navigate to the project directory and install all required dependencies using yarn

```
cd social-network-frontend
```

```
yarn install
```

Step 3. Configure Environment Variables

Copy the contents of env.sample to a new .env file in the root directory.

Step 4. Build the project: yarn build

Step 5. Run the application: yarn start

Variables	Description
REACT_APP_BACKEND_URL	The URL of the backend API that the React application will interact with. Typically includes the protocol, host, and port.
REACT_APP_FRONTEND_URL	The base URL or route of the frontend application. Typically set to / to denote the root path.
REACT_APP_CLOUDINARY_URL	The Cloudinary URL used to access uploaded images or resources stored in the Cloudinary account.

Table 5-3. Front-End env variables

5.1.2. Tools

No.	Tool's name	Descriptions
1	Github	Source code management tools
2	Vscode	Text editor - version: 1.95 - Link download: https://code.visualstudio.com/download

3	Microsoft Edge	Browser - version: 97.0.1072.55
4	Intellij IDEA	IDEA coding Back-End - Version: 22.04 - Link download: https://www.jetbrains.com/idea/download/?section=windows
5	Draw.io	Draw schema (mockup & sequence diagram)

Table 5-4. Tool needed for Social Application

5.1.3. TechStack

No.	Technologies	Description
1	PostgresSQL	Databse, storing all data of Social Media Application
2	ReactJS	Building User Interface for application
3	NodeJS	Environment to building Front-End
4	Restful API	Communication protocals for Clients and Servers
5	JsonWeb Token	
6	Redux Saga	State management system
7	Cloudinary	Storing all Image for Social Media Application
8	Spring boot	Server side for Social Media Application

Table 5-5. Technologies Stask of Social Media Application

5.2. Testing

To ensure the system's functionality, reliability, and user experience, we conducted a comprehensive suite of tests, including:

- **Functional Testing:** Verified that each feature, including core functionalities like login, chat, and CRUD operations, adhered to the specified requirements.
- **Boundary Value Analysis:** Assessed the system's behavior at the limits of input values and parameters, such as minimum and maximum allowable inputs and form field constraints.

- **Usabisility Testing:** Evaluated the system's user-friendliness and intuitiveness, focusing on seamless navigation, consistent layout, clear error messages, and overall user experience.
- **Performance Testing:** we test how the system behaves under different load conditions.
    - **Load Testing:** Simulated heavy user loads to assess the system's response times and scalability under stress.
    - **Response Time Testing:** Measured the system's performance in terms of page load times and database query execution speed.
    - **Database Performance Testing:** Evaluated the database's ability to handle large datasets and complex queries efficiently.

Tool Utilization Postman was employed as a tool for functional testing, enabling us to validate API endpoints, verify response accuracy, and compare actual outputs against expected results. This rigorous approach ensured that the backend services met the defined requirements effectively.

5.2.1. Test case

No.	Function	Description
1	Login	User Login with account that user login with wrong username or password on the system
2	Add post to folder	Add post to folder ‘Deafault’ to retrieve after
3	Send friend request	Send request to new user (make new friend request)

Table 5-6. Test case to test function (Login, Add to folder, Send friend request)

5.2.2. Result of Test case

5.2.2.1. Result of test case Login Function

Testcase1.1	
Description	Test case Login function that user have incorrect username or password and trying to access system
Step to work	1. Input Username and Password 2. Click ‘Login’ button
Test Data	Username = ‘duyho03’ Password = ‘12345’

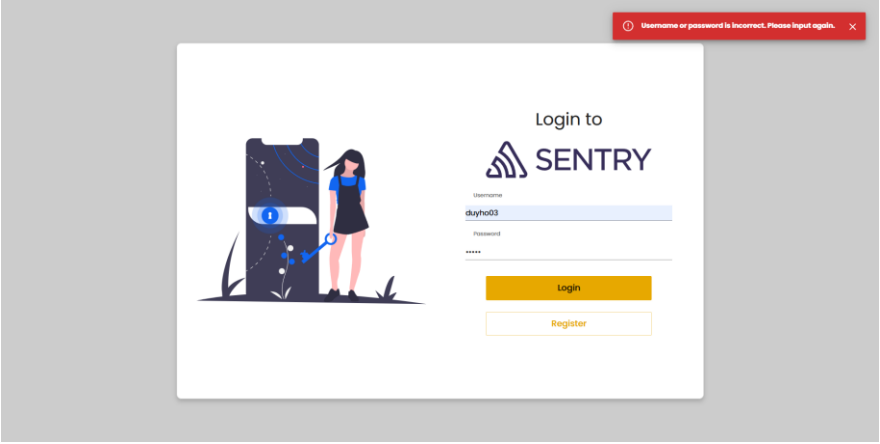
Expected Result	Alert notification: 'Username or password is incorrect. Please input again.'
Actual Result	As expected, result
Conclusion	**Passed**
Test Images	 Figure 5-1. Testcase Login with fail username and password

Table 5-7. Test case Login with wrong username or password

5.2.2.2. Result of test case Add Post to folder Function

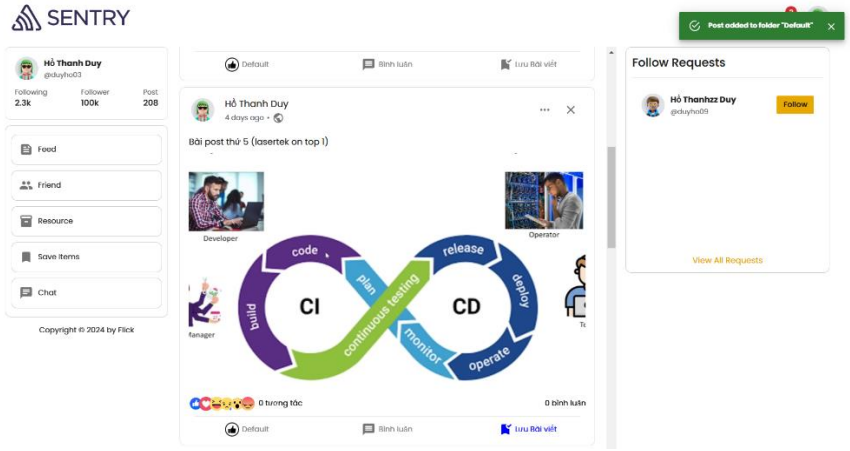
Testcase1.2	
Description	Test case Add post to folder function that user can save post to Default folder to retrieve after
Step to work	1. Input Username and Password 2. Click 'Login' button 4. Click button 'Lưu bài viết'
Test Data	Username = 'duyho03' Password = '123456'
Expected Result	Alert notification: 'Post added to folder "Default"'
Actual Result	As expected result
Conclusion	**Passed**
Test Images	

Figure 5-2. Testcase Save Items successfully

Table 5-8. Test case add post to folder successfully

5.2.2.3. Result of test case Send friend request

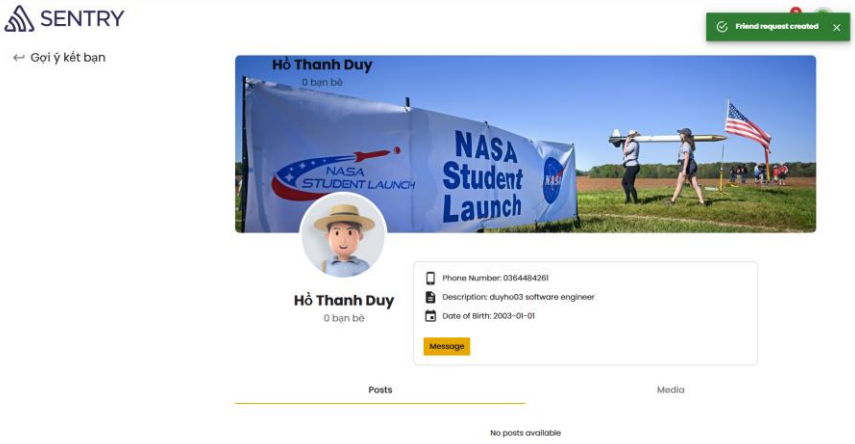
Testcase1.3	
Description	Test case Send friend request function that user can send friend request to other user
Step to work	1. Input Username and Password 2. Click 'Login' button 3. Redirect to friend page and access to 'Gợi ý kết bạn' 4. Click 'add friend' button in any friend's item
Test Data	Username = 'duyho03' Password = '123456'
Expected Result	Alert notification: 'Friend request created'
Actual Result	As expected result
Conclusion	**Passed**
Test Images	

Figure 5-3. Testcase Send Friend Request Successfully

Table 5-9. Test case Send friend request to other user

CHAPTER 6. CONCLUSION

6.1. Achievements

After the process of research and development of the topic "BUILDING A SOCIAL NETWORK SYSTEM". Our team has achieved following results:

6.1.1. Knowledge

- Our team has gained an in-depth understanding of ReactJS, focusing on its application in creating dynamic and user-friendly interfaces for a social networking platform.
- We have mastered the use of Material-UI, enabling the development of responsive and visually appealing designs to enhance the user experience.
- On the backend, we have acquired significant expertise in the Spring Boot framework, which facilitated the creation of a robust and scalable server-side architecture.
- Our knowledge of PostgreSQL has expanded, particularly in terms of efficient data storage and retrieval, ensuring the integrity and security of the application's data.
- The team has prioritized performance optimization, ensuring the system responds quickly and operates reliably under various conditions.

6.1.2. Skills

- The project has greatly enhanced the team's proficiency in full-stack development, particularly in combining ReactJS, Material-UI, Spring Boot, and PostgreSQL technologies.
- Members have developed problem-solving skills, effectively addressing challenges related to database management, backend logic, and frontend responsiveness.
- The collaborative nature of the project has fostered strong teamwork, communication, and project management skills, ensuring smooth and efficient progress.
- The team has also gained expertise in integrating and optimizing core social network features, such as user authentication, messaging, and media uploads.

6.1.3. Product

- The social networking system provides a seamless user experience with its modern, allowing users to navigate the platform effortlessly.
- The backend architecture is designed for scalability and maintainability, enabling smooth operations and facilitating future upgrades or expansions
- Secure login and registration functionalities enhance trust and protect user data, fostering a reliable environment for social interactions.

6.2. Pros and Cons

6.2.1. Pros

- The login and registration systems incorporate advanced encryption protocols, ensuring user data and credentials remain safe.
- The platform includes real-time chatting capabilities, allowing users to communicate instantly, enhancing engagement and fostering dynamic social interactions.
- Users receive instant updates for activities such as friend requests, or post interactions, ensuring they stay informed and engaged.
- The website's dynamic content delivery and intuitive navigation collectively enhance user engagement, providing a more interactive and enjoyable browsing experience.
- The application provides robust administrative tools for managing user accounts, moderating content, and ensuring a safe and organized community environment.

6.2.2. Cons

- Under extremely high user loads, there may be slight latency that requires further optimization.
- Storing and retrieving large media files like high images can increase server costs and require advanced management techniques.
- The application is not fully optimized for dynamic use on mobile phones or tablets, which may hinder the user experience for a significant portion of users who prefer these devices.

6.3. Future work

- Incorporate AI-based features like personalized content recommendations and intelligent chatbots for enhanced user engagement.
- Enable live streaming and improve media file compression and storage mechanisms for better performance.
- Implement multi-factor authentication (MFA) and regular security audits to maintain data integrity.
- Add a video calling feature to enable users to communicate face-to-face, enhancing personal connections and interactions.
- Introduce group features, allowing users to create, join, and manage interest-based groups, fostering community engagement.
- Expand the platform to include e-commerce capabilities, enabling users to buy and sell products directly on the platform, such as offering a marketplace for small businesses.

REFERENCES

[1] Gamma, E., Helm, R., Johnson, R. and Vlissides, J., 1994. *Design patterns*. 1st ed.

[2] Bruegge, B., & Dutoit, A. H. (2010). *Object-oriented software engineering: Using UML, patterns, and Java*. Prentice Hall.

[3] ReactJS Homepage, "*React A JavaScript library for building user interfaces"* 04/01/2016. https://reactjs.org/

[4] NodeJS Learn, "*Introduction to Node.js"* 16/11/2024. https://nodejs.dev/learn

[5] IBM Cloud Learn Hub, "*REST APIs*" 06/4/2021. https://www.ibm.com/cloud/learn/rest-apis

[6] Redux Official Tutorials, *"Redux Essentials, Part 1: Redux Overview and Concepts"* 7/7/2021. https://redux.js.org/tutorials/essentials/part-1-overview-concepts#what-is-redux

[7] Redux Toolkit Guide, "*Redux toolkit - Usage Guide*" 21/11/2024. https://redux-toolkit.js.org/usage/usage-guide

[8] Spring boot documentation, "*Documentation Spring Framework*" 26/11/2024. https://docs.spring.io/spring-boot/index.html

[9] Draw.io Documentation, "*Tutorial full course Draw.io proffesional*" 26/11/2024. https://www.drawio.com/doc/

[10] User Guide to draw.io. Introduction | by Hasali Edirisinghe, "*Guild to draw.io*" 26/11/2024. https://medium.com/@hasaliedirisinghe/user-guide-to-draw-io-6a1a4d7b5d33