



Information Technology Institute
Ministry of Communications and Information Technology



Year 2023/2024

SocialV

Supervised by:

Eng. Mohammed El-Sherbiny

Eng. Salma Abdullah

Eng. Sara Adel

Prepared by

| |
|-------------------------|
| Islam Ahmed Saber |
| Menna Moataz Mansour |
| Mohammed Amgad El-Sayed |
| Nada Emam Hanafy Azoz |
| Nada Mahmoud Mohammed |
| Omar Ashraf Labib |

Graduation Project

Final Documentation

Acknowledgement

We would like to express our great gratitude and immense pleasure at having the opportunity to work with such a dedicated staff of Java Enterprise and Web Applications Development Track. this project would not have been possible without their continuous guidance, insightful advice, and unwavering support throughout the development process.

We would also like to thank our friends, families, for their encouragement, patience and assistance over the years. Especially our parents as we are forever grateful to them, who have always done their best for us.

Finally, for the Information Technology Institute (ITI) and the 9-month scholarship program for providing the suitable environment that leaded us to achieve our best and represent the high standards expected of graduates from the Java Enterprise and Web Applications Development Track.

Table of contents

| | |
|------------------------------------|----|
| Chapter 1 : Introduction | 5 |
| Motivation | 6 |
| Project Objective | 6 |
| Tools..... | 7 |
| Methodology | 8 |
| Chapter 2 : Related work..... | 9 |
| Facebook | 10 |
| Chapter 3 : System Analysis..... | 11 |
| Project specification | 12 |
| System Architecture | 12 |
| Functional Requirements..... | 13 |
| Non-functional requirement | 15 |
| Chapter 4 : System Design..... | 17 |
| Class Diagrams..... | 18 |
| Database Schemas and Diagrams..... | 27 |
| Chapter 5 : Future work | 33 |
| Chapter 6 :User Guide | 35 |
| Chapter 7 : References | 41 |

Table of figures

| | |
|--|----|
| Figure 1 - Agile methodology..... | 8 |
| Figure 2 – Facebook..... | 10 |
| Figure 3 - System Architecture..... | 12 |
| Figure 4 - Api Gateway Class Diagram | 18 |
| Figure 5 - User Service Class Diagram | 19 |
| Figure 6 - Authentication Server Class Diagram | 19 |
| Figure 7 - Post Service Class Diagram | 20 |
| Figure 8 - Like Service Class Diagram | 21 |
| Figure 9 - Comment Service Class Diagram | 22 |
| Figure 10 - Search Service Class Diagram | 23 |
| Figure 11 - Upload Service Class Diagram | 23 |
| Figure 12 - Message Service Class Diagram | 24 |
| Figure 13 - Notification Service Class Diagram..... | 25 |
| Figure 14 - Friend Service Class Diagram..... | 26 |
| Figure 15 - ERD Diagram | 27 |
| Figure 16 - Graph Relationships in Friend Service..... | 32 |
| Figure 17 - Sign up..... | 36 |
| Figure 18 - Login | 36 |
| Figure 19 - Home Page | 37 |
| Figure 20 – Search friends | 37 |
| Figure 21 - sending friend request..... | 37 |
| Figure 22 - Accepting friend request..... | 38 |
| Figure 23 - Chatting with friend..... | 38 |
| Figure 24 - Display profile | 39 |
| Figure 25 - Edit profile | 40 |
| Figure 26 - Reflect the changes on profile | 40 |

Chapter 1 : Introduction

Motivation

In today's digital age, social media platforms have become integral to our daily lives, enabling us to connect, communicate, and share content with ease. The motivation behind this project stems from the desire to create a comprehensive and scalable social media platform. By addressing common challenges such as scalability, performance, and maintainability, this project aims to deliver a seamless and efficient user experience. Furthermore, the project emphasizes enhancing real-time communication and data processing capabilities, which are essential for a dynamic social media environment. The ultimate goal is to provide users with a reliable, feature-rich platform that can grow and adapt to their needs.

Project Objective

The primary objective of this project is to develop a fully functional social media platform using a microservices architecture. The project aims to deliver a scalable, maintainable, and high-performance platform that will include the following features:

- ❖ User Service: Manage user profiles and authentication.
- ❖ Post Service: Allow users to create posts.
- ❖ Like Service: Enable users to interact with posts by defined set of reactions.
- ❖ Comment Service: Facilitate user comments on posts.
- ❖ Friend Service: Manage friend requests and connections.
- ❖ Messaging Service: Provide real-time messaging capabilities.
- ❖ Notification Service: Notify users about various activities and updates.
- ❖ Auth Service: Handle user authentication and authorization.
- ❖ Search Service: Allow users to search for content and other users.
- ❖ API Gateway and Service Discovery: Ensure efficient communication and coordination between microservices.
- ❖ Upload Service: Handle uploading user's images and posts media.

Tools

- ❖ **GitHub**: for software development and version control.
- ❖ **Angular**: to build the application's front end.
- ❖ **Firebase**: We used cloud Firestore to store the assets.
- ❖ **Spring Boot**: Simplifies the development of standalone, production-grade Spring applications.
- ❖ **Spring Cloud**: Provides tools for quickly building microservices.
- ❖ **Spring Data JPA & Mongo**: Facilitates data access and management with JPA (relational databases) and MongoDB (NoSQL databases).
- ❖ **Spring Web**: Supports the development of web applications.
- ❖ **Spring Security**: Ensures application security, including authentication & authorization.
- ❖ **Kafka**: A distributed streaming platform for real-time data processing.
- ❖ **WebSocket**: Enables real-time, bidirectional communication between the client and server.
- ❖ **API Gateway**: Manages and routes requests to the appropriate microservices.
- ❖ **Service Discovery**: Facilitates the automatic detection of microservices in a distributed system.
- ❖ **Postman**: for testing the back-end server.
- ❖ **Swagger**: Generates interactive and user-friendly API documentation that allows developers to explore and test endpoints directly from the documentation interface.

Methodology

The development of this social media platform was carried out using the Agile methodology, which emphasizes iterative progress, collaboration, and flexibility in the software development process. Agile allowed the team to adapt to changes quickly and deliver incremental improvements throughout the project lifecycle.

Key Elements of Our Agile Methodology:

- ❖ **Daily Stand-ups:** Daily stand-up meetings were held to ensure effective communication among team members. Each team member shared updates on their progress, discussed any roadblocks, and coordinated efforts to address challenges.
- ❖ **Incremental Development:** The project was developed in small, manageable increments. Each sprint focused on developing specific features or components, allowing for continuous integration and testing. This approach ensured that the platform was progressively built and improved upon.
- ❖ **Collaborative Approach:** Team members worked closely together, leveraging each other's expertise. Regular collaboration sessions and code reviews facilitated knowledge sharing and improved the overall quality of the codebase.

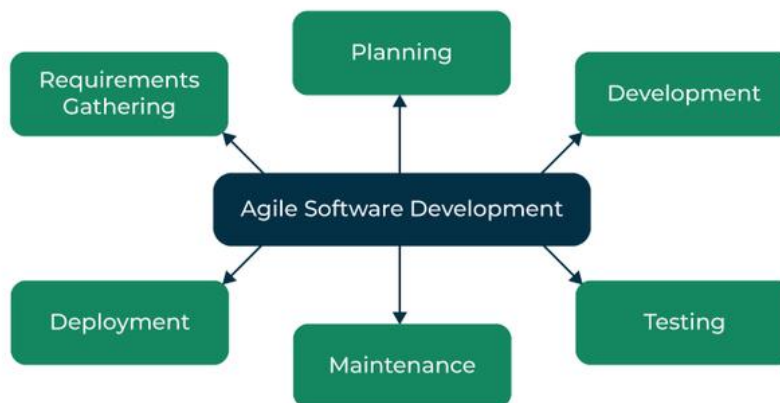


Figure 1 - Agile methodology

Chapter 2 : Related work

Facebook

Facebook is one of the largest and most popular social media platforms globally. It provides a comprehensive range of features including user profiles, news feeds, messaging, and notifications. Facebook's backend architecture has evolved over the years to handle massive user loads and provide real-time updates. Our project aspires to achieve a similar level of scalability and real-time capabilities as Facebook by leveraging a microservices architecture.

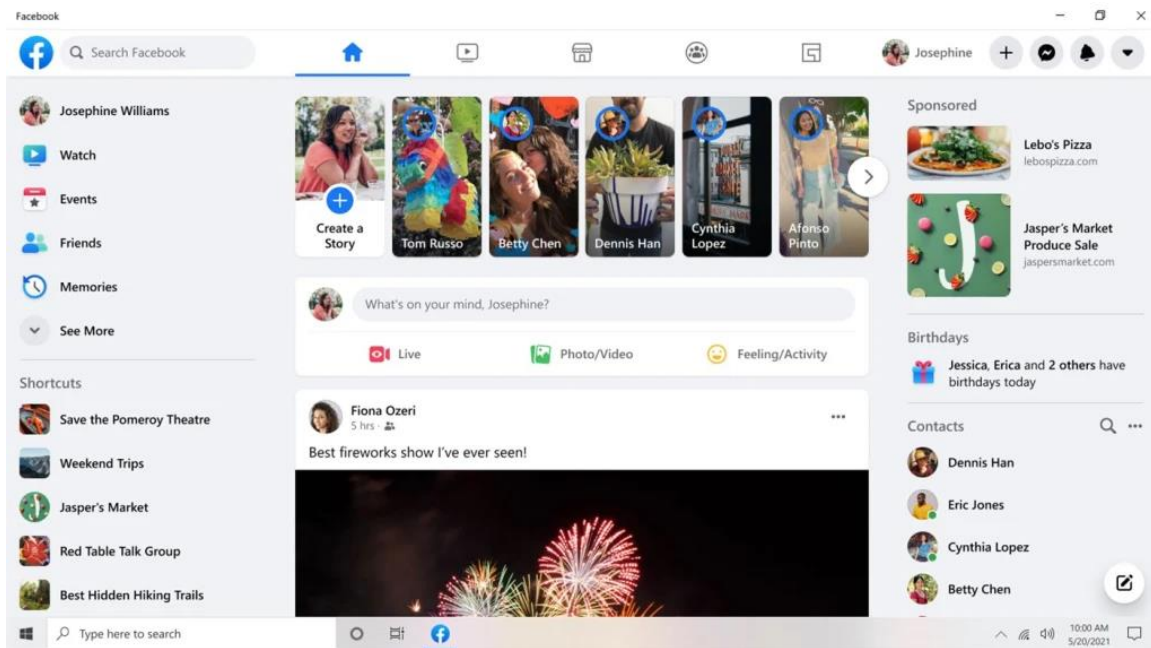


Figure 2 – Facebook

Chapter 3 : System Analysis

Project specification

System Architecture

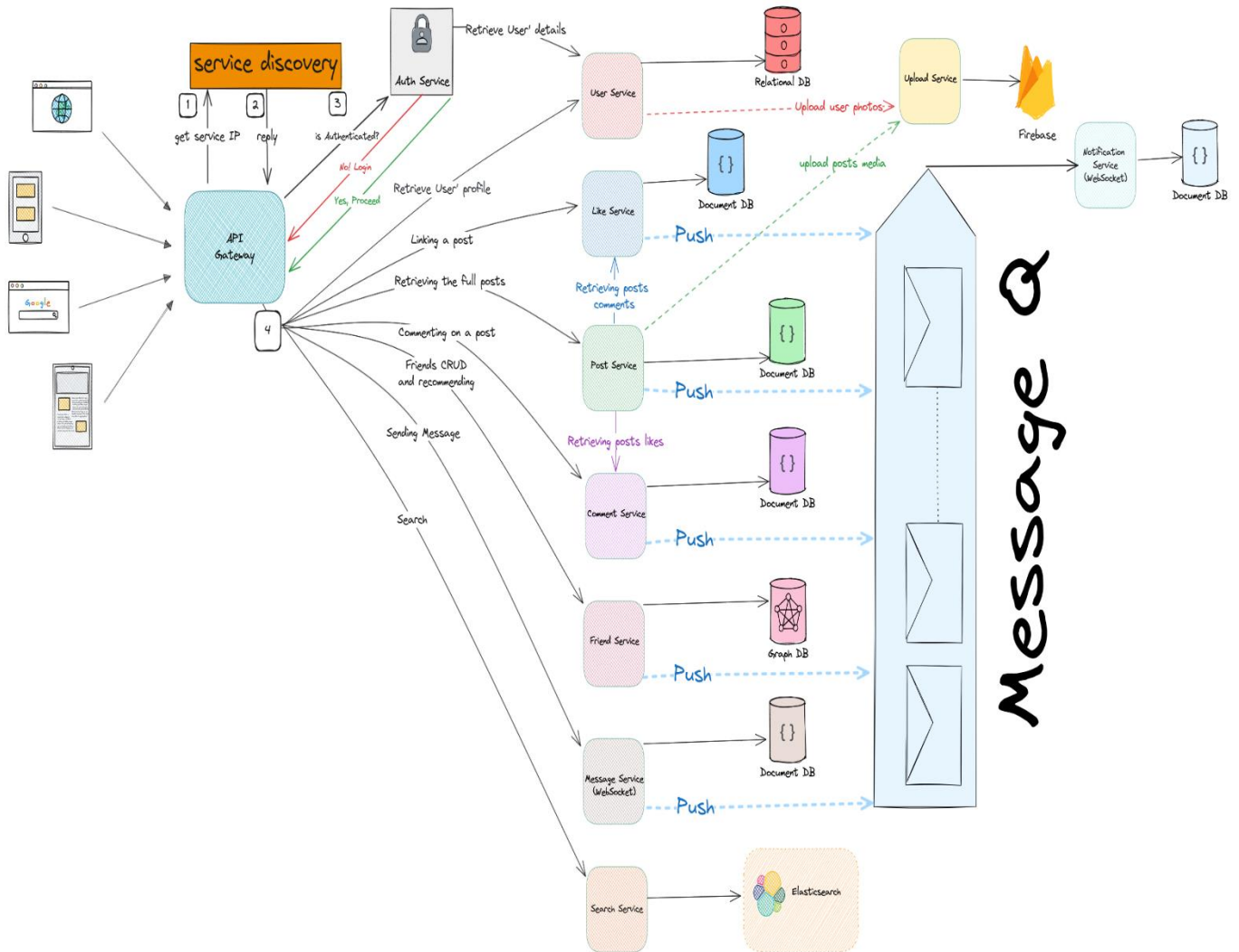


Figure 3 - System Architecture

Functional Requirements

1. Authentication and Authorization

- ❖ User Registration: Users should be able to register with an email, username and password.
- ❖ User Login: Users should be able to log in with their email and password.

2. Post management

- ❖ Create Post: Users should be able to create posts with text, images, and videos.
- ❖ Like/Unlike Post: Users should be able to like and unlike posts.
- ❖ Comment on Post: Users should be able to comment on posts.
- ❖ View Post: Users should be able to view posts from friends and public posts.

3. Search functionality

- ❖ Search Users: Users should be able to search for other users by name, username, or email.

4. Friend Management:

- ❖ Send Friend Request: Users should be able to send friend requests to other users.
- ❖ Accept/Reject Friend Request: Users should be able to accept or reject friend requests.
- ❖ Remove Friend: Users should be able to remove friends.
- ❖ View Friends List: Users should be able to view their list of friends.

5. Notification Management:

- ❖ **Real-time Notifications:** Users should receive real-time notifications for likes, comments, friend requests.
- ❖ **Notification History:** Users should be able to view their notification history.

6. Messaging:

- ❖ **Send Message:** Users should be able to send messages to their friends.
- ❖ **Receive Message:** Users should be able to receive messages from their friends.
- ❖ **Message Notifications:** Users should be notified of new messages.
- ❖ **View Conversation History:** Users should be able to view their conversation history with friends.

Non-functional requirement

1. Response Time:

- ❖ The application does not take more than 5 seconds to validate the user credential.
- ❖ A user obtains their search results in less than 10 seconds after typing and clicking on the search button.
- ❖ The overall average latency of the system is 10 seconds or less, or else users will lose focus and will switch to other applications while this application is still processing.

2. Scalability:

- ❖ With the rising workload and demand for the application's services, the application should be able to grow and be able to handle the increases of users.
- ❖ The architecture should support horizontal scaling to handle increased load.
- ❖ Microservices should be able to scale independently based on their load.

3. Usability:

- ❖ The application has an easy-to-use interface .
- ❖ The Application has a decent look and feels that encourages the user to use the application.
- ❖ The colors used in the application are not dull but don't strain the eyes.
- ❖ Voice commands in the app helper make the searching and buying process easier.

4. Reliability:

- ❖ Robustness: The system has the ability to respond and handle invalid inputs with error messages and exception handling.

5. Supportability:

a. Adaptability:

- The system must have the ability to accept new additions of features and concepts.

b. Maintainability:

- The system must have the ability to change to deal with new technologies or fix defects.

6. Security:

- a.** Email verification for user registering.
- b.** Password Hashing: Store user passwords securely using strong hashing algorithms such as Bcrypt.
- c.** Password Policies: Enforce strong password policies (e.g., minimum length, complexity requirements) to enhance security.
- d.** Secure Storage: JWTs and refresh tokens should be securely stored on the client side.
- e.** Token Expiry: JWTs should have a defined expiry time to enhance security. Tokens should be short-lived (e.g., 15 minutes) and accompanied by a refresh token mechanism.

7. Extensibility:

- a.** API Versioning: Implement API versioning to ensure backward compatibility as new features and updates are introduced.

Chapter 4 : System Design

Class Diagrams

1. Api Gateway

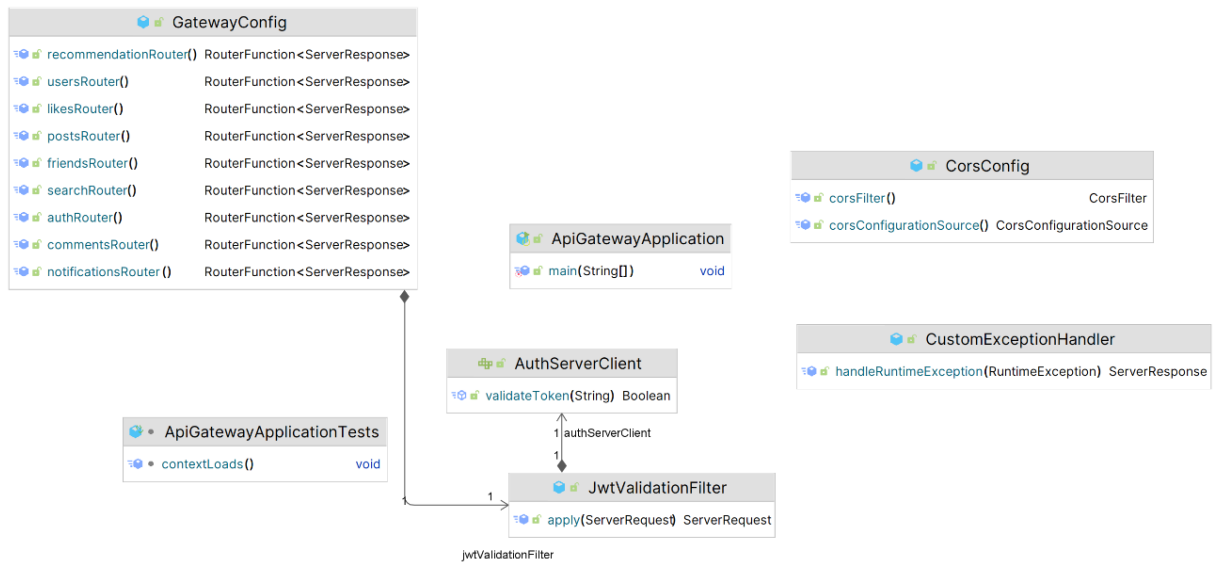


Figure 4 - Api Gateway Class Diagram

2. User Service

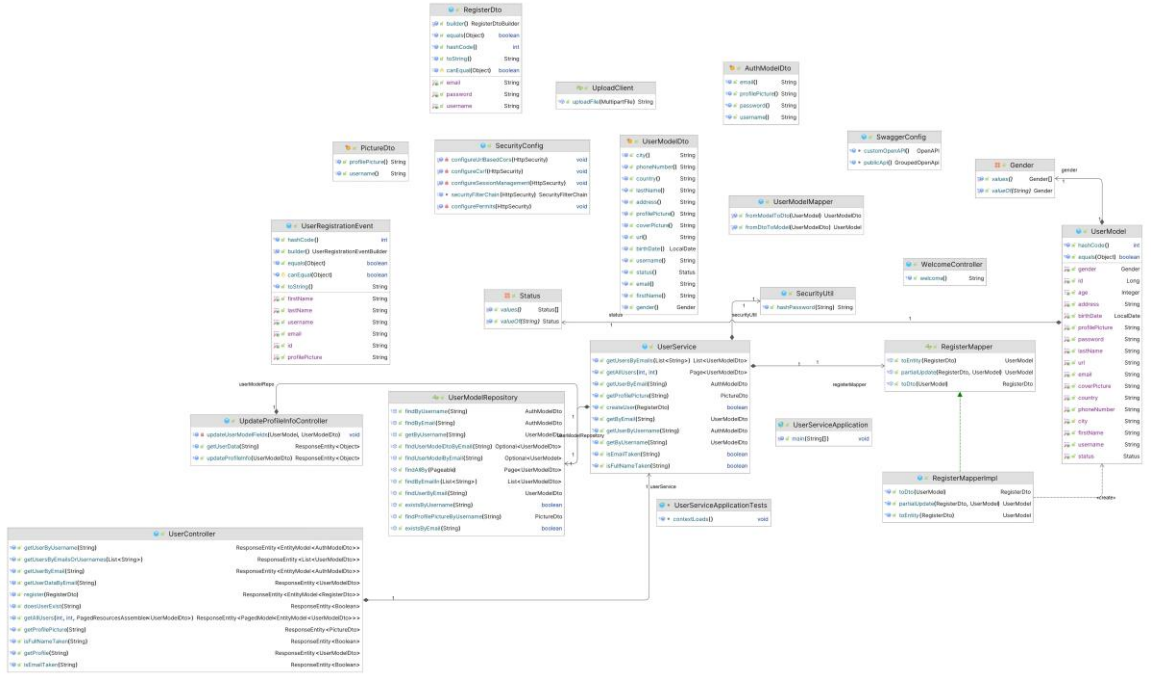


Figure 5 - User Service Class Diagram

3. Authentication Server

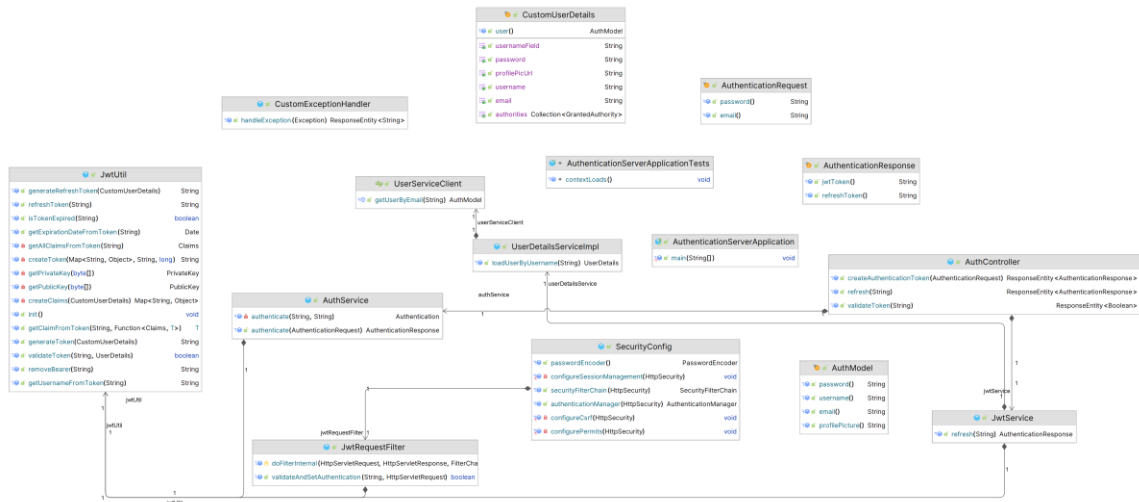


Figure 6 - Authentication Server Class Diagram

4. Post Service

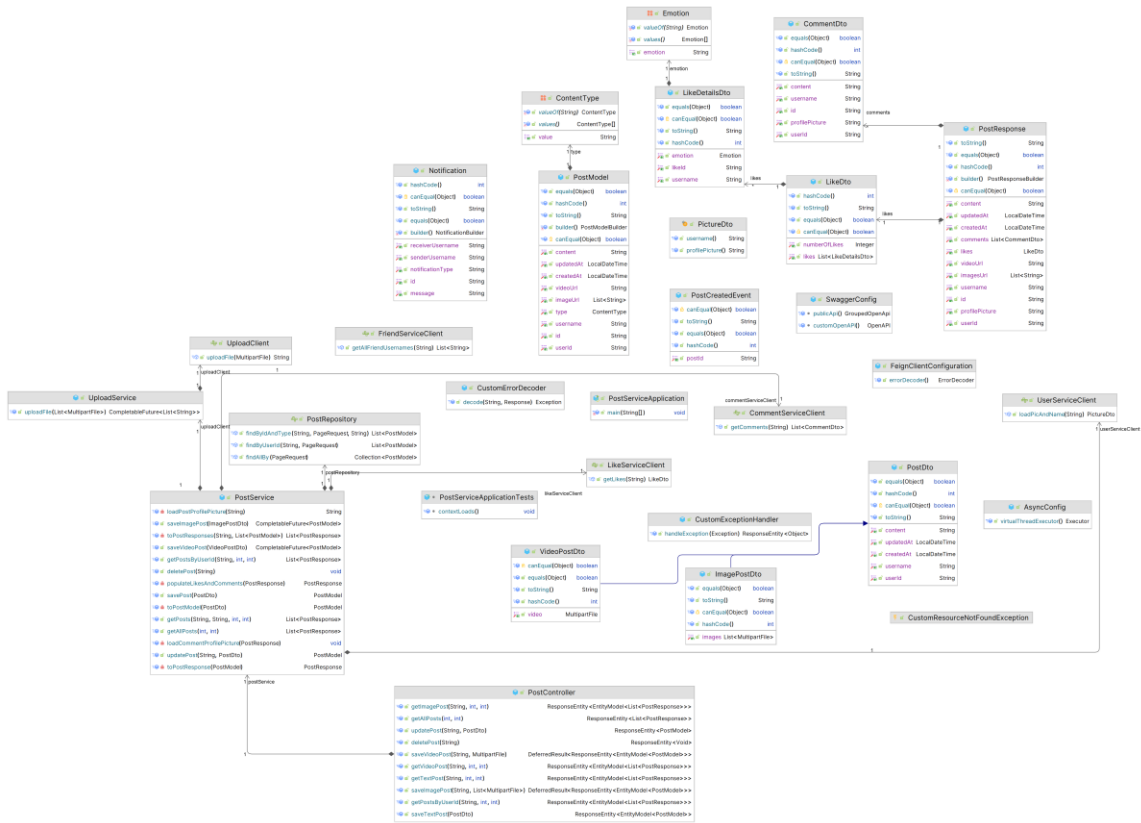


Figure 7 - Post Service Class Diagram

5. Like Service

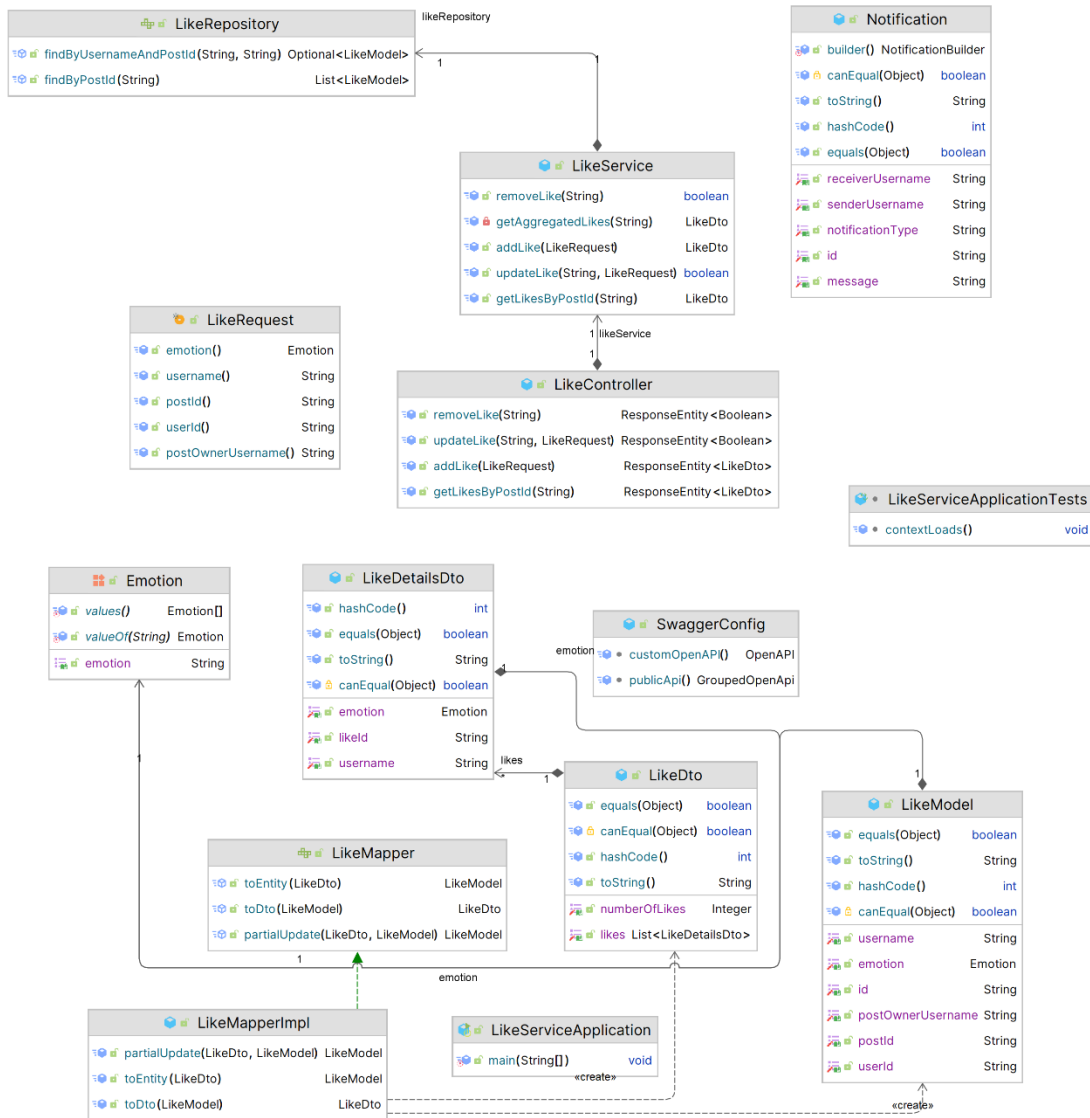


Figure 8 - Like Service Class Diagram

6. Comment Service

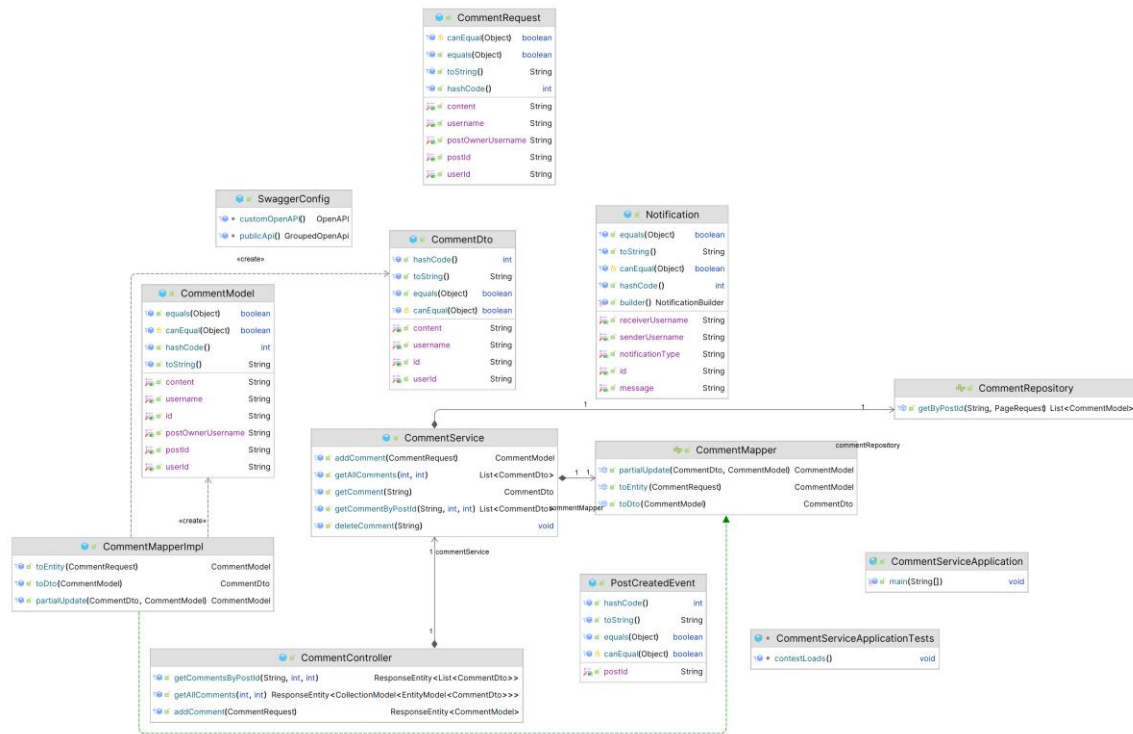


Figure 9 - Comment Service Class Diagram

7. Search service

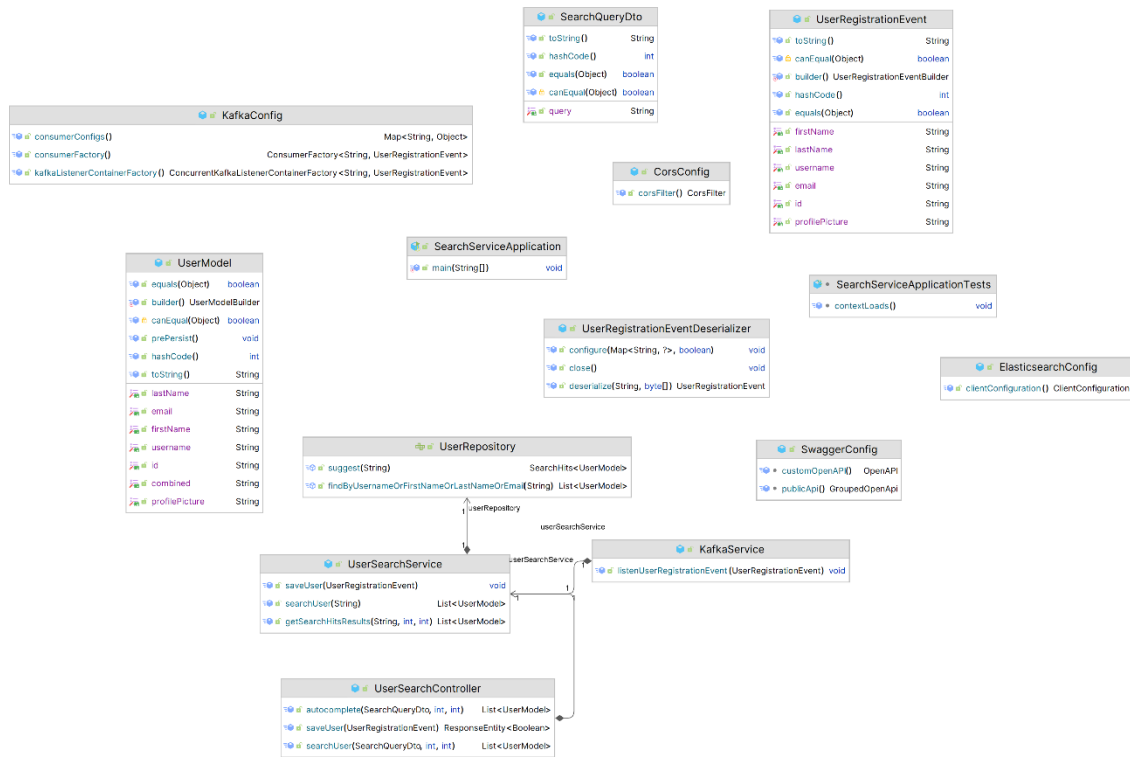


Figure 10 - Search Service Class Diagram

8. Upload Service

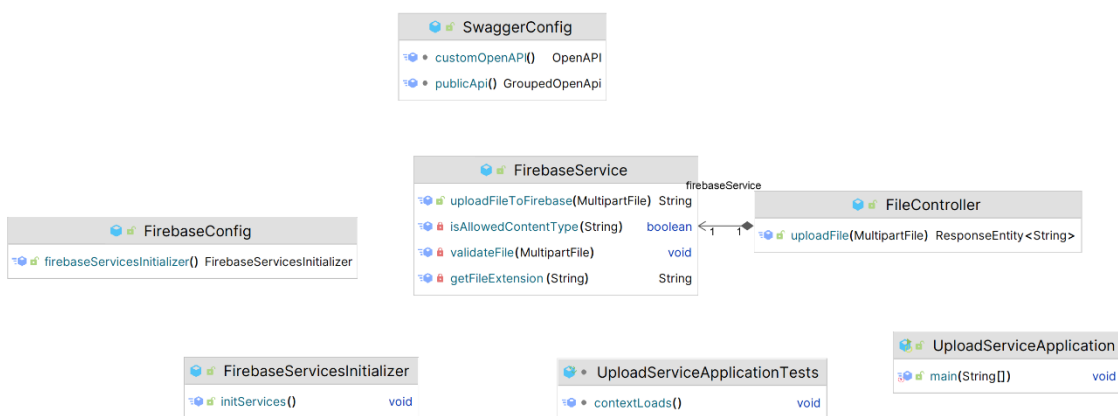


Figure 11 - Upload Service Class Diagram

9. Message service

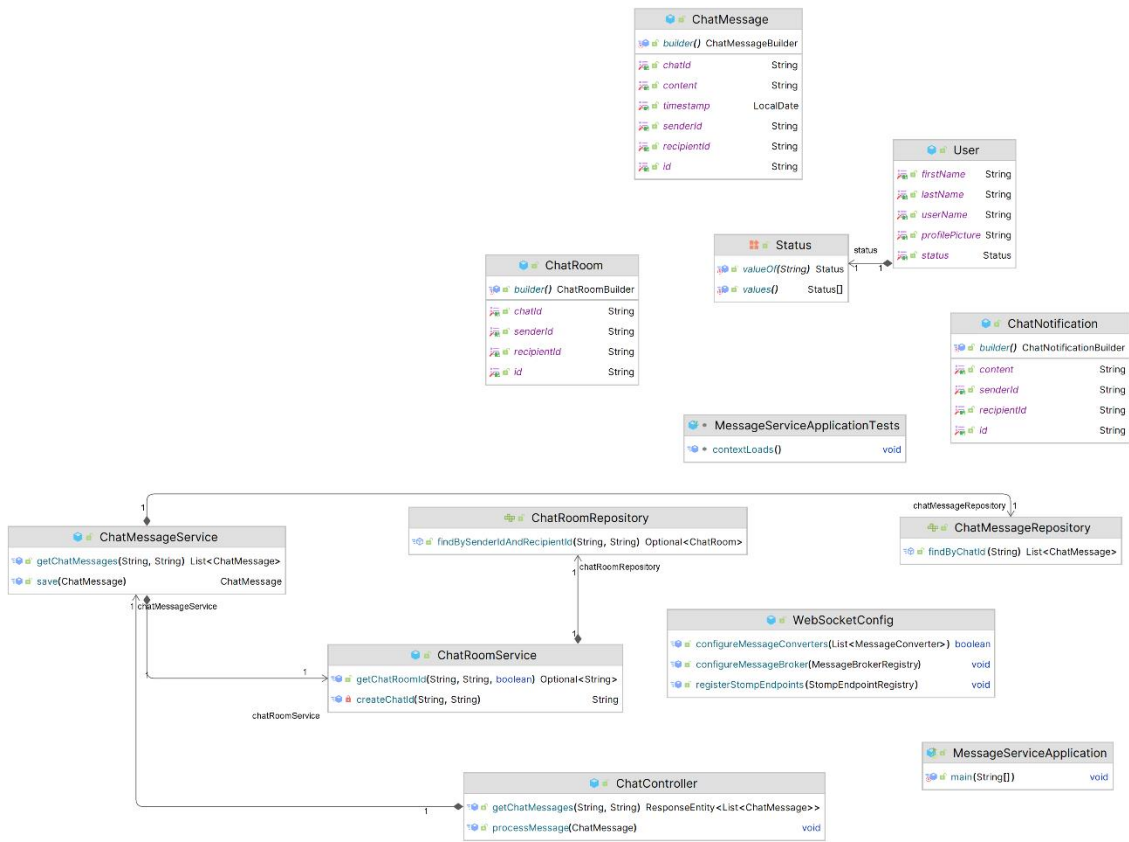


Figure 12 - Message Service Class Diagram

10. Notification Service



Figure 13 - Notification Service Class Diagram

11. Friend Service

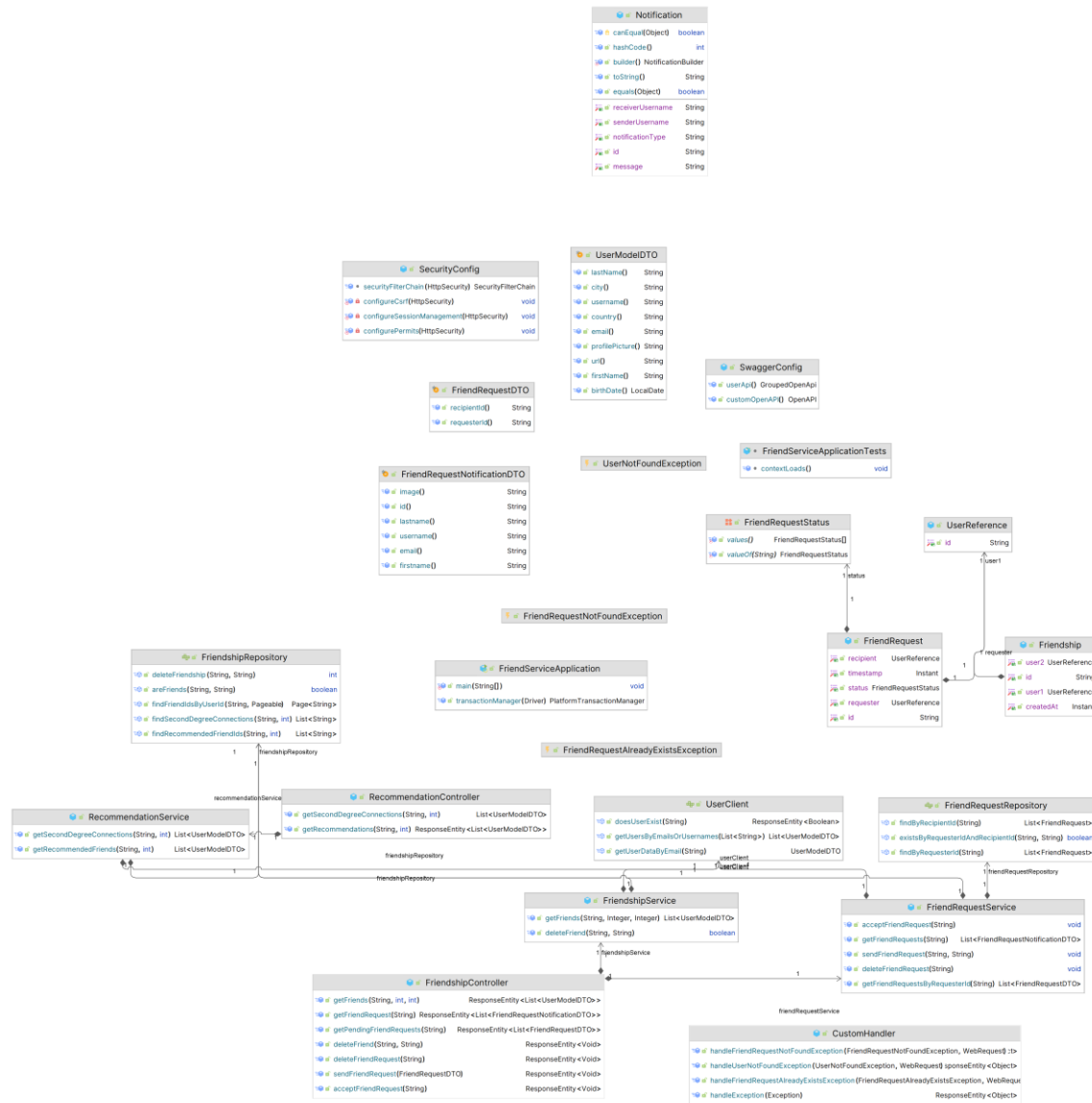


Figure 14 - Friend Service Class Diagram

1- ERD diagram

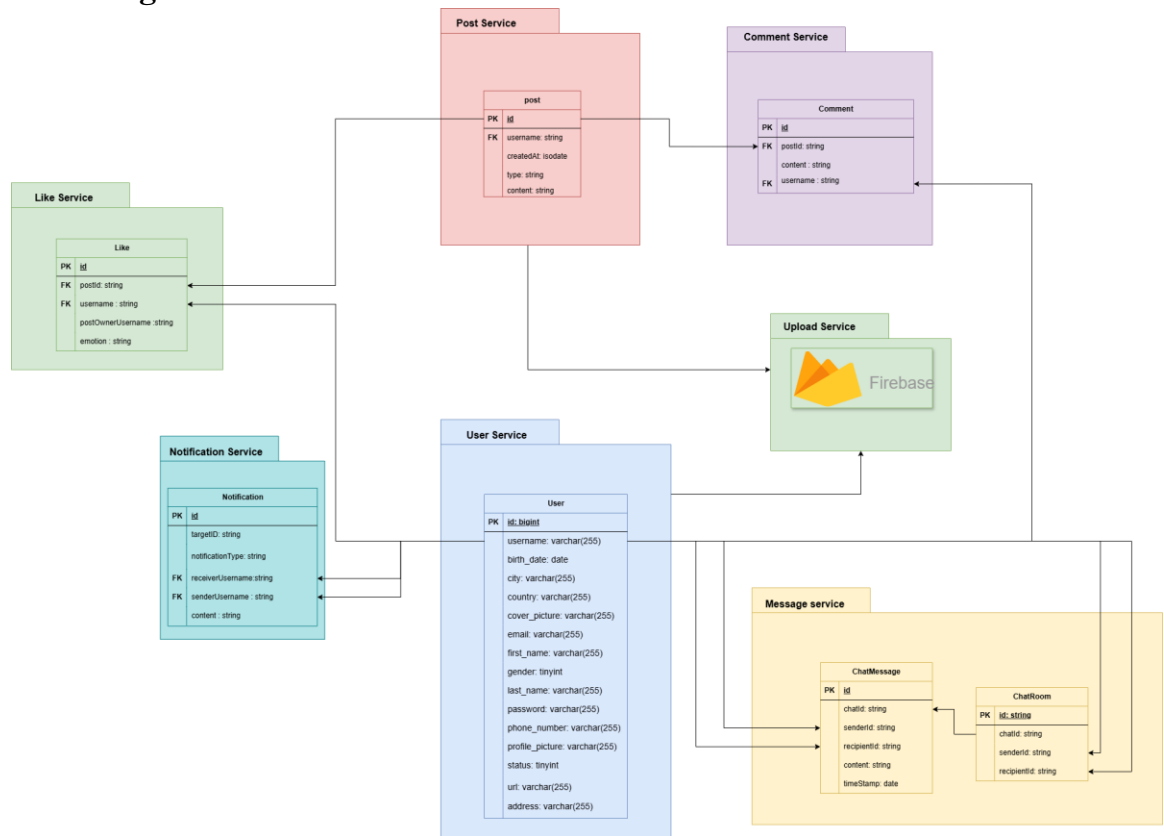


Figure 15 - ERD Diagram

2- Graph DB Schema

This schema represents a social networking model involving users (UserReference), friendships (Friendship), and friend requests (FriendRequest). It captures the essential entities and relationships in the social networking graph.

Entities:

1. UserReference

- **Description:** Represents a user in the network.
- **Count:** 20 instances.
- **Properties:**
 - **id:**
 - **Type:** STRING
 - **Unique:** No
 - **Indexed:** No
 - **Existence Constraint:** No
- **Relationships:**
 - **REQUESTED_TO:**
 - **Direction:** Inbound
 - **Target Node:** FriendRequest
 - **Count:** 13
 - **FRIEND_OF:**
 - **Direction:** Outbound
 - **Target Node:** Friendship
 - **Count:** 13
 - **REQUESTED_BY:**
 - **Direction:** Outbound
 - **Target Node:** FriendRequest
 - **Count:** 0

2. Friendship

- **Description:** Represents a friendship relationship between users.
- **Count:** 20 instances.
- **Properties:**
 - **id:**
 - **Type:** STRING
 - **Unique:** No
 - **Indexed:** No
 - **Existence Constraint:** No
 - **created_at:**
 - **Type:** DATE_TIME
 - **Unique:** No
 - **Indexed:** No
 - **Existence Constraint:** No
- **Relationships:**
 - **FRIEND_OF:**
 - **Direction:** Outbound
 - **Target Node:** UserReference
 - **Count:** 20

3. FriendRequest

Description: Represents a friend request from one user to another.

- **Count:** 13 instances.
- **Properties:**
 - **id:**
 - **Type:** STRING
 - **Unique:** No
 - **Indexed:** No
 - **Existence Constraint:** No
 - **status:**
 - **Type:** STRING
 - **Unique:** No
 - **Indexed:** No
 - **Existence Constraint:** No
 - **createdAt:**
 - **Type:** DATE_TIME
 - **Unique:** No
 - **Indexed:** No
 - **Existence Constraint:** No
 - **created_at:**
 - **Type:** DATE_TIME
 - **Unique:** No
 - **Indexed:** No
 - **Existence Constraint:** No
- **Relationships:**
 - **REQUESTED_TO:**
 - **Direction:** Outbound
 - **Target Node:** UserReference
 - **Count:** 0
 - **REQUESTED_BY:**
 - **Direction:** Inbound
 - **Target Node:** UserReference
 - **Count:** 13

Relationships:

1. FRIEND_OF

- **Type:** Relationship
- **Count:** 40 instances
- **Properties:** None
- **Direction:** Bidirectional (between UserReference nodes, managed through Friendship nodes)

2. REQUESTED_TO

- **Type:** Relationship
- **Count:** 13 instances
- **Properties:** None
- **Direction:** Outbound (from FriendRequest to UserReference)

3. REQUESTED_BY

- **Type:** Relationship
- **Count:** 13 instances
- **Properties:** None
- **Direction:** Inbound (to FriendRequest from UserReference)

The graph shows a network of individuals and their relationships. Nodes are represented by colored circles with text inside, and edges are labeled with relationship types.


- Nodes:**
 - Teal nodes: "nada5@gmail...", "nada4@gmail...", "nada3@gmail...", "nada2@gmail...", "nada1@gmail...", "nada0@gmail..."
 - Pink nodes: "2024-06-23T14...", "2024-06-22T07...", "2024-06-22T03...", "2024-06-22T03...", "2024-06-22T02...", "2024-06-22T02...", "2024-06-24T11...", "2024-06-24T15...", "2024-06-24T11...", "2024-06-24T15..."
 - Blue nodes: "b1a6c1 77-644...", "6853a8 59-3e6d...", "6635c6 2a0-496..."
- Relationships (Edges):**
 - "FRIEND OF": Connects teal nodes to pink nodes, and pink nodes to teal nodes.
 - "REQUESTED TO": Connects blue nodes to teal nodes.


Figure 16 - Graph Relationships in Friend Service

Chapter 5 : Future work

- Implement voice/video chat.
- Implement edit/delete posts.
- Implement edit/delete comments.
- Apply reactions on comments.
- Enhance GUI according to UX.
- Apply general search so that we can search on: posts and comments.
- Add notification for the new messages

Chapter 6 : User Guide





Find new friends

It is a long established fact that a reader will be distracted by the readable content.

Sign Up

Enter your email address and password to access admin panel.

Username

menna

Email address

menna@gmail.com

First name

Menna

Last name


Mansour


Password

☒ I accept Terms and Conditions

Sign Up

Figure 17 - Sign up





Find new friends

It is a long established fact that a reader will be distracted by the readable content.

Sign in

Enter your email address and password to access admin panel.

Email address

menna@gmail.com

Password

[Forgot password?](#)

☐ Remember Me

Sign in

Don't have an account? [Sign up](#)

G

Figure 18 - Login

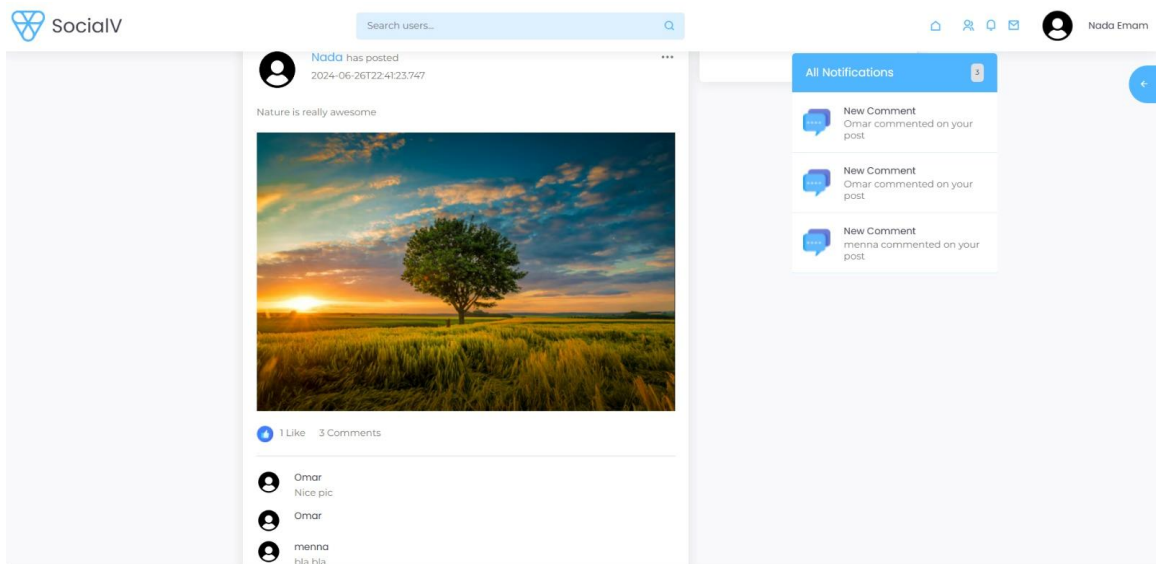


Figure 19 - Home Page

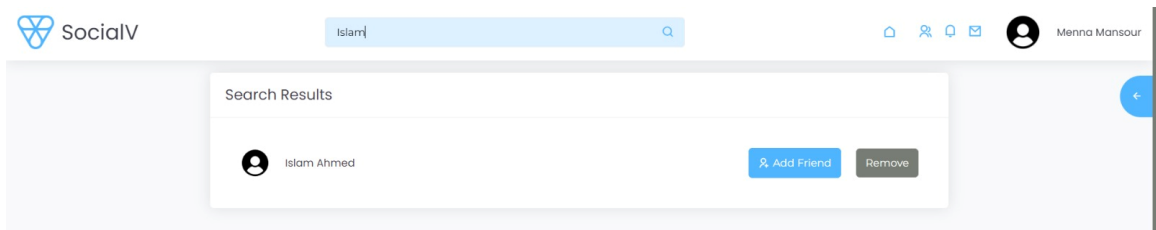


Figure 20 – Search friends

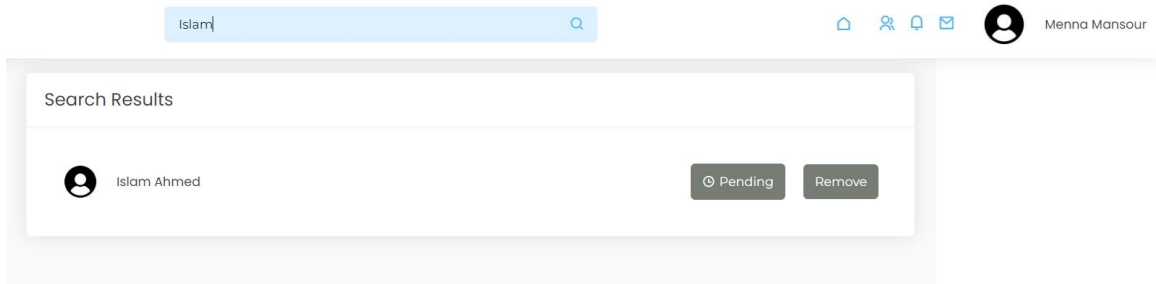


Figure 21 - sending friend request

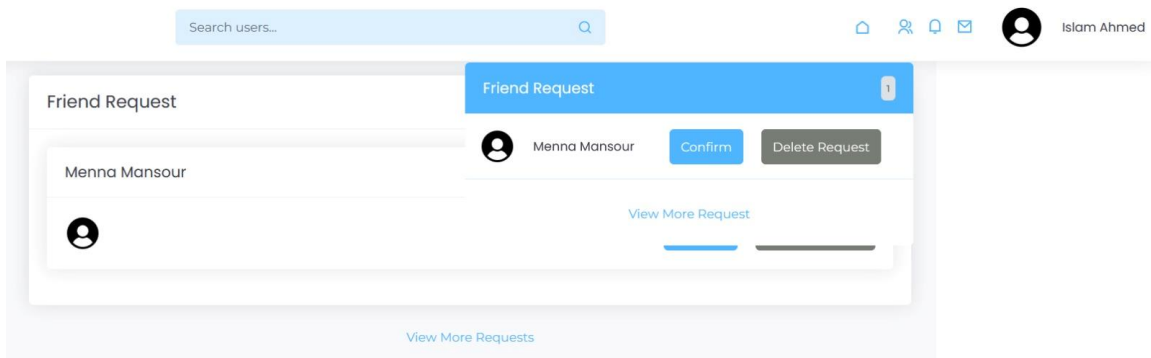


Figure 22 - Accepting friend request

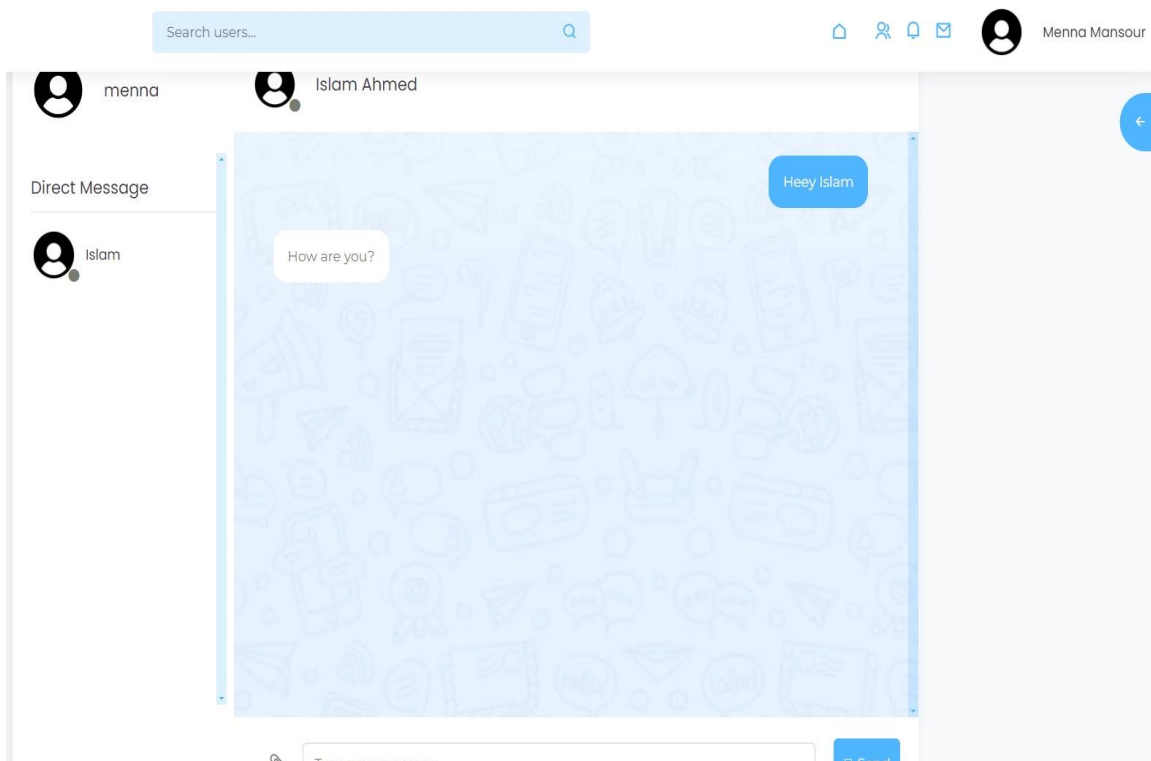


Figure 23 - Chatting with friend

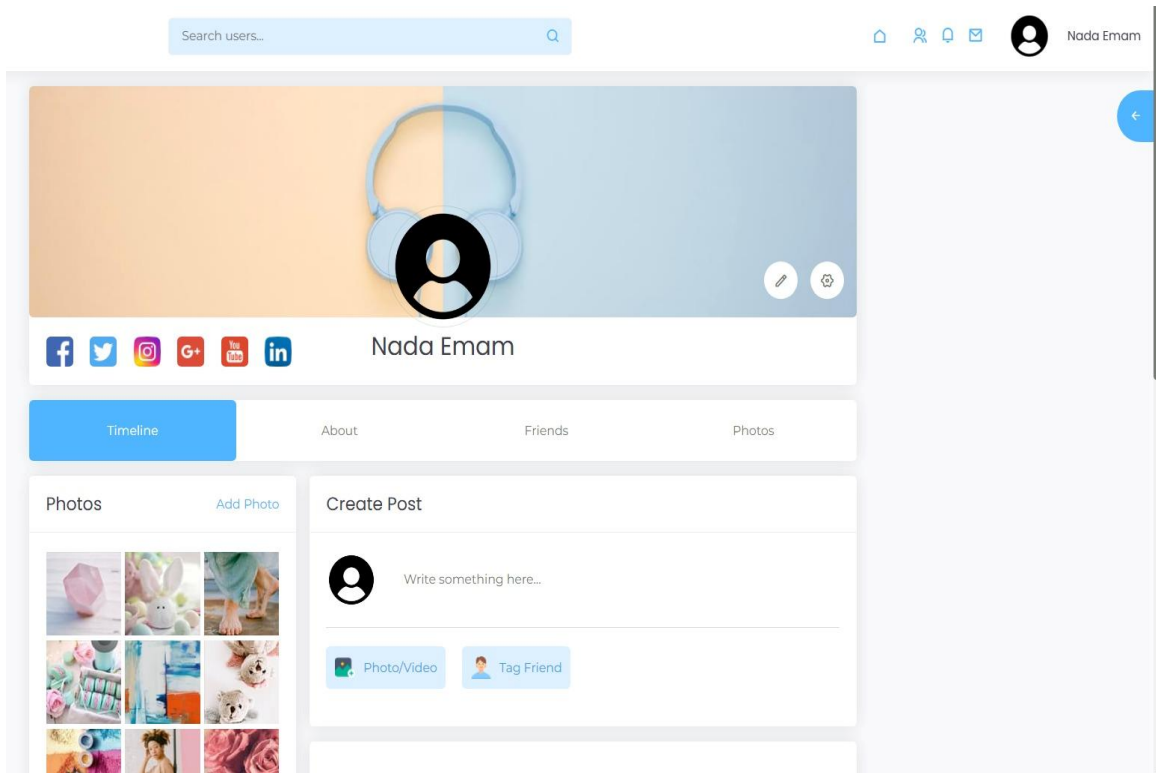



Figure 24 - Display profile

Search users...



First Name:

Nada

Last Name:

Emam

User Name:

Nada

City:

Giza

Gender:

☐ Male
 ☐ Female

Date Of Birth:

mm/dd/yyyy

Marital Status:

Country:

Enter country






Address:

Submit


Cancel



Figure 25 - Edit profile







Search users...

Nada Emam



Nada Emam

Timeline










About

Friends


Photos

Photos

Add Photo

Create Post



Write something here...

Photo/Video

Tag Friend

Figure 26 - Reflect the changes on profile

Chapter 7 : References

1. Collaborative whiteboarding made easy. Excalidraw. (n.d.). <https://excalidraw.com/>
2. ChatGPT. (n.d.). from <https://chatgpt.com>
3. Spring Framework reference documentation. (n.d.). <https://docs.spring.io/spring-framework/reference/index.html>