Final Project Report

Social App - Group 29 (FCS Project)

Sharad Jain (MT24138), Akash Singh (MT24110)

---

1. Introduction

We have named our social media platform a 'Social App.' This project we developed as part of the course to build a secure and scalable social media platform with end-to-end encrypted messaging, user verification, secure media sharing, and admin moderation capabilities. This platform brings connection creation capabilities to users with known and unknown persons regardless of distance, and it is designed to protect user privacy while offering essential social features like group and individual messaging.

2. Objectives

- Implement secure one-to-one and group messaging
- Enable secure media (file/image) sharing
- Validate user's identity with OTP-based registration
- Admin dashboard for moderation and approval of users
- Ensures the core web security standards (XSS, SQL Injection, etc.)

3. Tech Stack

- **Frontend:** Angular (TypeScript)
- **Backend:** Java (Spring Boot)
- **Database:** MySQL
- **Web Server:** Apache Tomcat (Spring Boot), Nginx (Angular hosting)
- **Platform:** Ubuntu (VM)

Angular Installation Steps

1. **Install Node.js:** Download and install Node.js from the official website: `https://nodejs.org`.
2. **Verify Node.js and npm:** Open a terminal and run:
   - node -v
   - npm -v

3. **Install Angular CLI:** In your terminal, run:
   - npm install -g @angular/cli
4. **Verify Angular CLI Installation:** Run the following command:
   - ng version

Java23 Installation

1. **Download Installer:**
   Visit the official Java website (such as Oracle or OpenJDK) and download the Java23 installer for your operating system.
2. **Run Installer:**
   Execute the downloaded file and follow the installation prompts. Accept the license agreement when prompted.
3. **(Optional) Set Environment Variables:**
   - **Windows:** Update your environment variables by setting `JAVA_HOME` to your installation folder (e.g., `C:\Program Files\Java\jdk-23`) and add `%JAVA_HOME%\bin` to your `PATH`.
   - **Mac/Linux:** Open your terminal and edit your shell configuration file (like `.bashrc` or `.zshrc`) to add:
     - `export JAVA_HOME=/path/to/jdk-23`
     - `export PATH=$JAVA_HOME/bin:$PATH`

     Save the changes and refresh your terminal session.
4. **Verification:**
   Open Command Prompt or Terminal and run: java -version

   Confirm that the output shows Java23 as the installed version.
5. **Additional Note:**
   You can choose a custom installation directory if needed, and a system restart may be required for environment variable changes to take effect.

MySQL Installation

1. **Download Installer:**
   Visit the official MySQL website and download the MySQL Community Server installer for your operating system.
2. **Run Installer:**
   Execute the installer file. During the setup, set a root password and configure necessary options such as selecting the default port (usually 3306).

3. **Start MySQL Service:**

   - **Windows/Mac:** The service may automatically start after installation.
   - **Linux:** Open your terminal and run:*"sudo systemctl start MySQL"*

4. **Verification:** Open Command Prompt or Terminal and log in by running:

$$MySQL - uroot - p \tag{1}$$

   Enter the root password to confirm that the MySQL server is running correctly.

5. **Additional Note:**
   During the installation, choose between different setup types (like Developer Default or Server Only) based on your needs. For troubleshooting, consult the official documentation or check the installation logs for any errors.

Apache Tomcat (Spring Boot Backend)

Install Java JDK:

- Ubuntu: `sudo apt install openjdk-17-jdk`

Install Maven:

- Ubuntu: `sudo apt install maven`

Clone the Spring Boot project or copy the code.
Go to the project folder and run:

- mvn clean install
- java -jar target/app.jar

Spring Boot app will start on: `http://localhost:8080`

Nginx (Angular Frontend Hosting)

1. Install Node.js and npm:

   - Ubuntu: `sudo apt install nodejs npm`

2. Install Angular CLI:

```
npm install -g @angular/cli
```

3. Go to the frontend project folder and run:

```
npm install
ng build --prod
```

4. Install Nginx:
   - Ubuntu: `sudo apt install nginx`
5. Copy the `dist/` folder files to:

   `/var/www/html`

6. Open and edit Nginx config file:

   `sudo nano /etc/nginx/sites-available/default`

   Change the root path to:

   ```
   root /var/www/html;
   index index.html;
   ```

7. Restart Nginx:

   `sudo systemctl restart nginx`

8. Now access the frontend at: `http://localhost`

4. Features Implemented

a. User Authentication & OTP Verification

- Sign up using the email, and the verification is done using OTP.
- Login is done by backend verification, and the user gets logged in.
- The time period of being logged on the website is 1 hour. After that, you have to log back.

b. Messaging System

- One-to-one messaging.
- Messages, group, and user list refresh every 5 seconds.
- Group messaging with add/remove members, the group controlled by Admin.
- Attachments are supported in both personal and group chats.

c. User Profile & Public Info

- Search users by username or email.
- Users can make changes to their profile.
- Advanced features like groups, file-sharing and relations will be accessible for all the verified users.

d. Admin Moderation

- Admin can view verified/unverified users and approve/suspend/delete the user's account.
- The Admin dashboard includes toggles for verification of the user.

e. Media Uploads & Attachments

- A file can be uploaded to chat (images, docs).
- All uploads are handled over HTTPS.

f. Security & Logging

- Passwords hashed using Bcrypt, Spring Security + JWT for backend authentication, Angular + Spring validators for input validation, and logging of critical user/admin actions.

5. Security Mechanisms

- **HTTPS (SSL/TLS):** Encrypted data transmission
- **JWT Authentication:** Secure user sessions
- **Input Validation:** XSS, SQL/NoSQL injection prevention
- **PKI Usage:** Implements RSA-based Public Key for secure key management and digital signatures.
- **Audit Logging:** Records activities like registration, login, and moderation for security monitoring.
- **Admin Moderation:** Active user verification/suspension

6. Interface Design

To simplify things, we created many TypeScript files under interfaces. These are so helpful for easily connecting the front end (Angular) with the back end (Spring Boot).

- Each interface is made for a clear purpose.
- They help in sending and receiving data easily.

- They make the code simple, clean, and easy to use.

a. User and Profile

- Interfaces for user details, login, profile picture.
- Used for signup, search, and document upload.
- Interfaces: `user.ts`, `user-response.ts`, `user-search.ts`, `user-relations.ts`, `profile-picture-request.ts`, and El. at.

b. Chat and Groups

- Interfaces for messages and group chat.
- Used for sending messages, adding members, and group data.
- Interfaces: `message-request.ts`, `message-response.ts`, `message-page-request.ts`.

c. Login and OTP

- Interfaces for login and OTP code.
- Used to check users and keep the app secure.
- Interfaces: `login-request.ts`, `login-response.ts`, `otp.ts`

d. Pages and Lists

- Interfaces to send data in pages.
- Helps show users, messages, or groups in parts.
- Interfaces: `page-request.ts`, `page-response.ts`

f. Stats

- Interface: `stats.ts`
- Interface to show total users, reviews, and products.

7. Structure and Flow

- Our App is created using a modular structure. It includes a frontend (Angular), a backend (Spring Boot), and a database (MySQL), with Nginx acting as the frontend server and reverse proxy.

Components Involved

- **User Browser:** Where users open the browser and interact with the web app.
- **Nginx:** Serves the Angular frontend and forwards API requests.
- **Angular App:** Provides the user interface and makes API calls.
- **Spring Boot (Tomcat):** Backend server that handles requests and logics.
- **MySQL Database:** Stores user data, messages, groups, etc.

How It Works

1. User opens the web app that is `https://VMhost`.
2. Nginx serves the Angular frontend files.
3. User interacts with the UI (e.g., login, send a message, create a group).
4. Angular sends API requests to the backend (e.g., `POST /api/login`).
5. Nginx forwards these requests to Spring Boot running on Tomcat.
6. Spring Boot will verify the request, process the data, and connect with the MySQL database.
7. The backend will send a response (in JSON format) back to Angular.
8. Then Angular will receive the data from the backend and then update the UI accordingly.

Example: Sending a Chat Message

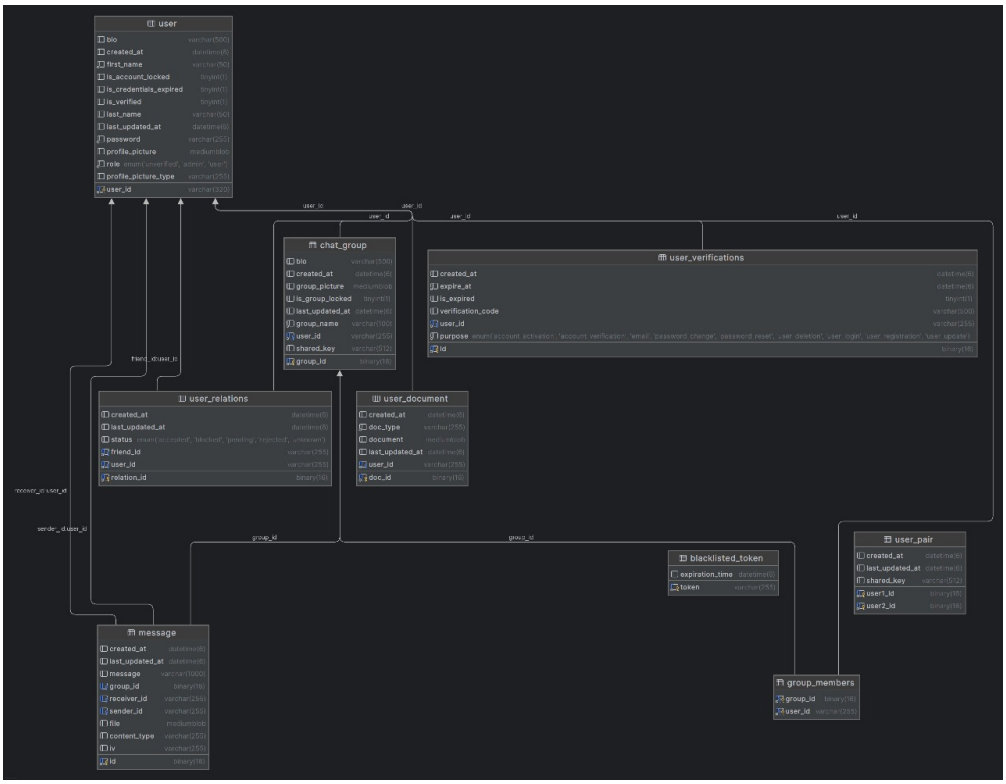
- User sends a message from the chat window.
- Angular sends the message to the backend via API.
- Backend saves the message in MySQL and returns success.
- Angular shows the sent message in the chat window.

Communication Flow

`[Browser]` $\leftrightarrow$ `[Nginx]` $\leftrightarrow$ `[Spring Boot (Tomcat)]` $\leftrightarrow$ `[MySQL]`

8. Screenshots for verifications

8.1 DataBase Schema



9. HTTPS Access Note

Our app uses a self-signed SSL certificate. Most probably, most of the browsers will show a warning that the website is not safe. So, click "Advanced" and then "Proceed" to continue. This is safe.