Ain Shams University
Faculty of Engineering
Computer and Artificial Intelligence Engineering

CSE343: Web Development

Fall 2025

FINAL PROJECT REPORT

Reddit Clone Application

A Full-Stack Application using Spring Boot

Built with:
React.js • Spring Boot • MSSQL Server • Tailwind CSS

December 20, 2025

Team Contributions

The following table outlines the student IDs and specific project responsibilities for each team member:

Member Name	Student ID	Contribution Detail
Abdelrhman Mohammed	**23P0370**	**Data Architecture:** Defined all JPA Entities and relational mappings for MSSQL. Created the DTO layer for API contracts and implemented the Repository interfaces for data persistence.
Nagy Ahmed	**23P0365**	**Backend Logic:** Developed the REST Controllers and Service layers. Integrated Google Gemini AI for summarization, implemented multipart image upload, and configured Spring CORS.
Ahmed Mohamed Al Amin	**22P0137**	Designed the global layout and navigation, and Built authentication pages.
Kareem Younis	**22P0136**	Implemented home feed data fetching, dynamic community routing, and search functionality.
Abdallah Omar	**22P0129**	Post submission forms with image support, and user profile management modules.

Contents

1 Introduction

In the modern digital era, social news aggregation platforms have revolutionized the way information is shared and discussed globally. This project, titled **"Reddit clone,"** is a comprehensive full-stack application designed to replicate the dynamic social environment of Reddit. By combining robust backend logic with a modern, responsive frontend and cutting-edge AI integration, this project demonstrates a scalable approach to building community-driven social media platforms.

2 Project Overview

2.1 Project Description

The Heart of the Internet is a full-stack Reddit clone that replicates the core functionality of Reddit's social news aggregation and discussion platform. The application allows users to create accounts, join communities, publish posts containing text or images, participate in comment threads, and interact through upvote and downvote mechanisms.

Additionally, the system integrates AI-powered post summarization using **Google's Gemini API** to enhance content accessibility and user experience. This project is developed as an educational and practical demonstration of modern full-stack web application design.

2.2 Project Goals and Objectives

The primary objectives of this project are:

- **Community Building:** Enable users to create, join, and manage topic-based communities.
- **Content Sharing:** Support rich content creation, including text, images, and links.
- **Social Interaction:** Provide threaded comments, voting mechanisms, and user engagement features.
- **AI Integration:** Utilize artificial intelligence to generate concise summaries of post content.
- **User Management:** Implement secure authentication, authorization, and profile customization.
- **Modern UX:** Deliver a responsive, Reddit-inspired interface with intuitive navigation.

2.3 Technologies Used

Frontend

- **Next.js 15 (App Router):** React-based framework with server-side rendering.
- **React 19:** User interface development.
- **Tailwind CSS 4:** Utility-first styling framework.

- **Lucide React:** Icon library.

Backend
- **Spring Boot:** Backend application framework.
- **Spring Security:** Authentication and authorization.
- **Java:** Core backend programming language.
- **Maven:** Dependency management and build automation.
- **RESTful APIs:** Communication layer.

Database & AI
- **Relational Database:** Microsoft SQL Server for persistent storage.
- **Google Gemini API:** AI-powered post content summarization.

Infrastructure
- **CORS Configuration:** Secure frontend-backend communication.
- **Multipart File Upload:** Image handling for posts and profiles.
- **HTTP/JSON:** Standard client-server communication protocol.

2.4 Features and Implemented Requirements

The Reddit Clone application implements a comprehensive set of functional requirements to replicate the core behavior of a modern social discussion platform. The following features have been successfully developed and integrated:

- ✔ **User Management:** A full lifecycle for users, allowing them to register new accounts, securely authenticate (login), manage profile details, and exercise the ”right to be forgotten” through account deletion.
- ✔ **Community Management:** Enables topic-based social organization. Users can create new communities, join existing ones to personalize their experience, and view detailed community metadata.
- ✔ **Post Management:** Provides rich content creation capabilities. Authenticated users can publish posts containing text and optional images (handled via multipart storage), browse posts via global or community-specific feeds, and manage their own content.
- ✔ **Commenting System:** Implements a threaded discussion model, allowing users to engage in conversations, provide feedback on posts, and view organized comment chains.
- ✔ **Voting Mechanism:** Promotes community-driven content ranking. Users can interact with comments using upvote and downvote functionality, directly affecting the visibility and popularity of the content.
- ✔ **AI-Powered Summarization:** Utilizes the **Google Gemini API** as a microservice to generate concise, automated summaries of long post content, improving accessibility and reducing information overload.
- ✔ **Search and Discovery:** Facilitates platform exploration through keyword-based search functionality and discovery routes (Explore, Popular), allowing users to find new communities and relevant content easily.

3 Architecture Overview

The ”Heart of the Internet” application follows a standard **three-tier architecture**, ensuring a clean separation of concerns between the user interface, the business logic, and the data persistence layers. This modularity allows for independent scaling and maintenance of each component.

3.1 Frontend (Presentation Layer)

The frontend is a modern web application built with **Next.js 16 (App Router)** and **React 19**, styled using **Tailwind CSS**.

- **Responsibility:** Rendering the user interface, managing client-side state, and handling user interactions.
- **Communication:** It interacts with the backend by sending asynchronous HTTP requests to RESTful API endpoints.
- **User Experience:** Utilizing Next.js Server Components and client-side hydration to deliver a fast, responsive, Reddit-inspired experience.

3.2 Backend (Application Layer)

The backend logic is implemented using the **Spring Boot** framework, providing a robust environment for managing the application's core functionality.

- **Architecture:** It follows a Controller-Service-Repository pattern. Controllers handle incoming REST requests, Services process the business logic, and Repositories manage data flow.
- **AI Integration:** A dedicated service layer manages communication with the **Google Gemini API** to generate real-time post summaries.
- **Security:** Integrated with **Spring Security** to handle authentication and secure API access.

3.3 Data Layer (Persistence Layer)

For persistent storage, the system utilizes **Microsoft SQL Server (MSSQL Server)**.

- **Structure:** A relational schema is used to define complex relationships between users, subreddits, posts, comments, and votes.
- **Abstraction: Spring Data JPA (Hibernate)** is used to map Java entities to MSSQL tables, providing an abstraction layer that simplifies database queries and ensures data integrity.

3.4 Security and Integration

- **CORS:** Cross-Origin Resource Sharing is configured to allow secure communication between the Next.js frontend and the Spring Boot backend.
- **External APIs:** The system integrates with Google's Gemini AI to provide enhanced content accessibility through automated summarization.

3.5 Runtime Model

The application operates as two separate processes:

1. **Frontend Process:** Runs the Next.js server, handling SSR (Server-Side Rendering) and static asset delivery.
2. **Backend Process:** Runs the Spring Boot JVM application, managing business logic and the MSSQL database connection.

This separation ensures that the system remains maintainable, modular, and ready for deployment in a cloud environment.

4 Frontend Documentation

The frontend is a modern web application designed to provide a highly responsive, Reddit-inspired user experience. Built with **Next.js 16 (App Router)** and **React 19**, it leverages server-side rendering and client-side hydration to ensure fast load times and SEO-friendly content delivery.

4.1 Technology Stack

- **Framework:** Next.js 16 (App Router)
- **Library:** React 19
- **Styling:** Tailwind CSS v4 (Integrated via PostCSS)
- **Icons:** Lucide React
- **Linting:** ESLint 9 (eslint-config-next)

4.2 Project Structure

The project follows a modular directory structure to separate concerns and improve maintainability:

- `src/app/`: Contains all routes, layouts, and global styles.
- `src/components/`: Reusable UI components (Navbar, Sidebar, PostCard).
- `src/lib/api.js`: Centralized API layer for all backend communication.
- `public/`: Static assets like logos and default images.

4.3 Routing System

The application utilizes file-based routing via the Next.js App Router. Below is the mapping of URLs to their respective source files:

Route	Source File Path
Home Feed (/)	`src/app/page.js`
Community	`src/app/community/[name]/page.js`
Post View	`src/app/post/[id]/page.js`
User Profile	`src/app/user/[email]/page.js`
Auth Pages	`src/app/(auth)/login` or `/register`

4.4 Data Integration and API Layer

All interactions with the Spring Boot backend are abstracted into `src/lib/api.js`. This layer handles asynchronous requests and ensures consistent data flow.

- **User Service:** Functions for *loginUser*, *registerUser*, and *updateProfile*.
- **Content Service:** Functions for *getAllPosts*, *createPost*, and *addComment*.
- **Social Service:** Functions for *upvoteComment*, *downvoteComment*, and *joinCommunity*.
- **AI Service:** Integrations for *summarizePost* using the Gemini API.

4.5 Image Handling and UI/UX

- **Binary Image Conversion:** Since the MSSQL database returns images as byte arrays, the frontend uses a utility function `getImageDataUrl()` to convert raw data into displayable Base64 URLs on the fly.
- **Global Styles:** `globals.css` defines the color palette, dark mode tokens, and custom Reddit-style animations.
- **Component Library:**
  - **Navbar.js:** Handles search, auth status, and navigation.
  - **Sidebar.js:** Displays communities and resource links.
  - **PostCard.js:** A complex component responsible for rendering post metadata, images, and engagement buttons.

5 Backend API Documentation

The backend layer is a RESTful web service developed using **Spring Boot**. It facilitates communication between the Next.js frontend and the MSSQL database. The API handles data exchange using `JSON` for standard requests and `multipart/form-data` for file and image uploads.

5.1 User Management (UserController)

This controller handles the lifecycle of user accounts, including registration, authentication, and profile updates.

Method	Endpoint	Description
POST	/create-account	Registers a new user with an optional profile image.
GET	/test_login	Authenticates user via *userEmail* and *password*.
POST	/update-profile	Updates profile info and/or user image.
POST	/delete-account	Removes user account based on email.

5.2 Post Management (PostController)

Handles the creation, retrieval, and organization of content across different communities.

Method	Endpoint	Description
POST	/create-post	Creates a new post with text and optional images.
GET	/get-post	Retrieves specific post details by ID.
GET	/get-all-posts	Fetches a global feed of all available posts.
GET	/get-community-posts	Filters posts by community name.
POST	/delete-post	Deletes a post from the system.

5.3 Commenting and Voting (CommentController)

Manages threaded discussions and user engagement through voting mechanisms.

Method	Endpoint	Description
POST	/add-comment	Appends a comment to a specific post.
GET	/get-post-comments	Retrieves the comment thread for a post.
POST	/up-vote	Increments the score of a comment.
POST	/down-vote	Decrements the score of a comment.

5.4 Community Management (CommunityController)

Handles the creation of sub-communities and user memberships.

Method	Endpoint	Description
POST	/create-community	Initializes a new topic-based community.
POST	/join-community	Subscribes a user to a community.
GET	/get-communities	Lists all available communities.
POST	/delete-community	Deletes a community (restricted to owner).

5.5 AI Summarization (AiController)

Leverages the **Google Gemini API** to provide automated content insights.

Method	Endpoint	Description
POST	/summarize	Generates an AI summary for a single post.
POST	/summarize-all-posts	Bulk summarizes multiple post contents.

5.6 Data Transfer Objects (DTOs)

To ensure strict data validation and decouple the internal database entities from the API layer, the following DTOs are implemented:

- **User DTOs:** `UserReqDto`, `UserUpdateDto`, `UserResDto`
- **Post DTOs:** `PostReqDto`, `PostResDto`
- **Comment DTOs:** `CommentReqDto`, `CommentResDto`
- **Community DTOs:** `CommunityReqDto`, `CommunityResDto`

These objects ensure that only necessary data is exposed to the frontend, improving security and reducing payload size.

5.7 Backend Implementation Screenshots

This section provides visual evidence of the Spring Boot implementation, showcasing the Controller logic for user management, communities, comments, and the structured DTO layer.

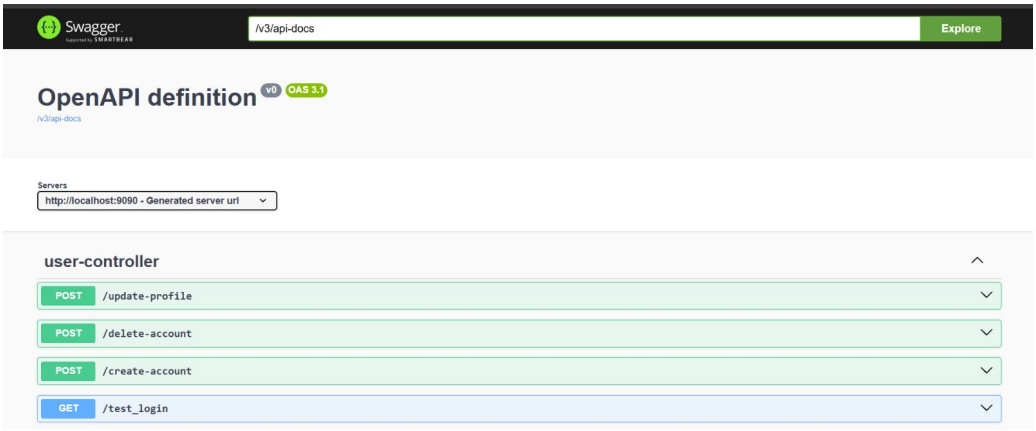

Figure 1: UserController Implementation in Spring Boot

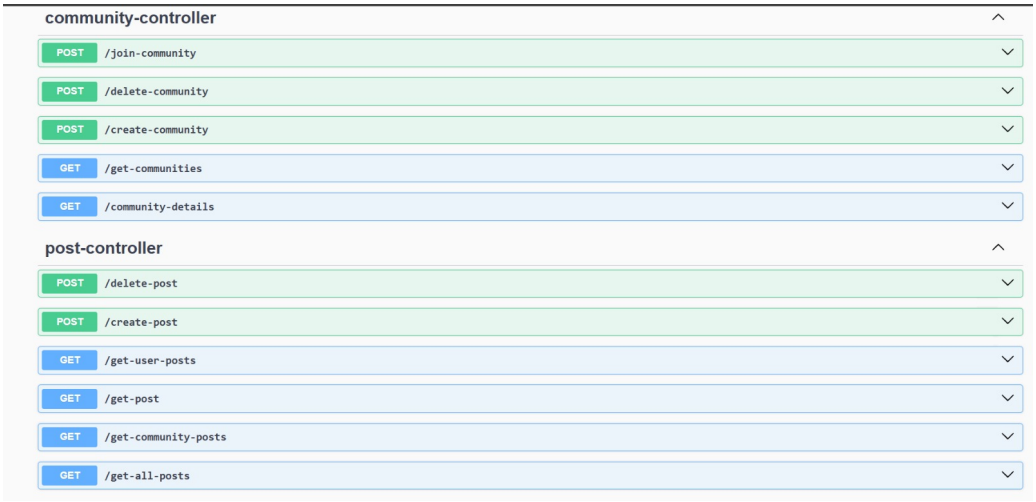

Figure 2: CommunityController Implementation and Routing

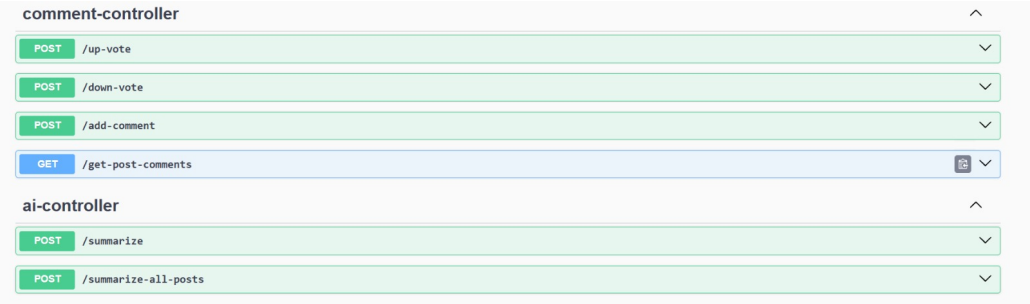

Figure 3: CommentController and Voting Logic Implementation

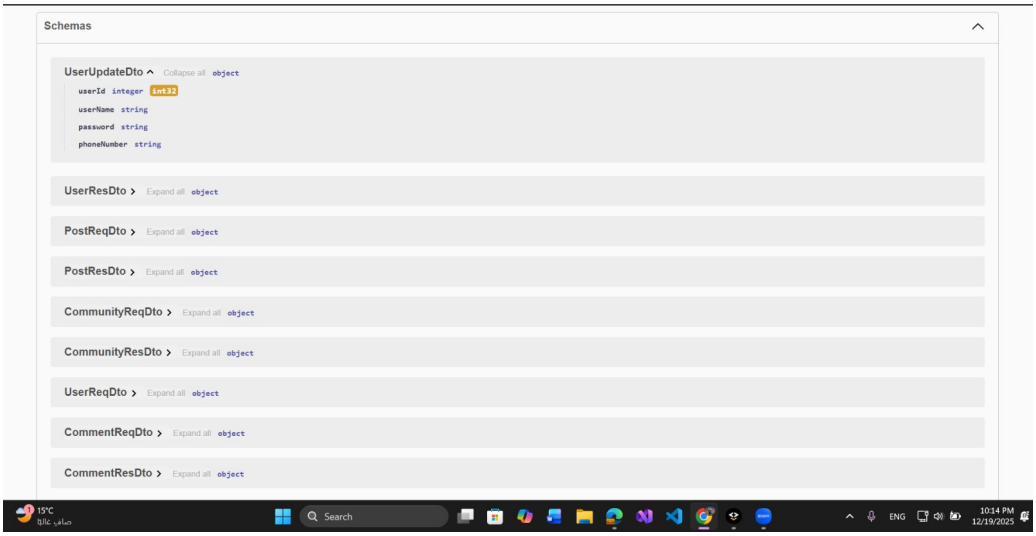

Figure 4: Data Transfer Object (DTO) Class Structures

6 Database Design

The persistence layer of ”The Heart of the Internet” is powered by **Microsoft SQL Server (MSSQL)**. The schema is optimized for relational integrity while supporting social media features like complex memberships and rich media storage. The complete schema definition is maintained in the project’s `RedditCloneSchema.sql` file.

6.1 Core Tables

The following table describes the primary entities within the system and their respective roles:

Table Name	Description and Key Features
users	Stores account data (email, phone, password), timestamps, and profile images (`VARBINARY(MAX)`). Supports soft deletion via *delete_at*.
communities	Defines topic-based groups with unique names and descriptions. Linked to an owner via *user_id*.
posts	Contains the core content (title, text, optional images) and maintains foreign keys to both the author and the community.
comments	Stores threaded discussion data. Includes an *is_edited* flag and a *votes* counter for social ranking.
community_members	A join table enabling a many-to-many relationship between users and communities using a composite primary key.

6.2 Entity Relationships

The database enforces the following logical connections to maintain data consistency:

- **One-to-Many (1:N):** A single user can author multiple posts and comments.
- **One-to-Many (1:N):** Each community hosts many posts, and each post can contain many comments.
- **Many-to-Many (M:N):** Users can join multiple communities, and communities consist of many users, managed via the membership join table.

6.3 Constraints and Data Integrity

To ensure the reliability of the system, several SQL-level constraints are applied:

- **Referential Integrity:** Foreign keys are enforced with `ON DELETE NO ACTION`. Referential handling is managed at the Spring Boot service layer to prevent accidental data loss.
- **Uniqueness:** Strict `UNIQUE` constraints are placed on *users.email*, *users.phone_number*, and *communities.community_name*.
- **Soft Deletion:** Instead of permanent removal, *delete_at* timestamps are used for users and communities, allowing for data recovery and audit trails.

6.4 Storage and Performance Optimization

- **Image Storage:** Unlike traditional file-path storage, images for posts and profiles are stored directly in the database as **VARBINARY(MAX)** blobs. This ensures that the database remains the single source of truth for all application data.
- **Indexing:** Performance is optimized by indexing lookup columns (*email, community_name*), join columns (*user_id, post_id*), and frequently filtered fields (*create_at, delete_at*).
- **Voting Logic:** For performance efficiency, comment voting is implemented as an atomic integer counter (*votes*) within the comments table, reducing the need for expensive join operations during feed retrieval.

6.5 Entity Relationship Diagram (ERD)

The following diagram illustrates the relational structure of the system, including the primary keys, foreign keys, and cardinality between the Users, Communities, Posts, and Comments entities.

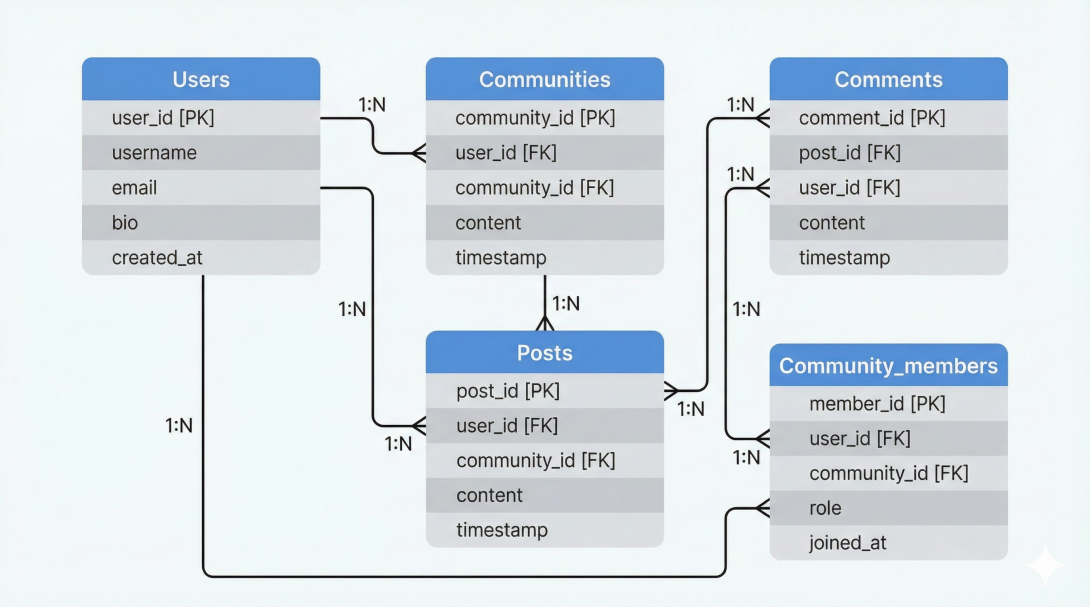

Figure 5: Entity Relationship Diagram of the Reddit Clone

7 User Interface Design

This section showcases the responsive frontend of ”The Heart of the Internet,” built using Next.js and Tailwind CSS. The screenshots demonstrate the authentication flow, content discovery, and user interaction modules.

7.1 Authentication and Profile

The application provides a secure entry point for users and personalized profile management.

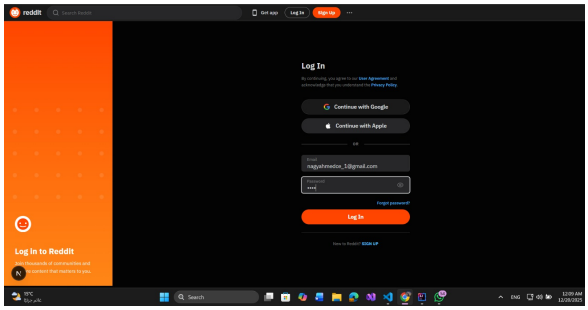

Figure 6: User Login Interface

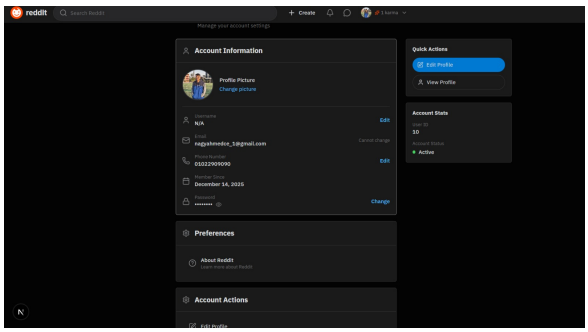

Figure 7: User Profile Management

7.2 Content Feed and Communities

Users can browse a global feed of posts or explore specific community-driven subreddits.

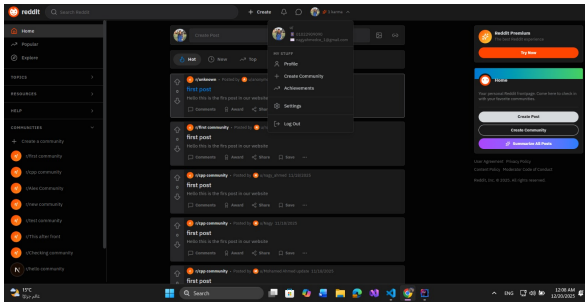

Figure 8: Home Feed and Post Listing

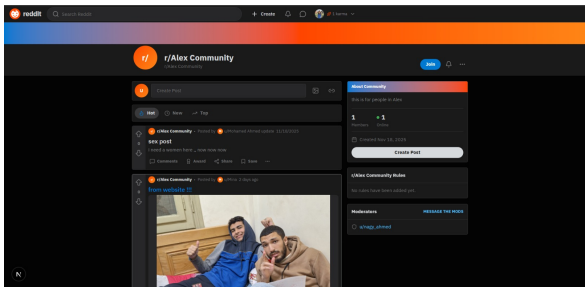

Figure 9: Community Discovery Page

7.3 Post Interaction and Creation

The platform supports rich text and image submissions with threaded commenting systems.

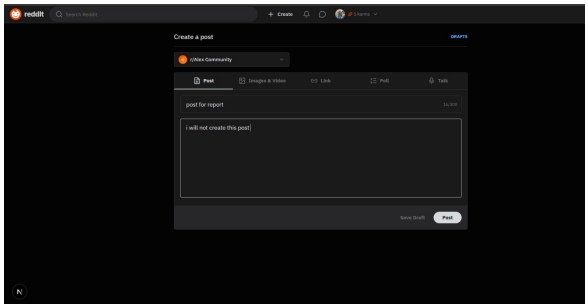

Figure 10: Post Creation Interface

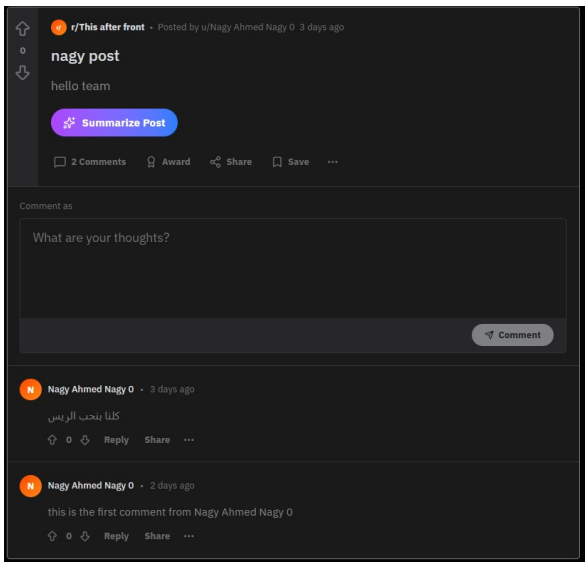

Figure 11: Threaded Commenting and Voting

8 Source Code Repository

To facilitate review and further development, the complete source code for ”The Heart of the Internet” is hosted on GitHub. This repository contains the full stack implementation, including the Next.js frontend, the Spring Boot backend, and the SQL initialization scripts.

GitHub Repository Link

https://github.com/Nagy-Ahmed-cs/The-heart-of-the-Internet.git

8.1 Repository Structure

The repository is organized into distinct modules for the frontend and backend to maintain a clean separation of concerns:

- **Frontend Application:** Located in the `/frontend` directory. This contains the Next.js 16 project, Tailwind CSS configurations, and React 19 components.
- **Backend Application:** Located in the `/reddit_clone` directory. This contains the Spring Boot Java source code, Maven configuration (`pom.xml`), and service logic.
- **Database Schema:** The SQL script used to initialize the MSSQL database and define the relational structure is located at:
  `/reddit_clone/src/main/resources/RedditCloneSchema.sql`

End of Documentation