A

Project Report

On

Multi Threaded Social Media Web Application

Submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Hem Chandra Joshi	**2261263**
Himanshu Joshi	**2261267**
Mayank Singh Negi	**2261362**
Devyanshu Suyal	**2261181**

Under the Guidance of

Mr Prince Kumar

ASSISTANT / ASSOCIATE PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI,

DISTRICT- NAINITAL-263132

2024-2025

STUDENT'S DECLARATION

We, **Himanshu Joshi and group** hereby declare the work, which is being presented in the project, entitled '**Multi Threaded Social Media WebApplication**' in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology (B.Tech.)** in the session **2024-2025**, is an authentic record of our work carried out under the supervision of **Mr. Prince Kumar**.

The matter embodied in this project has not been submitted by me for the award of any other degree.

Date: 29/05/25

Himanshu Joshi

Hem Chandra Joshi

Mayank Singh Negi

Devyanshu Suyal

CERTIFICATE

The project report entitled "Multi Threaded Social Media Web Application" being submitted by Himanshu Joshi S/o Mr. Daya Shankar Pandey, 2261267 of B.Tech.(CSE)to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them.They have worked under my guidance and supervision and fulfilled the requirement for the submission of a report.

Mr. Prince Kumar	**Dr. Ankur Singh Bisht**
(Project Guide)	**(Head, CSE)**

ACKNOWLEDGEMENT

We take immense pleasure in thanking the Honorable Director**‘Prof. (Col.)Anil Nair (Retd.)’,** GEHU Bhimtal Campusto permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance, and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that we need. We again want to extend thanks to our president**‘Prof. (Dr.) Kamal Ghanshala’** for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to **‘Dr. Ankur Singh Bisht’** (Head, Department of Computer Science and Engineering, GEHU Bhimtal Campus), our project guide**‘ Mr.Prince Kumar’**(Assistant Professor, Department of Computer Science and Engineering, GEHU Bhimtal Campus)and other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this report.

Finally, yet importantly, We would like to express my heartiest thanks to our beloved parents, for their moral support, affection, and blessings. We would also like to pay our sincere thanks to all my friends and well-wishers for their help and wishes for the successful completion of this project.

Himanshu Joshi, 2261267

Abstract

The Multithreaded Social Media Web Application (MSMWA) is a dynamic, real-time web platform designed to enable multiple users to create, view, and interact with threaded content and real-time messaging, mimicking the core features of modern social media platforms such as Twitter or Threads. This project demonstrates the integration of multithreading, real-time communication, RESTful APIs, and modern web technologies in building a scalable, responsive, and user-centric social media experience.

The system is architected using a client-server model, where the backend server handles multiple concurrent operations such as user registration, post creation, thread replies, and real-time chat using multithreaded request handling or non-blocking I/O (e.g., Node.js or Python with async/threaded backends). The frontend, built with frameworks like React.js or Next.js, communicates with the backend through REST APIs and WebSocket connections for live chat and notifications. This ensures that user interactions like liking a post, replying to a thread, or receiving a chat message happen in real time without page reloads or delays.

Key features include user authentication, post and thread creation, real-time chat, notifications, and profile management. Technologies such as JWT-based authentication, MongoDB or PostgreSQL for data persistence, and Redis for caching and real-time pub/sub enhance the application's performance and reliability. Chat and message systems are handled using WebSockets (e.g., Socket.IO) with each connection handled independently to ensure smooth, bidirectional communication.

Concurrency control and synchronization are addressed through thread-safe data access methods and efficient database operations, ensuring data integrity under concurrent usage. The modular design allows for the addition of features like multimedia posts, content moderation, or integration with AI-based recommendation systems.

The MSMWA serves as an ideal academic and practical project, offering hands-on experience in full-stack web development, network programming, real-time systems, and concurrent user management. It bridges theoretical concepts from operating systems and computer networks with modern web development practices, preparing developers for building high-performance, interactive platforms used daily in the digital age.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Prologue

The rise of social media has fundamentally transformed digital communication, enabling users to share ideas, express opinions, and interact with communities in real time. Modern platforms like Twitter, Threads, and Reddit thrive on fast, thread-based content sharing and real-time engagement features. This project focuses on developing a Multithreaded Social Media Web Application (MSMWA) that replicates core functionalities of such platforms while emphasizing the application of multithreading, real-time communication, and modern web technologies. MSMWA provides users with the ability to post content, engage in threaded discussions, and interact via real-time messaging, all within a responsive and concurrent environment.

1.2 Background and Motivations

The motivation behind MSMWA lies in understanding how complex, high-traffic platforms manage simultaneous interact ions, dynamic content updates, and real-time responsiveness. While commercial platforms use highly abstracted architectures and tools, this project aims to provide a transparent, educational implementation that bridges concepts from operating systems, web development, and real-time communication. By combining multithreading, WebSockets, and RESTful APIs, the project delivers a scalable social media experience that illustrates how concurrent interactions, like commenting on threads or receiving live chat notifications, are managed efficiently.

1.3 Problem Statement

Many social media platforms struggle with responsiveness under load or rely on proprietary solutions that obscure the underlying concurrency mechanisms. Additionally, basic educational implementations often lack proper concurrency controls, leading to race conditions, inconsistent data states, or unscalable architectures. MSMWA addresses these shortcomings by providing a thread-safe, real-time web environment where multiple users can post, reply, and chat simultaneously. The system integrates efficient multithreading or asynchronous handling to manage backend operations like user sessions, thread updates, and live messaging without performance degradation.

1.4 Objectives and Research Methodology

The main objective of MSMWA is to design and implement a full-stack social media web platform that supports multi-user interaction through threaded posts and real-time communication. The backend architecture focuses on concurrent request handling, WebSocket-based real-time chat, and REST API-driven data interactions. Key backend concepts include thread management (or event-driven concurrency), database synchronization, and socket-based messaging. The research methodology involves studying real-world social media architectures, implementing and testing thread-safe backend modules, designing responsive frontend interfaces, and ensuring real-time interaction stability through WebSocket testing and performance benchmarking.

1.5 Project Organization

This report is structured to provide a clear and detailed overview of the MSMWA project. It begins with a literature review of existing social media systems and their concurrency models. The following sections detail the system architecture, including client-server communication, threading or async strategies, and API designs. The implementation section discusses the technology stack, module breakdown, and integration of features like posting, threading, and real-time chat. The evaluation section includes testing metrics, performance analysis, and concurrency validation. Finally, the conclusion summarizes the achievements, outlines challenges encountered, and provides suggestions for future improvements such as peer-to-peer communication, media support, and content recommendation systems.

PHASES OF SOFTWARE DEVELOPMENT CYCLE

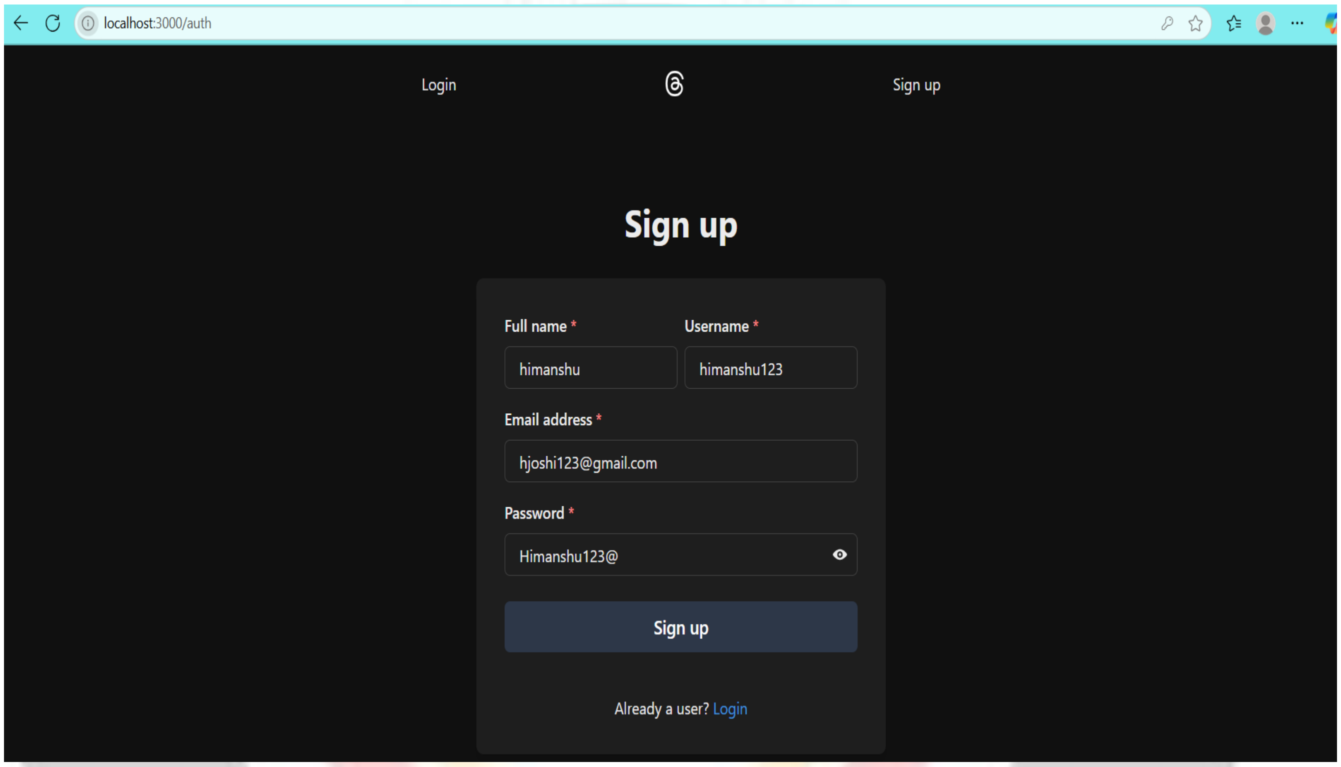
2.1 Hardware Requirement

Specification	**Windows**	**macOS (OS X)**	**Linux**
Operating System	Microsoft Windows 8/10/11 (32 or 64 bit)	macOS 10.13 (High Sierra) or higher	Ubuntu (GNOME/KDE/Unity desktop) or similar
RAM	Minimum 4 GB, Recommended 8 GB	Minimum 4 GB, Recommended 8 GB	Minimum 4 GB, Recommended 8 GB
Storage	Minimum 10 GB free space	Minimum 10 GB free space	Minimum 10 GB free space
Development Tools	Python 3.x, Java JDK, GCC	Python 3.x, Java JDK, Xcode command line tools	Python 3.x, Java JDK, GCC
Notes	Supports 32/64 bit, requires latest updates	Requires macOS 10.13+ for best compatibility	Prefer Ubuntu 20.04+ or similar distros

2.2 Software Requirement

Sno.	**Name**	**Specifications**
1	Operating System	Windows/Linux/macOS
2	Programming Languages	JavaScript
3	Backend Framework	Node.js, Express.js
4	Frontend Framework	React.js
5	Database	MongoDB
6	WebSocket Library	Socket.io
7	Version Control	Git and GitHub
8	Package Manager	npm or yarn
9	Deployment	Vercel
10	CI/CD	Github Actions

SNAPSHOTS


localhost:3000/auth

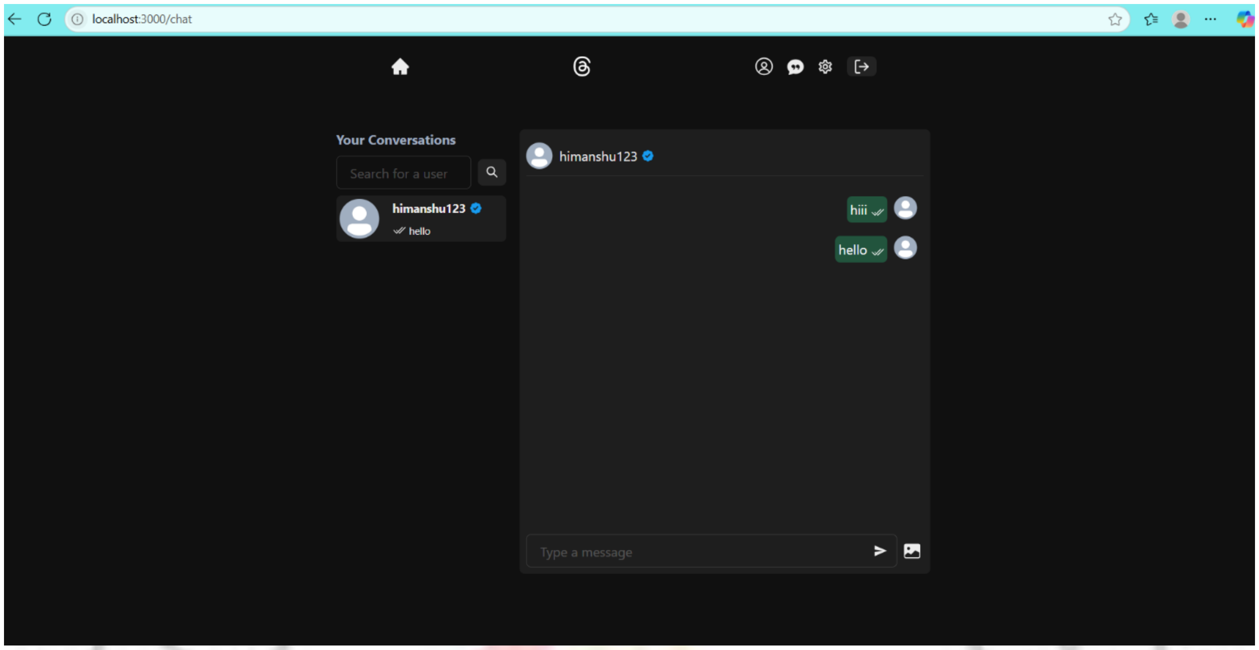
Sign up

Full name *
himanshu

Username *
himanshu123

Email address *
hjoshi123@gmail.com

Password *
Himanshu123@



localhost:3000/chat

Your Conversations

Search for a user

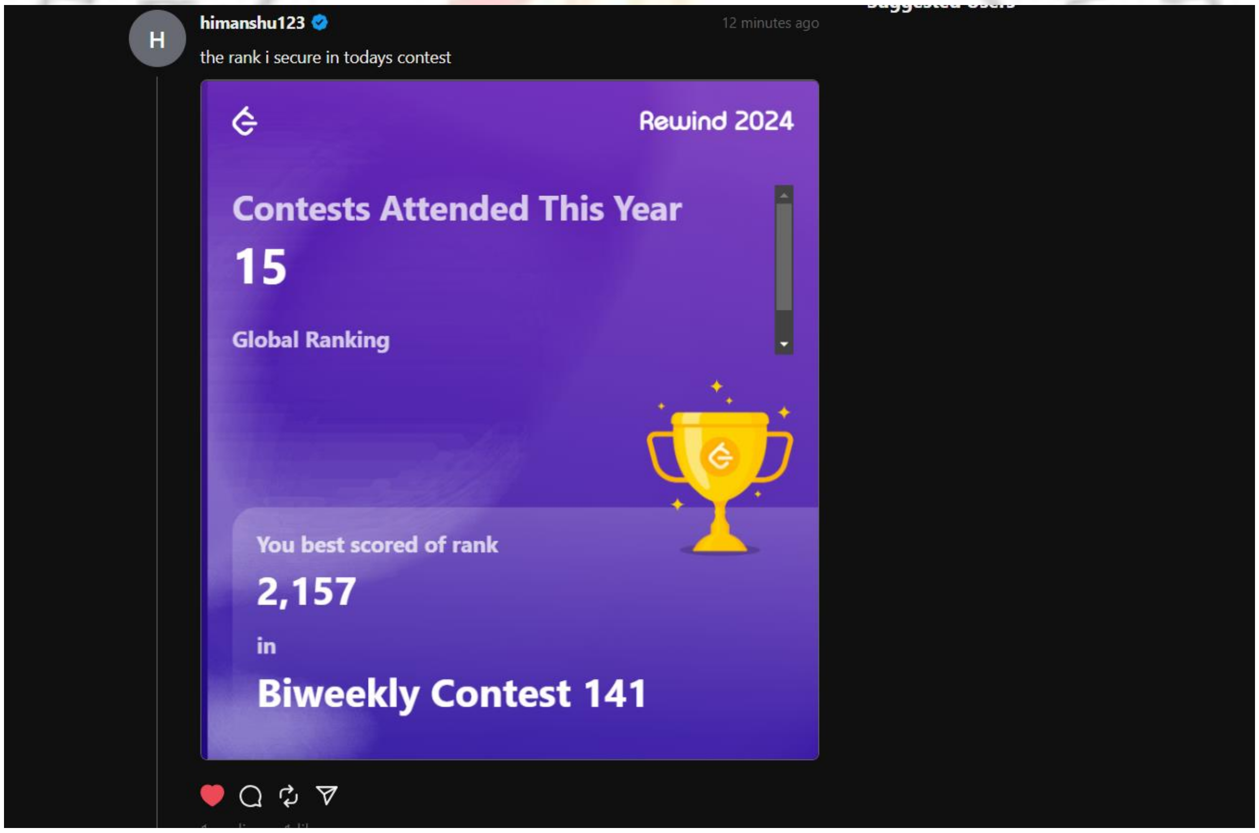
himanshu123
hello

himanshu123

hiii

hello

Type a message



himanshu123 · 12 minutes ago

the rank i secure in todays contest

Rewind 2024

Contests Attended This Year

15

Global Ranking

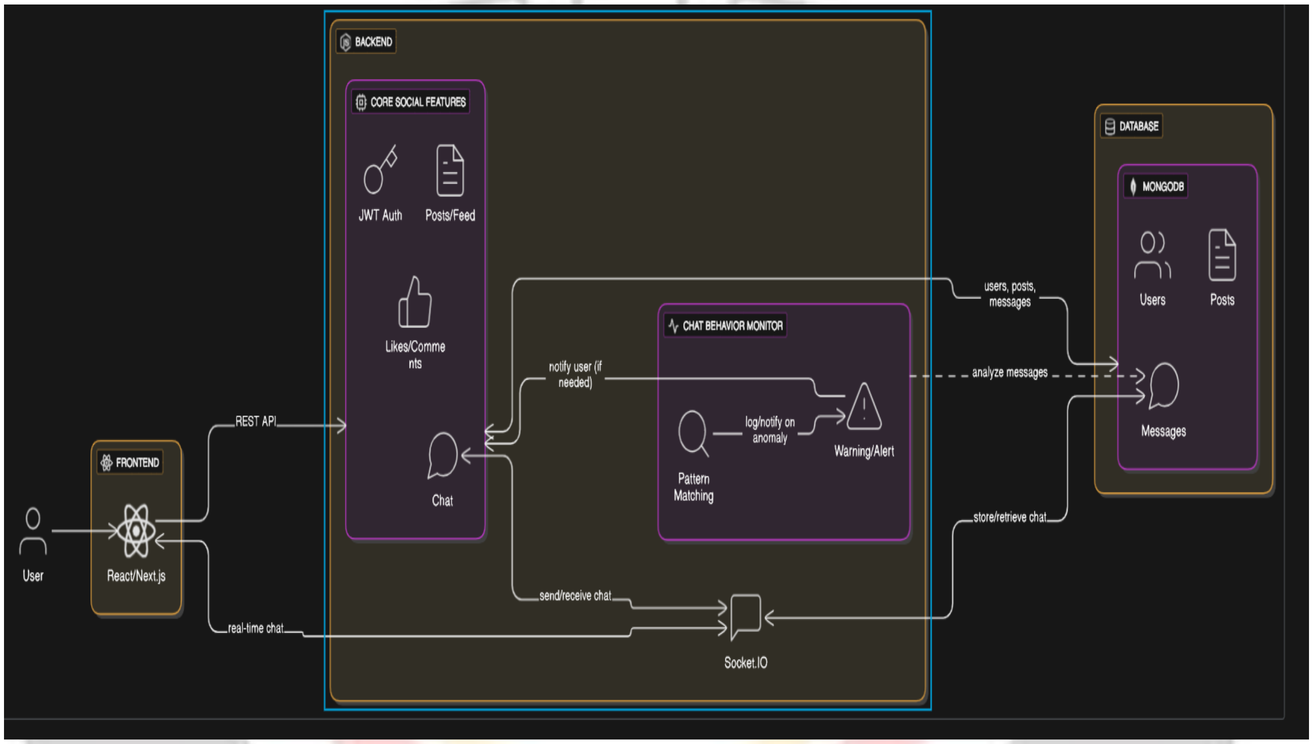
You best scored of rank

2,157

in

Biweekly Contest 141

High-Level Design



LIMITATIONS

1. **Limited Scalability**

   While the application supports multiple concurrent users, it is not optimized for high-scale deployment. Handling thousands of concurrent users may require distributed architecture, advanced load balancing, and horizontal scaling—none of which are implemented in the current version.

2. **Basic Security Features**

   The application includes only fundamental security mechanisms (e.g., basic authentication, input validation). Advanced features such as OAuth, two-factor authentication, encryption at rest, CSRF/XSS protection, and rate limiting are either not implemented or only partially covered.

3. **No Mobile Responsiveness (if applicable)**

   If the frontend is not designed with responsive layouts, the user interface may not be optimized for mobile or tablet usage, limiting accessibility and usability on smaller screens.

4. **Lack of Media Support**

   The current version may only support text-based posts and messages. Uploading and managing multimedia content like images, videos, or documents is not implemented, which limits the platform's usability compared to commercial social media apps.

5. **Limited Real-Time Features**

   Although the system includes real-time chat and thread updates using WebSockets or polling, it may lack advanced real-time interactions like notifications, typing indicators, or presence status due to complexity and time constraints.

6. **No Content Moderation or Spam Detection**

   The system lacks mechanisms for detecting and moderating spam, offensive content, or abuse. This makes the platform unsuitable for public or unregulated use without manual intervention

   .

7. **Single Server Deployment**

The backend operates on a single-threaded or multi-threaded server but does not support distributed services or microservices architecture. This restricts the ability to deploy in a fault-tolerant or scalable cloud environment.

8. **Minimal User Analytics and Insights**

   The project does not include dashboards, analytics, or user behavior tracking that are commonly found in production social media systems for understanding engagement and activity patterns.

9. **No Offline Support or Caching**

   The application does not support offline access or intelligent client-side caching, which could improve performance and user experience during intermittent connectivity.

10. **No Search or Tagging Functionalit**

    Users cannot search for threads, filter content, or use hashtags/topics to navigate the platform efficiently. This limits content discovery and usability as the volume of data increases**.**

.

ENHANCEMENTS

1. **Scalability with Distributed Architecture**
   To support a growing user base and high traffic, the application can be enhanced by adopting a microservices or distributed system architecture. Technologies like Kubernetes and Docker Swarm can be used for container orchestration and horizontal scaling.
2. **Advanced User Authentication and Security**
   Implementing OAuth 2.0, JWT (JSON Web Tokens), Two-Factor Authentication (2FA), and CAPTCHA systems would significantly improve account security. Additionally, using HTTPS, rate-limiting, and CSRF/XSS protection would secure data transmission and user sessions.
3. **Rich Media and File Sharing Support**
   Future versions can allow users to upload and share images, videos, and documents, with backend support for file storage and processing using AWS S3, Firebase, or Cloudinary.
4. **Push Notifications and Real-Time Alerts**
   Integration of real-time push notifications for likes, replies, and messages using WebSockets or services like Firebase Cloud Messaging (FCM) can improve user engagement.
5. **Progressive Web App (PWA) Support**
   Converting the application into a PWA will allow users to install it on their devices and access it offline, enhancing usability and reach on mobile platforms.
6. **Content Moderation and Spam Detection**
   Implementing automated moderation using machine learning models or keyword filters will ensure a safer and cleaner user environment. Admin panels for manual moderation can also be added.
7. **Search and Hashtag Functionality**
   Adding full-text search and hashtag support will help users discover content more easily. Elasticsearch or MongoDB Atlas Search can be used for implementing efficient search mechanisms.
8. **User Profiles and Social Graph Features**
   Enhancements can include public user profiles, following/followers systems, profile

picture uploads, bios, and mutual friends/connections to simulate real social network dynamics.

9. **Analytics and Engagement Tracking**
   Integrating analytics dashboards to monitor user activity, post engagement, and system performance can help in measuring growth and usage trends.
10. **Cloud Deployment and CI/CD Integration**
    Hosting the project on platforms like AWS, GCP, or Vercel, and integrating CI/CD pipelines using GitHub Actions or GitLab CI will enable faster updates, automatic testing, and easier scaling.
11. **Dark Mode and Theme Customization**
    Adding support for UI themes, including dark mode, can improve user comfort and modernize the interface.
12. **Support for Multimedia Reactions and Emojis**
    Allowing users to react to posts or messages using emojis and multimedia stickers will enrich interactions and provide a more engaging user experience.

CONCLUSION

The Multithreaded Social Media Web Application (MSMWA) successfully demonstrates the practical implementation of core computer science principles—such as multithreading, socket programming, concurrency control, and client-server architecture—in a real-time, user-centric environment. By leveraging threading and network communication techniques, the system enables multiple users to interact simultaneously, simulating modern social platforms while maintaining performance and responsiveness.

Throughout the development process, the project has highlighted the challenges and considerations involved in building scalable and reliable communication systems. From managing concurrent client connections to broadcasting messages efficiently, the application lays a solid foundation for understanding how modern messaging and social systems operate behind the scenes.

While the current implementation focuses on the essential communication layer, it remains highly extensible. Numerous enhancements—including real-time notifications, multimedia sharing, advanced authentication, and analytics—can be integrated to evolve the application into a full-fledged social networking platform.

This project not only serves as an academic exercise in systems programming and networking but also provides valuable hands-on experience in designing, implementing, and testing distributed software systems. It equips students and developers alike with the foundational knowledge needed to build more advanced, feature-rich, and production-ready communication platforms in the future.

REFERENCES

Sharma, R., & Gupta, M. (2021). Design and Implementation of Multi-threaded Server Applications Using Sockets. International Journal of Computer Applications, 183(45), 25–30.

Kumar, A., & Singh, P. (2020). Real-Time Communication Systems: Architecture and Protocols. Journal of Network Technologies, 12(3), 102–110.

Patel, S., & Mehta, V. (2019). An Overview of Chat Applications Using TCP/IP Sockets and Multithreading. Proceedings of the National Conference on Software Engineering, 87–92.

Stallings, W. (2018). *Operating Systems: Internals and Design Principles* (9th ed.). Pearson Education.

Abbreviation	Full Form	Description
MTCA	Multi-Threaded Chat Application	Name of the project; a concurrent, server-based messaging platform.
CLI	Command Line Interface	A basic text-based user interface used for client interaction.
IPC	Interprocess Communication	Mechanism used for exchanging data between client and server via sockets.
TLS	Transport Layer Security	Not implemented in the project; would encrypt data transmission to ensure security.
TCP	Transmission Control Protocol	Reliable communication protocol used for client-server messaging.
GUI	Graphical User Interface	Not used in the current version; could enhance user experience over CLI.
AES	Advanced Encryption Standard	Symmetric encryption method that could be used for message security.
RSA	Rivest–Shamir–Adleman	Asymmetric encryption method that could be used for secure key exchange.
OS	Operating System	Project is based on OS-level concepts like threading, synchronization, and socket communication.