AUDISANKARA COLLEGE OF ENGINEERING AND TECHNOLOGY

A Full Semester Internship Report on

SOCIAL LEARN : SOCIAL LEARNING NETWORK FOR STUDENTS USING MERN STACK WEB DEVELOPMENT

Submitted in a partial fulfillment for the award of the degree

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING - ARTIFICIAL INTELLIGENCE

Submitted by

DAMA HARI

(21G25A3103)

Under the esteemed Guidance of

MR. SD. AKHTAR BHASHA, M.Tech.,

Assistant Professor, Dept. of CSE

AUDISANKARA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(ARTIFICIAL INTELLIGENCE)

AUDISANKARA
COLLEGE OF ENGINEERING & TECHNOLOGY
An Autonomous Institute affiliated to JNTUA, Ananthapuram & Accredited by NAAC with 'A+' Grade

(Autonomous)

NH5 Bypass Road, Gudur – 524101, Tirupati (DT.)

Andhra Pradesh

(2020-2024)

AUDISANKARA
COLLEGE OF ENGINEERING & TECHNOLOGY
An Autonomous Institute affiliated to JNTUA, Ananthapuram & Accredited by NAAC with 'A+' Grade

(AUTONOMOUS)
NH5 Bypass Road, Gudur – 524101, Tirupati (DT.)
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING - ARTIFICIAL INTELLIGENCE

AUDISANKARA

CERTIFICATE

This is to certify that the full semester internship project report entitled **"SOCIAL LEARN : SOCIAL LEARNING NETWORK FOR STUDENTS USING MERN STACK WEB DEVELOPMENT"** is the bonafide work done by the student **DAMA HARI (21G25A3103)** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE)**, from Jawaharlal Nehru Technological University Anantapur, Anantapur during the year 2020– 2024.

Internship Guide
Mr. SD. AKHTAR BHASHA, M.Tech.,
Assistant Professor,
Department of CSE,
ASCET , GUDUR.

Head of the Department
Dr.S. V. PADMAVATHI DEVI, Ph.D.,
Head of The Department,
Department of CSE(AI),
ASCET , GUDUR.

Submitted for the viva-voice Examination held on:

Internal Examiner **External Examiner**

AUDISANKARA
COLLEGE OF ENGINEERING & TECHNOLOGY
An Autonomous Institute affiliated to JNTUA, Ananthapuram & Accredited by NAAC with 'A+' Grade

(AUTONOMOUS)
NH5 Bypass Road, Gudur – 524101,
Tirupati (DT.) Andhra Pradesh

AUDISANKARA

DECLARATION

I, **DAMA HARI (21G25A3103)** hereby declare that the Project Work entitled **"SOCIAL LEARN : SOCIAL LEARNING NETWORK FOR STUDENTS USING MERN STACK WEB DEVELOPMENT"** done by us under the esteemed guidance of Assistant Professor **Mr.SD. AKHTAR BHASHA, M.Tech.,** Department of Computer Science and Engineering and is submitted in partial fulfillment of the requirements for the award of the Bachelor Of Degree in **COMPUTER SCIENCE AND ENGINEERING (ARTIFICIAL INTELLIGENCE)**.I have not copied from any other students' work or from any other sources except where due reference or acknowledgment is made explicitly, nor has any part been authored by another person. I, as a candidate, declare that in case of any violation of intellectual property right or copyright will be fully responsible for the same. My supervisor should not be held responsible for full or partial violation of copyright or intellectual property right.

Date:

Place:

DAMA HARI
(21G25A3103)

ACKNOWLEDGMENT

I would like to express our heartful gratitude our honorable chairman of **AUDISANKARA GROUP OF INSTITUTION, Dr. VANKI PENCHALAIAH, M.A, M.L, Ph.D,** who provided all facilities and necessary encouragement during the course of study.

I would like to thank **AUDISANKARA COLLEGE OF ENGINEERING AND TECHNOLOGY** for providing the extraordinary support in the completion of the project by utilizing the laboratories, library and Software's required for our project.

I extend my gratitude and sincere thanks to our beloved Director **Dr. A. MOHAN BABU, Ph.D.,** and principal, **Prof K. DHANUMJAYA, Ph.D.,** for motivating and providing necessary infrastructure and permitting us to completion of the project.

I would like to express the sense of gratitude towards our Dean acadamics **Dr.M. RAJAIAH, Ph.D.,** and Head of the Department **Dr.S. V. PADMAVATHI DEVI, Ph.D.,** our external guide **Mr.SD. AKHTAR BHASHA, M.Tech.,** in Computer Science and Engineering for their valuable inputs, able guidence, encouragement, whole-hearted cooperation and constructive criticism throughtout the duration of our project..

We express my sincere thanks to all the **teaching and non-teaching staff** that guided directly or indirectly helped me to complete the project work successfully.

Last, but not the least, we would like to thank our **team members, my friends and my parents** for supporting me in all the aspects for the completion of this project.

DAMA HARI
(21G25A3103)

COMPANY PROFILE

Blackbucks Professional Studies, a prominent tech company launched in 2015, provides job-oriented postgraduate programs in data science, AI, and full-stack development through universities like JNTU Hyderabad, emphasizing employability with small class sizes, personalized mentorship, and a claimed 100% placement record. Additionally, a partnership with IIDT Tirupati offers internship opportunities to bridge the gap between academic concepts and real-world tech experience.

Founded: 2015 Specialties: ISO Certified, MSME, AICTE, EUAS, and IT Services

Mission and Vision

Mission: Revolutionize the logistics industry through technology, making transportation and freight services more efficient, reliable, and transparent for businesses and truckers alike.

Vision: To be the leading logistics platform not only in India but globally, by continuously innovating and improving the logistics ecosystem, making it environmentally sustainable and economically beneficial for all stakeholders.

IT Service

BlackBuck's core IT services revolve around its logistics platform, offering utilizing algorithms to match loads with the best-suited trucks. Providing real-time GPS tracking for shipments and fleet management solutions.

Education Services

Direct education services are not typically a focus for logistics companies. However, BlackBuck may offer training programs for its employees, truckers on its platform, and possibly workshops or seminars on logistics management.

Internship Opportunities

Tech-driven companies like BlackBuck often offer internship opportunities in areas such as software development, data analysis, business development, and operations. These opportunities can provide hands-on experience in a fast-paced startup environment, focusing on technology, logistics, and supply chain management.

Website : https://www.theblackbucks.com
Industry : Software Development

CERTIFICATES

ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION

(A Statutory Body of the Government of A.P.)

Certificate of Completion

Certificate Id: **BBAPSCHDEIIDT2024PART003997**

This is to certify that **Hari Dama**, bearing Reg. No: **21G25A3103**, from **AUDISANKARA COLLEGE OF ENGINEERING & TECHNOLOGY, GUDUR** of **JNTU Anantapur**, has successfully completed a long-term internship for 240 hours on **Full Stack Development.** This internship was organized by **International Institute of Digital Technologies**, with its **industry partner Blackbuck Engineers**, in association with the **Andhra Pradesh State Council of Higher Education (APSCHE).**

Anuradha Thota
Chief Executive Officer
Blackbuck Engineers Pvt. Ltd.

Dr. Sundar Balakrishna
Director General
International Institute of Digital Technologies

Date: 15/04/2024 **Place**: Tirupati, Andhra Pradesh

ANDHRA PRADESH STATE COUNCIL OF HIGHER EDUCATION

(A Statutory Body of the Government of Andhra Pradesh.)

and

INTERNATIONAL INSTITUTE OF DIGITAL TECHNOLOGIES, TIRUPATI

(Information Technology, Electronics & Communication Department, Government of Andhra Pradesh.)

Certificate Id:
BBAPSCHEIIDT2024PART003997

Certificate of Completion

This is to certify that **Mr/Ms Hari Dama,** with Roll No: **21G25A3103** from **Audisankara College of Engineering & Technology, Gudur** of **JNTU Anantapur**, has successfully completed the Internship on **Full Stack Development** conducted by **International Institute of Digital Technologies,** the **Knowledge Partner, Blackbuck Engineers** and **Andhra Pradesh State Council of Higher Education (APSCHE).** His/her performance in the Internship is **"Good".**

Anuradha Thota
Chief Executive Officer
Blackbuck Engineers Pvt. Ltd.

Dr. Sundar Balakrishna
Director General
International Institute of Digital Technologies

Date: 15/04/2024 **Place**: Tirupati

ABSTRACT

In today's digital age, education delivery is rapidly changing due to technology integration and the emergence of new learning methods. the concept of a social learning network (SLN) is gaining popularity as an effective tool for improving learning outcomes and fostering collaborative learning environments. This abstract provides a comprehensive overview of the design, implementation, and benefits of a student-focused Social Learning Network.

The Social Learning Network for Students is designed to be an interactive online platform within academic settings, promoting peer-to-peer collaboration, interactive participation, and knowledge exchange. By leveraging social networking principles, Social Learning Network for Students incorporates features such as group projects, discussion boards, and instant messaging.

Moreover, by connecting learners with peers who share similar academic interests, Social Learning Network for Students fosters a sense of community and motivation in the learning process. Lastly, also acts as a repository of collective intelligence, enabling students to access a variety of perspectives, insights, and resources contributed by their peers.

In summary, implementing a Social Learning Network for Students represents a groundbreaking approach to education that empowers students, enriches their academic experiences, and promotes lifelong learning through digital platforms. Social Learning Network for Students stands as a pioneer of innovation, promoting collaboration, engagement, and academic success as technology continues to shape the educational landscape.

LIST OF FIGURES

INDEX

CHAPTER 1

INTRODUCTION

1.Introduction:

In today's dynamic technological landscape, the demand for robust and engaging web applications is ever on the rise. Businesses seek to forge deeper connections with their audiences, while users expect seamless digital experiences. In response to these evolving needs, the role of web developers has become increasingly pivotal. It's in this context that the MERN stack shines as a beacon of innovation, offering a comprehensive toolkit for full-stack web development.

At its heart, the MERN stack embodies the amalgamation of four potent technologies: MongoDB, Express.js, React.js, and Node.js. Each component plays a distinct yet interconnected role, empowering developers to craft modern, feature-rich web applications that resonate with today's tech-savvy users.

MongoDB, a frontrunner in NoSQL database technology, represents a paradigm shift in data management. Its document-oriented data model, built on flexible JSON-like documents, offers unmatched scalability and adaptability. With MongoDB, developers can effortlessly manage vast volumes of data while effortlessly adapting to changing application requirements.

Express.js, often hailed as the "Swiss Army knife" of web frameworks, forms the backbone of server-side development in the MERN stack. Built atop Node.js, Express.js streamlines the creation of APIs and management of HTTP requests, freeing developers to focus on building robust backend systems devoid of unnecessary boilerplate code.

React.js, born from the halls of Facebook, redefines frontend development with its declarative and component-based architecture. By decomposing intricate user interfaces into reusable and composable components, React.js empowers developers to craft highly interactive and responsive UIs. Its virtual DOM and one-way data flow model ensure optimal performance, even in the most demanding web applications. Completing the MERN stack ecosystem is Node.js, the cornerstone of server-side JavaScript. As a lightweight and efficient JavaScript runtime environment, Node.js blurs the lines between frontend and backend development.

The allure of the MERN stack lies not solely in its individual components but also in the synergy they generate when combined. By seamlessly integrating MongoDB, Express.js, React.js, and Node.js, developers harness the full power of JavaScript across the entire web development stack. This unified approach streamlines development workflows and fosters collaboration and innovation within development teams.

In this project documentation, we embark on a journey to unravel the intricacies of the MERN stack and unlock its full potential. Through a blend of theoretical insights, practical examples, and hands-on tutorials, our aim is to equip developers with the knowledge and skills necessary to thrive in the fast-paced realm of full-stack web development.

As we delve deeper into the MERN stack, we'll unearth best practices, design patterns, and optimization techniques that can elevate web application development to new heights. Whether you're a seasoned developer seeking to broaden your skill set or a newcomer eager to commence your web development journey, this documentation serves as your trusted companion, guiding you every step of the way. Welcome to the world of MERN stack development, where innovation knows no bounds, and the possibilities are limitless.

1.1 Domain

Full Stack Web Development

Full stack web development is the art and science of crafting fully functional and interactive web applications from top to bottom. It's the digital architecture that powers everything from dynamic e-commerce platforms to engaging social media networks, and it's the backbone of countless online experiences we encounter daily.

At its core, full stack web development involves handling every aspect of a web application's lifecycle. From designing and building the user-facing interface that visitors interact with, to managing the behind-the-scenes server infrastructure that stores and retrieves data, full stack developers are the architects who bring these digital worlds to life.

In this domain, developers navigate a vast landscape of technologies and frameworks, each serving a specific purpose in the grand symphony of web development. They harness the power of databases like MongoDB to securely store and organize vast amounts of data, ensuring seamless scalability as

user demand grows. They wield Express.js to create efficient server-side applications, handling incoming requests and orchestrating complex workflows with ease. They harness the versatility of React.js to craft dynamic user interfaces that adapt and respond to user input in real-time, creating rich and engaging experiences that captivate audiences.

But full stack web development is more than just writing code. It's about solving problems, embracing challenges, and continuously learning and evolving in a fast-paced digital landscape. It's about understanding the needs and desires of end-users, and crafting experiences that exceed their expectations. It's about collaboration, communication, and creativity, as developers work together to transform ideas into reality.

In this domain, possibilities are endless, and the journey is never-ending. From building the next groundbreaking application to refining and optimizing existing systems, full stack web development offers a world of opportunities for those who dare to dream and strive to innovate. And as technology continues to evolve and shape the world around us, full stack developers will remain at the forefront, shaping the digital landscape for generations to come.

1.2 Statement Of The Problem:

In an era marked by digital connectivity, the need for a dedicated social learning network for students has become increasingly apparent. Our project seeks to address this need by providing a platform tailored to the unique educational requirements of students. By fostering collaboration, knowledge sharing, and peer-to-peer support, our social learning network aims to enhance learning outcomes and promote a sense of community among students, thus bridging the gap between traditional education and modern technological advancements.

1.3 Objectives:

The Social Learning Network for Students project are to create a dynamic platform that cultivates collaborative learning experiences. We aim to foster a supportive environment where students can engage in meaningful discussions, share knowledge, and collaborate on projects. Through interactive features and user-friendly interfaces, we strive to enhance students' learning outcomes and promote a sense of belonging within the educational community. Our goal is to empower students to take ownership of their learning journey and thrive academically.

1.4 Scope:

Encompasses the development of a comprehensive platform tailored to the specific needs of students. This includes features such as discussion forums, collaborative project spaces, resource sharing capabilities, and interactive learning modules. Our focus is on creating a user-friendly interface that promotes active engagement and knowledge exchange among students while ensuring privacy, security, and accessibility. Additionally, we aim to integrate feedback mechanisms to continuously improve the platform's functionality and usability.

1.5 Applications:

Social Learning Network for Students project extend across various educational settings, from primary schools to higher education institutions. It serves as a valuable resource for facilitating remote learning, fostering peer-to-peer collaboration, and enhancing student engagement. Additionally, the platform can be utilized for professional development purposes, enabling educators to share resources and best practices. Furthermore, it provides opportunities for community building, networking, and extracurricular learning activities, enriching the overall educational experience.

1.6 Limitations:

Social Learning Network for Students project offers a robust platform for collaborative learning, it also faces certain limitations. These include potential challenges in user adoption and engagement, particularly in environments with limited internet access or digital literacy. Additionally, ensuring the privacy and security of user data poses ongoing concerns that require careful monitoring and implementation of appropriate safeguards. Furthermore, scalability issues may arise as the platform grows, necessitating continuous optimization and resource allocation.

CHAPTER 2

SYSTEM CONFIGURATION

2.1 Software Requirements:

The software requirements document outlines the specifications of the MERN stack system. It defines what the system should do rather than how it should do it, aiding in cost estimation, team planning, and tracking progress throughout development.

- Operating System : Windows
- Coding Language : JavaScript
- Front End : React JS
- Tools : Visual Studio Code, Git & GitHub

2.2 Hardware Requirements:

The hardware requirements provide a foundation for implementing the MERN stack system and should be a comprehensive specification of the entire setup.

- Hardware : Intel Core I3 Processor or higher
- RAM : 8 GigaBytes
- Hard Disk : 128 GigaBytes SSD

CHAPTER 3

LITERATURE SURVEY

3.1 Introduction:

The literature surrounding social learning networks for students encompasses a diverse range of topics, including educational technology, social networking, and pedagogical theories. With the increasing integration of technology in education, there has been a growing interest in leveraging social learning platforms to enhance student engagement, collaboration, and knowledge sharing. This literature survey aims to explore relevant research and best practices in the development and implementation of social learning networks tailored specifically for students.

3.2 Educational Technology and Social Learning:

Research in educational technology highlights the potential of social learning platforms to transform traditional learning environments by providing opportunities for interactive learning, peer-to-peer collaboration, and personalized learning experiences. Studies have shown that social learning networks can foster a sense of community among students, facilitate knowledge construction, and promote active participation in learning activities.

3.3 Design and Usability of Social Learning Platforms:

The design and usability of social learning platforms play a crucial role in facilitating effective communication, collaboration, and learning experiences among students. Research in this area emphasizes the importance of user-centered design principles, intuitive interface design, and accessibility features to ensure that social learning platforms are inclusive and user-friendly for students with diverse backgrounds and abilities.

CHAPTER 4

FEASIBILITY REPORT

4. Feasibility Study:

The feasibility report for the Social Learning Network for Students project assesses the viability and practicality of developing a web-based platform using the MERN (MongoDB, Express.js, React.js, Node.js) stack. This report examines various aspects of the project, including technical feasibility, economic feasibility, and operational feasibility, to determine the project's likelihood of success and identify potential challenges and risks.

4.1 Technical Feasibility:

The MERN stack, comprising MongoDB, Express.js, React.js, and Node.js, offers a robust and scalable framework for developing modern web applications. Each component of the stack provides unique capabilities and advantages, including flexibility, performance, and developer productivity.

4.2 Development Tools:

The availability of a wide range of development tools, libraries, and frameworks for each technology in the MERN stack facilitates rapid development and deployment of the social learning platform. Additionally, the active developer community and extensive documentation provide valuable resources and support for overcoming technical challenges.

4.3 Integration:

Integrating various components of the MERN stack seamlessly requires expertise in full-stack development and proficiency in JavaScript programming. Ensuring compatibility and interoperability between front-end and back-end components is critical for the smooth functioning of the social learning platform.

Economic Feasibility:

4.4 Development Costs:

The development of the Social Learning Network for Students project involves expenses related to software development, infrastructure setup, and personnel costs. While open-source technologies like the MERN stack reduce licensing fees, investment in skilled developers and infrastructure may constitute significant upfront costs.

4.5 Return on Investment (ROI):

The potential return on investment for the project depends on factors such as user adoption, monetization strategies, and scalability. By offering value-added features, premium subscriptions, or targeted advertising, the social learning platform can generate revenue streams to offset development costs and achieve profitability.

Operational Feasibility:

4.6 User Engagement:

The success of the social learning platform hinges on user engagement and adoption. Operational feasibility entails designing intuitive user interfaces, implementing robust features for collaboration and interaction, and providing ongoing support and maintenance to ensure a seamless user experience.

4.7 Scalability:

As the user base grows, the platform must scale efficiently to accommodate increased traffic and usage. Operational feasibility involves implementing scalable architecture, employing cloud-based hosting solutions, and monitoring system performance to meet evolving demands and ensure high availability and reliability.

CHAPTER 5

SYSTEM REQUIREMENT ANALYSIS

5.1 Input Design:

The input design for the Social Learning Network for Students project involves defining the various forms of user input that the system will accept. This includes user registration information, login credentials, profile details, course enrollment, discussion threads, text messaging and file uploads. Input validation mechanisms will be implemented to ensure data integrity and security. Additionally, intuitive user interfaces will be designed to streamline the input process and enhance user experience.

5.2 Output Design:

The output design focuses on presenting information and feedback to users in a clear, organized, and user-friendly manner. This includes displaying user profiles, text messages and discussion threads. Dynamic content generation and responsive design techniques will be employed to adapt the presentation of information based on the user's device and screen size. Emphasis will be placed on readability, accessibility, and visual appeal to enhance user engagement.

5.3 Existing System:

The existing system may comprise traditional learning management systems (LMS), generic social networking platforms, or standalone educational websites. These systems may lack features specifically tailored to the needs of students, such as seamless integration of social interaction with learning materials, personalized recommendations, and collaborative learning tools.

5.3.1 Disadvantages:

Lack of Integration:

Existing systems may not effectively integrate social networking features with educational content and activities, hindering collaborative learning opportunities.

Limited Personalization:

Generic platforms may offer limited customization options and fail to provide personalized learning experiences tailored to individual student preferences and needs.

Complexity:

Traditional LMS systems may be complex and cumbersome to navigate, leading to user frustration and disengagement.

Outdated Technology:

Some existing systems may rely on outdated technologies or lack support for modern web standards and mobile devices, resulting in compatibility issues and poor user experience.

5.4 Proposed System:

The proposed system, the Social Learning Network for Students, will address the limitations of existing systems by offering a comprehensive platform that seamlessly integrates social networking features with educational content and activities. Key features of the proposed system include:

User-Friendly Interface:

An intuitive and user-friendly interface will be designed to enhance accessibility and usability for students of all skill levels.

Personalized Learning Experience:

The system will leverage personalized learning paths, and targeted interventions based on individual student preferences and performance.

Collaborative Learning Tools:

Robust collaboration features, such as discussion forums, group projects, and peer-to-peer feedback mechanisms, will foster active learning and knowledge sharing among students.

Mobile Compatibility:

The system will be optimized for mobile devices, allowing students to access course materials, participate in discussions, and collaborate with peers anytime, anywhere.

Security and Privacy:

Stringent security measures will be implemented to protect user data and ensure compliance with data protection regulations, safeguarding the privacy and confidentiality of student information.

5.4.1 Advantages:

Enhanced Collaboration:

The proposed system will facilitate collaboration and knowledge sharing among students, fostering a sense of community and engagement.

Personalized Learning:

By offering personalized recommendations and adaptive learning paths, the system will cater to individual learning styles and preferences, enhancing learning outcomes.

Accessibility:

Mobile compatibility and user-friendly design will ensure that students can access educational resources and participate in learning activities conveniently.

Data-Driven Insights:

The system will provide educators with valuable insights into student engagement, performance, and learning behavior, enabling data-driven decision-making and targeted interventions.

Improved User Experience:

With its intuitive interface, collaborative tools, and personalized features, the proposed system will offer an engaging and rewarding learning experience for students, motivating them to actively participate in their education.

CHAPTER 6

SYSTEM DESIGN

6.1 System Architecture:

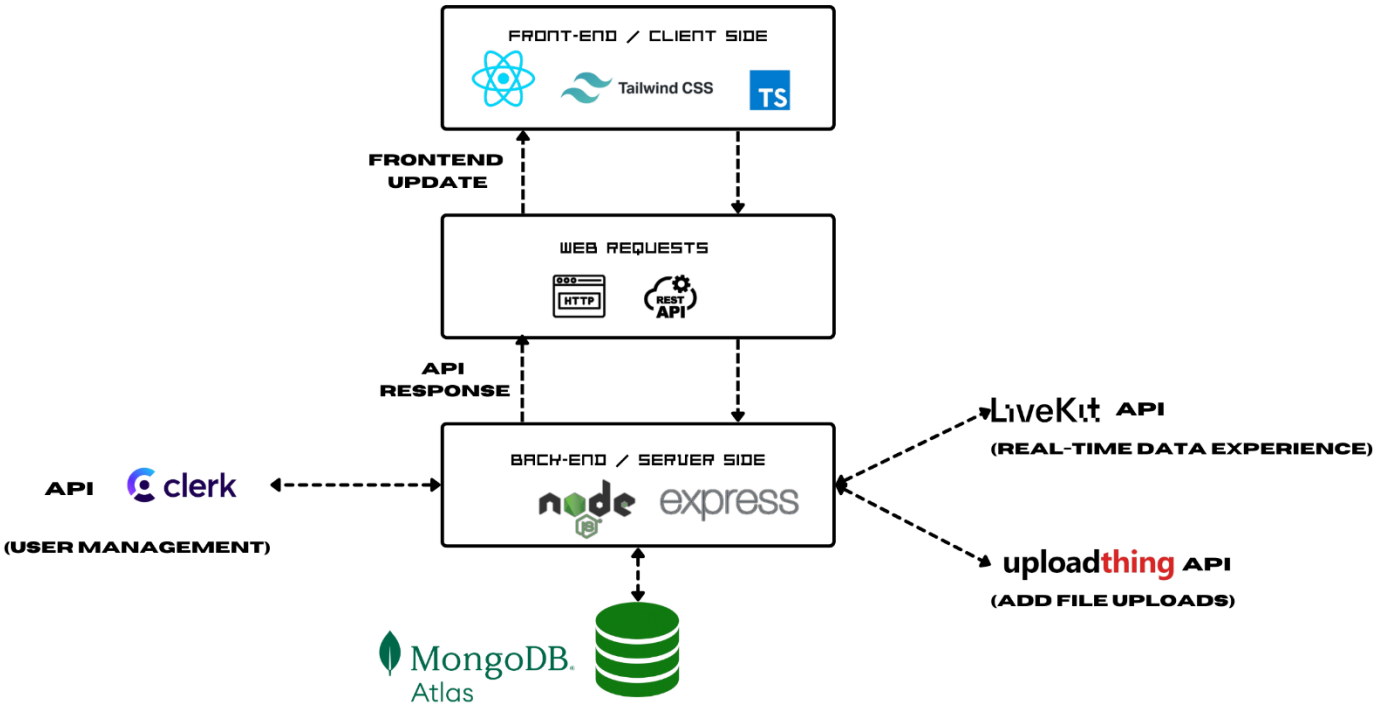
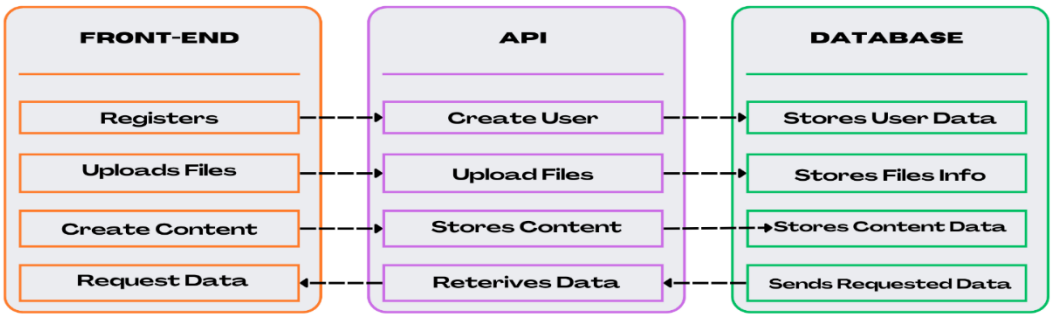


FIG : SOCIAL LEARN -SYSTEM ARCHITECTURE

6.2 Data Flow:



6.3 Usecase Diagram:

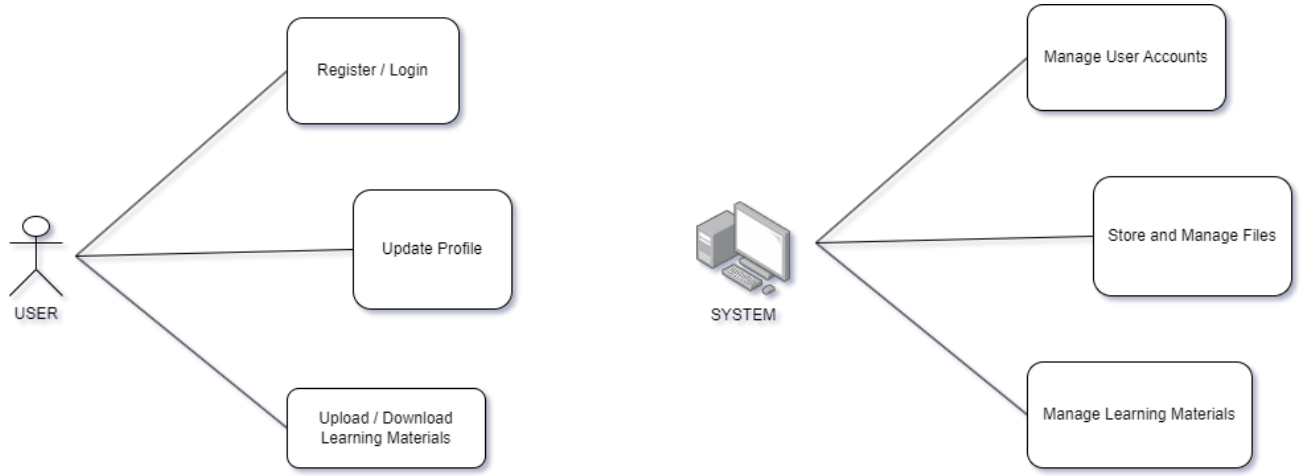


Fig :: Usecase Diagram

6.4 Sequence Diagram:

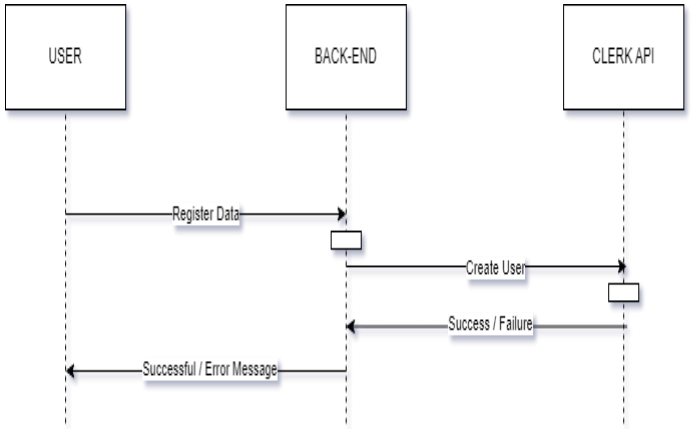


Fig :: Registration Sequence Diagram

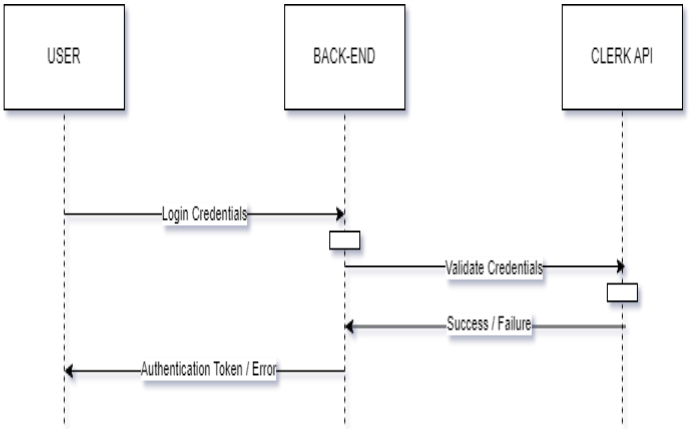


Fig :: Login Sequence Diagram

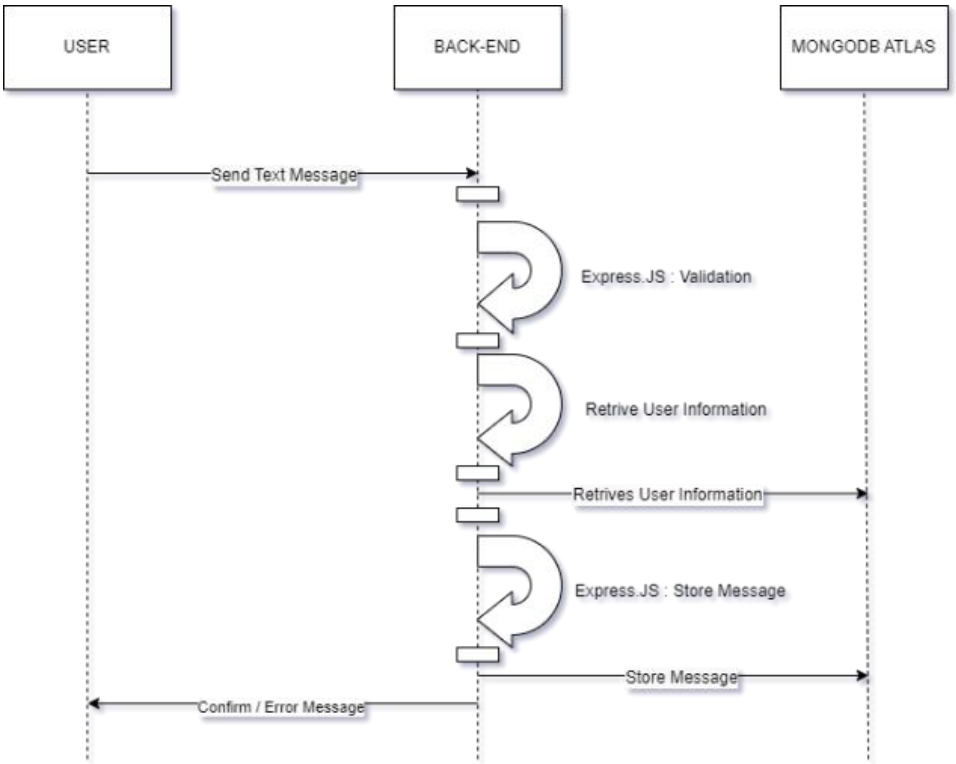


Fig :: Message Sending Sequence Diagram

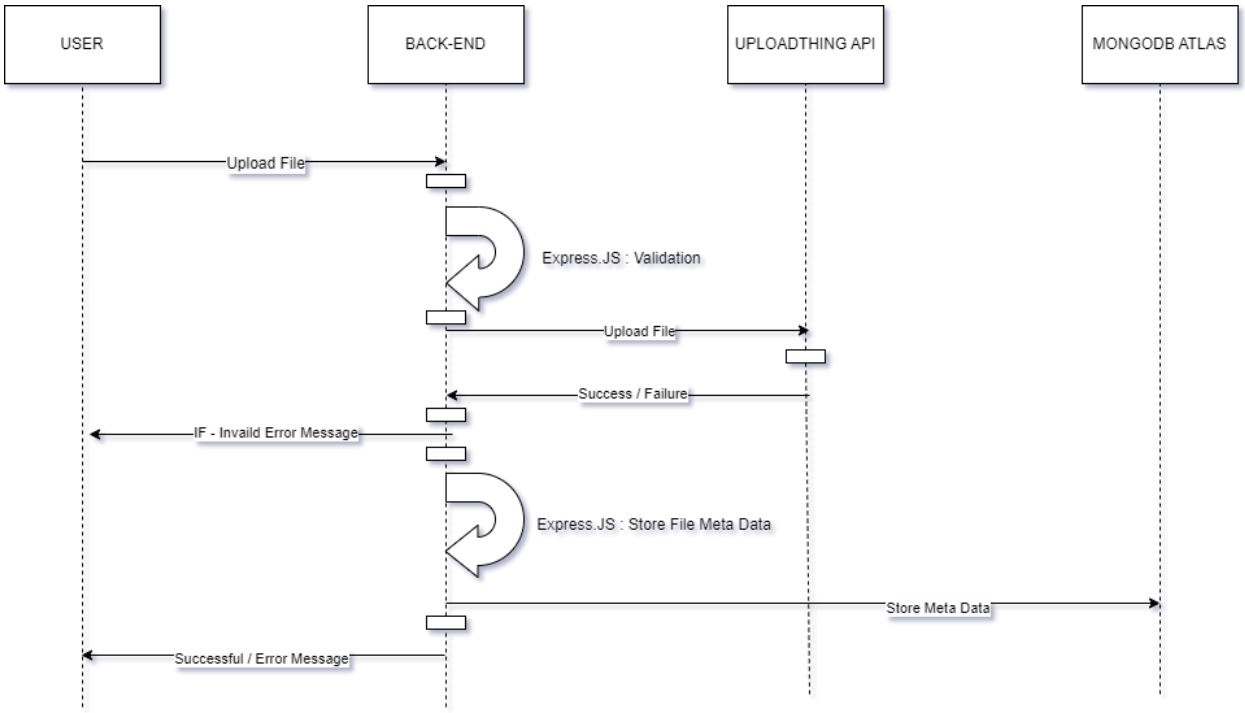


Fig :: File Upload Sequence Diagram

CHAPTER 7

PROPOSED SYSTEM AND ITS OBJECTIVES

7.1 Introduction:

The Social Learning Network for Students project proposes an innovative platform aimed at enhancing student engagement, collaboration, and learning outcomes. Leveraging modern technologies and educational methodologies, this system seeks to create a dynamic online environment where students can interact, share knowledge, and access educational resources seamlessly.

7.2 Existing System:

Currently, traditional learning management systems (LMS) offer limited interactivity and lack personalized learning experiences. Students often face challenges in accessing relevant course materials, collaborating with peers, and receiving timely feedback from instructors.

7.3 Proposed System:

The proposed system introduces a comprehensive social learning platform tailored specifically for students. Key features include:

- User-friendly interface for seamless navigation and interaction.
- Personalized user profiles with customizable preferences and settings.
- Discussion forums and chat functionality for real-time communication and collaboration among students and instructors.
- Integrated file upload functionality for sharing documents, presentations, and multimedia resources.
- Integration with external APIs such as Clerk API for user management and Uploadthing API for file uploads.
- Backend powered by the MERN stack (MongoDB, Express.js, React.js, Node.js) for scalability, flexibility, and performance.

7.4 Objectives:

The objectives of the proposed system are as follows:

- **Enhanced Student Engagement:** Foster a collaborative learning environment that encourages active participation and engagement among students.
- **Efficient Resource Sharing:** Facilitate seamless sharing of educational resources, including documents, presentations, and multimedia content, to support student learning and collaboration.
- **Real-time Communication:** Enable real-time communication and collaboration among students and instructors through discussion forums, chat functionality, and interactive sessions.
- **Scalability and Reliability:** Ensure the scalability, reliability, and performance of the platform to accommodate growing user bases and deliver a seamless user experience.
- **Integration with External APIs:** Integrate with external APIs such as Clerk API for user management and Uploadthing API for file uploads to enhance functionality and extend platform capabilities.

7.5 Modules:

The proposed system consists of several modules, including:

User Management:

Handles user registration, authentication, and profile management.

Communication:

Facilitates real-time communication and collaboration among students and instructors through discussion forums, chat functionality, and messaging features.

Resource Sharing:

Enables users to upload, share, and access educational resources such as documents, presentations, and multimedia content.

Administration:

Administer and manage the overall system, including user accounts, permissions, and system settings.

CHAPTER 8

SOFTWARE ENVIRONMENT

8.1 Software Environment:

The software environment for the Social Learning Network project encompasses a range of technologies tailored to support its functionalities and objectives. This environment is designed to provide a robust, scalable, and user-friendly platform for students to engage in collaborative learning activities.

8.2 Frontend Technologies:

HTML: Used for structuring the web pages and providing semantic markup.

Tailwind CSS: A utility-first CSS framework for rapidly building custom user interfaces.

TypeScript: A statically typed superset of JavaScript that adds optional static typing to the language.

React JS Framework: A popular JavaScript library for building user interfaces, providing component-based architecture and efficient rendering.

Visual Studio Code (VS Code): An open-source code editor developed by Microsoft, featuring support for debugging, syntax highlighting, and intelligent code completion, enhancing developer productivity.

Git & GitHub: A distributed version control system and web-based hosting service that facilitates collaboration, code management, and project sharing..

8.3 Backend Technologies:

Express JS: A minimalist web application framework for Node.js, used for building robust and scalable web applications and APIs.

Node JS: A JavaScript runtime environment that allows developers to run JavaScript code server-side, enabling building scalable network applications.

8.4 Communication Protocol:

HTTP: The Hypertext Transfer Protocol is the foundation of data communication on the World Wide Web, facilitating communication between clients and servers.

RESTful APIs: Representational State Transfer (REST) is an architectural style for designing networked applications. RESTful APIs adhere to REST principles and enable interaction with server resources.

8.5 Real-Time Data Experience:

LiveKit API: A platform for building real-time video, voice, and data experiences. LiveKit enables seamless integration of real-time communication features such as video conferencing, chat, and interactive whiteboarding within the Social Learning Network platform.

8.6 User management:

Clerk API: Provides authentication and user management services, enabling secure login, registration, and management of user accounts within the Social Learning Network platform.

8.7 File Upload:

Uploadthing API: Facilitates file upload functionality within the platform, allowing users to upload files securely and efficiently.

8.8 Database:

MongoDB: MongoDB Atlas: A fully managed cloud database service built on MongoDB, offering high availability, scalability, and security. MongoDB Atlas serves as the database backend for storing user data, content, and application state.

CHAPTER 9

SOURCE CODE

#C:\Users\6truy\Downloads\FSD\.env

```
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=pk_test_YWJvdmUtaG9yc2UtMC5jbGVyay5hY2NvdW50cy5kZXYk
CLERK_SECRET_KEY=sk_test_Ptnvj9Q8obFVtOJh5xV2qla9xwbyNWJiwOE0AHYDbd
NEXT_PUBLIC_CLERK_SIGN_IN_URL=/sign-in
NEXT_PUBLIC_CLERK_SIGN_UP_URL=/sign-up
NEXT_PUBLIC_CLERK_AFTER_SIGN_IN_URL=/
NEXT_PUBLIC_CLERK_AFTER_SIGN_UP_URL=/
DATABASE_URL="mongodb+srv://harisetty21:nYSmz2Ds5b446gbF@sociallearn.bupi3hq.mongodb.net/SocialLearn"
UPLOADTHING_SECRET=sk_live_5990975dc7bfa49a3479751dc7f080090833601990a690e23c740e1dec937161
UPLOADTHING_APP_ID=4r7lctbz27
LIVEKIT_API_KEY=APIRv9uj6ccud5A
LIVEKIT_API_SECRET=hDnGShr8zfykB5fw0pbo2xqvcLeXYGooqqpb4Qeu4CiA
NEXT_PUBLIC_LIVEKIT_URL=wss://social-learn-sy3svswq.livekit.cloud
```

#C:\Users\6truy\Downloads\FSD\package.json

```
{
  "name": "commspace",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "next dev",
    "build": "next build",
    "start": "next start",
    "lint": "next lint"
  },
  "dependencies": {
    "@aws-amplify/adapter-nextjs": "^1.0.28",
    "@aws-amplify/ui-react": "^6.1.8",
    "@clerk/nextjs": "^4.23.2",
    "@emoji-mart/data": "^1.1.2",
    "@emoji-mart/react": "^1.1.1",
    "@hookform/resolvers": "^3.3.0",
    "@livekit/components-react": "^1.1.3",
    "@livekit/components-styles": "^1.0.6",
    "@prisma/client": "^5.1.1",
    "@radix-ui/react-avatar": "^1.0.3",
    "@radix-ui/react-dialog": "^1.0.4",
    "@radix-ui/react-dropdown-menu": "^2.0.5",
```

```
    "@radix-ui/react-label": "^2.0.2",
    "@radix-ui/react-popover": "^1.0.6",
    "@radix-ui/react-scroll-area": "^1.0.4",
    "@radix-ui/react-select": "^1.2.2",
    "@radix-ui/react-separator": "^1.0.3",
    "@radix-ui/react-slot": "^1.0.2",
    "@radix-ui/react-tooltip": "^1.0.6",
    "@tanstack/react-query": "^4.33.0",
    "@types/node": "20.5.1",
    "@types/react": "18.2.20",
    "@types/react-dom": "18.2.7",
    "@uploadthing/react": "^5.3.0",
    "autoprefixer": "10.4.15",
    "aws-amplify": "^6.0.28",
    "axios": "^1.4.0",
    "class-variance-authority": "^0.7.0",
    "clsx": "^2.0.0",
    "cmdk": "^0.2.0",
    "date-fns": "^2.30.0",
    "emoji-mart": "^5.5.2",
    "eslint": "8.47.0",
    "eslint-config-next": "13.4.12",
    "install": "^0.13.0",
    "livekit-client": "^1.13.0",
    "livekit-server-sdk": "^1.2.6",
    "lucide-react": "^0.268.0",
    "next": "^13.5.6",
    "next-themes": "^0.2.1",
    "postcss": "8.4.28",
    "query-string": "^8.1.0",
    "react": "18.2.0",
    "react-dom": "18.2.0",
    "react-dropzone": "^14.2.3",
    "react-hook-form": "^7.45.4",
    "socket.io": "^4.7.2",
    "socket.io-client": "^4.7.2",
    "tailwind-merge": "^1.14.0",
    "tailwindcss": "3.3.3",
    "tailwindcss-animate": "^1.0.6",
    "uploadthing": "^5.3.3",
    "uuid": "^9.0.0",
    "zod": "^3.22.2",
    "zustand": "^4.4.1"
  },
  "devDependencies": {
    "@aws-amplify/backend": "^0.12.1",
    "@aws-amplify/backend-cli": "^0.11.1",
    "@types/uuid": "^9.0.2",
```

```
    "aws-cdk": "^2.138.0",
    "aws-cdk-lib": "^2.138.0",
    "constructs": "^10.3.0",
    "prisma": "^5.1.1",
    "typescript": "^5.4.5"
  }
}
```

#C:\Users\6truy\Downloads\FSD\tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  darkMode: ["class"],
  content: [
    './pages/**/*.{ts,tsx}',
    './components/**/*.{ts,tsx}',
    './app/**/*.{ts,tsx}',
    './src/**/*.{ts,tsx}',
  ],
  theme: {
    container: {
      center: true,
      padding: "2rem",
      screens: {
        "2xl": "1400px",
      },
    },
    extend: {
      colors: {
        border: "hsl(var(--border))",
        input: "hsl(var(--input))",
        ring: "hsl(var(--ring))",
        background: "hsl(var(--background))",
        foreground: "hsl(var(--foreground))",
        primary: {
          DEFAULT: "hsl(var(--primary))",
          foreground: "hsl(var(--primary-foreground))",
        },
        secondary: {
          DEFAULT: "hsl(var(--secondary))",
          foreground: "hsl(var(--secondary-foreground))",
        },
        destructive: {
          DEFAULT: "hsl(var(--destructive))",
          foreground: "hsl(var(--destructive-foreground))",
        },
        muted: {
```

```
          DEFAULT: "hsl(var(--muted))",
          foreground: "hsl(var(--muted-foreground))",
        },
        accent: {
          DEFAULT: "hsl(var(--accent))",
          foreground: "hsl(var(--accent-foreground))",
        },
        popover: {
          DEFAULT: "hsl(var(--popover))",
          foreground: "hsl(var(--popover-foreground))",
        },
        card: {
          DEFAULT: "hsl(var(--card))",
          foreground: "hsl(var(--card-foreground))",
        },
      },
      borderRadius: {
        lg: "var(--radius)",
        md: "calc(var(--radius) - 2px)",
        sm: "calc(var(--radius) - 4px)",
      },
      keyframes: {
        "accordion-down": {
          from: { height: 0 },
          to: { height: "var(--radix-accordion-content-height)" },
        },
        "accordion-up": {
          from: { height: "var(--radix-accordion-content-height)" },
          to: { height: 0 },
        },
      },
      animation: {
        "accordion-down": "accordion-down 0.2s ease-out",
        "accordion-up": "accordion-up 0.2s ease-out",
      },
    },
  },
  plugins: [require("tailwindcss-animate")],
}
```

#C:\Users\6truy\Downloads\FSD\prisma\schema.prisma

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "mongodb"
  url      = env("DATABASE_URL")
}

model Profile {
  id       String @id @default(auto()) @map("_id") @db.ObjectId
  userId   String @unique
  name     String
  imageUrl String @db.String
  email    String @db.String

  servers  Server[]
  members  Member[]
  channels Channel[]

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
}

model Server {
  id         String @id @default(auto()) @map("_id") @db.ObjectId
  name       String
  imageUrl   String @db.String
  inviteCode String @unique

  profileId String  @db.ObjectId
  profile   Profile @relation(fields: [profileId], references: [id], onDelete: Cascade)

  members Member[]

  channels Channel[]

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@index([profileId])
}

enum MemberRole {
  ADMIN
  MODERATOR
```

```
  GUEST
}

model Member {
  id   String     @id @default(auto()) @map("_id") @db.ObjectId
  role MemberRole @default(GUEST)

  profileId String  @db.ObjectId
  profile   Profile @relation(fields: [profileId], references: [id], onDelete: Cascade)

  serverId String @db.ObjectId
  server   Server @relation(fields: [serverId], references: [id], onDelete: Cascade)

  messages       Message[]
  directMessages DirectMessage[]

  conversationsInitiated Conversation[] @relation("MemberOne")
  conversationsReceived  Conversation[] @relation("MemberTwo")

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@index([profileId])
  @@index([serverId])
}

enum ChannelType {
  TEXT
  AUDIO
  VIDEO
}

model Channel {
  id   String      @id @default(auto()) @map("_id") @db.ObjectId
  name String
  type ChannelType @default(TEXT)

  profileId String  @db.ObjectId
  profile   Profile @relation(fields: [profileId], references: [id], onDelete: Cascade)

  serverId String @db.ObjectId
  server   Server @relation(fields: [serverId], references: [id], onDelete: Cascade)

  messages  Message[]
  createdAt DateTime  @default(now())
  updatedAt DateTime  @updatedAt

  @@index([profileId])
```

```
  @@index([serverId])
}

model Message {
  id      String @id @default(auto()) @map("_id") @db.ObjectId
  content String

  fileUrl String?

  memberId String @db.ObjectId
  member   Member @relation(fields: [memberId], references: [id], onDelete: Cascade)

  channelId String  @db.ObjectId
  channel   Channel @relation(fields: [channelId], references: [id], onDelete: Cascade)

  deleted Boolean @default(false)

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@index([channelId])
  @@index([memberId])
}

model Conversation {
  id String @id @default(auto()) @map("_id") @db.ObjectId

  memberOneId String @db.ObjectId
  memberOne       Member   @relation("MemberOne",   fields:   [memberOneId],   references:   [id],
onDelete: Cascade)

  memberTwoId String @db.ObjectId
  memberTwo       Member   @relation("MemberTwo",   fields:   [memberTwoId],   references:   [id],
onDelete: Cascade)

  directMessages DirectMessage[]

  @@unique([memberOneId, memberTwoId])
  @@index([memberTwoId])
}

model DirectMessage {
  id      String  @id @default(auto()) @map("_id") @db.ObjectId
  content String
  fileUrl String?

  memberId String @db.ObjectId
  member   Member @relation(fields: [memberId], references: [id], onDelete: Cascade)
```

```
  conversationId String       @db.ObjectId
  conversation   Conversation @relation(fields: [conversationId], references: [id], onDelete: Cascade)

  deleted Boolean @default(false)

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@index([memberId])
  @@index([conversationId])
}
```

#C:\Users\6truy\Downloads\FSD\pages\api\socket\direct-messages\[directMessageId].ts

```
import { NextApiRequest } from "next";
import { MemberRole } from "@prisma/client";

import { NextApiResponseServerIo } from "@/types";
import { currentProfilePages } from "@/lib/current-profile-pages";
import { db } from "@/lib/db";

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponseServerIo,
) {
  if (req.method !== "DELETE" && req.method !== "PATCH") {
    return res.status(405).json({ error: "Method not allowed" });
  }

  try {
    const profile = await currentProfilePages(req);
    const { directMessageId, conversationId } = req.query;
    const { content } = req.body;

    if (!profile) {
      return res.status(401).json({ error: "Unauthorized" });
    }

    if (!conversationId) {
      return res.status(400).json({ error: "Conversation ID missing" });
    }

    const conversation = await db.conversation.findFirst({
      where: {
```

```
        id: conversationId as string,
        OR: [
          {
            memberOne: {
              profileId: profile.id,
            }
          },
          {
            memberTwo: {
              profileId: profile.id,
            }
          }
        ]
      },
      include: {
        memberOne: {
          include: {
            profile: true,
          }
        },
        memberTwo: {
          include: {
            profile: true,
          }
        }
      }
    })

    if (!conversation) {
      return res.status(404).json({ error: "Conversation not found" });
    }

        const   member   =   conversation.memberOne.profileId   ===   profile.id   ?
conversation.memberOne : conversation.memberTwo;

    if (!member) {
      return res.status(404).json({ error: "Member not found" });
    }

    let directMessage = await db.directMessage.findFirst({
      where: {
        id: directMessageId as string,
        conversationId: conversationId as string,
      },
      include: {
        member: {
          include: {
            profile: true,
```

```
      }
    }
  }
})

if (!directMessage || directMessage.deleted) {
  return res.status(404).json({ error: "Message not found" });
}

const isMessageOwner = directMessage.memberId === member.id;
const isAdmin = member.role === MemberRole.ADMIN;
const isModerator = member.role === MemberRole.MODERATOR;
const canModify = isMessageOwner || isAdmin || isModerator;

if (!canModify) {
  return res.status(401).json({ error: "Unauthorized" });
}

if (req.method === "DELETE") {
  directMessage = await db.directMessage.update({
    where: {
      id: directMessageId as string,
    },
    data: {
      fileUrl: null,
      content: "This message has been deleted.",
      deleted: true,
    },
    include: {
      member: {
        include: {
          profile: true,
        }
      }
    }
  })
}

if (req.method === "PATCH") {
  if (!isMessageOwner) {
    return res.status(401).json({ error: "Unauthorized" });
  }

  directMessage = await db.directMessage.update({
    where: {
      id: directMessageId as string,
    },
    data: {
```

```
          content,
        },
        include: {
          member: {
            include: {
              profile: true,
            }
          }
        }
      })
    }

    const updateKey = `chat:${conversation.id}:messages:update`;

    res?.socket?.server?.io?.emit(updateKey, directMessage);

    return res.status(200).json(directMessage);
  } catch (error) {
    console.log("[MESSAGE_ID]", error);
    return res.status(500).json({ error: "Internal Error" });
  }
}
```

#C:\Users\6truy\Downloads\FSD\pages\api\socket\direct-messages\index.ts

```
import { NextApiRequest } from "next";

import { NextApiResponseServerIo } from "@/types";
import { currentProfilePages } from "@/lib/current-profile-pages";
import { db } from "@/lib/db";

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponseServerIo,
) {
  if (req.method !== "POST") {
    return res.status(405).json({ error: "Method not allowed" });
  }

  try {
    const profile = await currentProfilePages(req);
    const { content, fileUrl } = req.body;
    const { conversationId } = req.query;

    if (!profile) {
      return res.status(401).json({ error: "Unauthorized" });
    }
```

```
    if (!conversationId) {
      return res.status(400).json({ error: "Conversation ID missing" });
    }

    if (!content) {
      return res.status(400).json({ error: "Content missing" });
    }


    const conversation = await db.conversation.findFirst({
      where: {
        id: conversationId as string,
        OR: [
          {
            memberOne: {
              profileId: profile.id,
            }
          },
          {
            memberTwo: {
              profileId: profile.id,
            }
          }
        ]
      },
      include: {
        memberOne: {
          include: {
            profile: true,
          }
        },
        memberTwo: {
          include: {
            profile: true,
          }
        }
      }
    })

    if (!conversation) {
      return res.status(404).json({ message: "Conversation not found" });
    }

        const   member   =   conversation.memberOne.profileId   ===   profile.id   ?
conversation.memberOne : conversation.memberTwo

    if (!member) {
      return res.status(404).json({ message: "Member not found" });
    }
```

```
    const message = await db.directMessage.create({
      data: {
        content,
        fileUrl,
        conversationId: conversationId as string,
        memberId: member.id,
      },
      include: {
        member: {
          include: {
            profile: true,
          }
        }
      }
    });

    const channelKey = `chat:${conversationId}:messages`;

    res?.socket?.server?.io?.emit(channelKey, message);

    return res.status(200).json(message);
  } catch (error) {
    console.log("[DIRECT_MESSAGES_POST]", error);
    return res.status(500).json({ message: "Internal Error" });
  }
}
```

#C:\Users\6truy\Downloads\FSD\pages\api\socket\messages\[messageId].ts

```
import { NextApiRequest } from "next";
import { MemberRole } from "@prisma/client";

import { NextApiResponseServerIo } from "@/types";
import { currentProfilePages } from "@/lib/current-profile-pages";
import { db } from "@/lib/db";

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponseServerIo,
) {
  if (req.method !== "DELETE" && req.method !== "PATCH") {
    return res.status(405).json({ error: "Method not allowed" });
  }

  try {
    const profile = await currentProfilePages(req);
```

```
const { messageId, serverId, channelId } = req.query;
const { content } = req.body;

if (!profile) {
  return res.status(401).json({ error: "Unauthorized" });
}

if (!serverId) {
  return res.status(400).json({ error: "Server ID missing" });
}

if (!channelId) {
  return res.status(400).json({ error: "Channel ID missing" });
}

const server = await db.server.findFirst({
  where: {
    id: serverId as string,
    members: {
      some: {
        profileId: profile.id,
      }
    }
  },
  include: {
    members: true,
  }
})

if (!server) {
  return res.status(404).json({ error: "Server not found" });
}

const channel = await db.channel.findFirst({
  where: {
    id: channelId as string,
    serverId: serverId as string,
  },
});

if (!channel) {
  return res.status(404).json({ error: "Channel not found" });
}

const member = server.members.find((member) => member.profileId === profile.id);

if (!member) {
  return res.status(404).json({ error: "Member not found" });
```

```
}

let message = await db.message.findFirst({
  where: {
    id: messageId as string,
    channelId: channelId as string,
  },
  include: {
    member: {
      include: {
        profile: true,
      }
    }
  }
})

if (!message || message.deleted) {
  return res.status(404).json({ error: "Message not found" });
}

const isMessageOwner = message.memberId === member.id;
const isAdmin = member.role === MemberRole.ADMIN;
const isModerator = member.role === MemberRole.MODERATOR;
const canModify = isMessageOwner || isAdmin || isModerator;

if (!canModify) {
  return res.status(401).json({ error: "Unauthorized" });
}

if (req.method === "DELETE") {
  message = await db.message.update({
    where: {
      id: messageId as string,
    },
    data: {
      fileUrl: null,
      content: "This message has been deleted.",
      deleted: true,
    },
    include: {
      member: {
        include: {
          profile: true,
        }
      }
    }
  })
}
```

```
    if (req.method === "PATCH") {
      if (!isMessageOwner) {
        return res.status(401).json({ error: "Unauthorized" });
      }

      message = await db.message.update({
        where: {
          id: messageId as string,
        },
        data: {
          content,
        },
        include: {
          member: {
            include: {
              profile: true,
            }
          }
        }
      })
    }

    const updateKey = `chat:${channelId}:messages:update`;

    res?.socket?.server?.io?.emit(updateKey, message);

    return res.status(200).json(message);
  } catch (error) {
    console.log("[MESSAGE_ID]", error);
    return res.status(500).json({ error: "Internal Error" });
  }
}
```

#C:\Users\6truy\Downloads\FSD\pages\api\socket\messages\index.ts

```
import { NextApiRequest } from "next";

import { NextApiResponseServerIo } from "@/types";
import { currentProfilePages } from "@/lib/current-profile-pages";
import { db } from "@/lib/db";

export default async function handler(
  req: NextApiRequest,
  res: NextApiResponseServerIo,
) {
  if (req.method !== "POST") {
```

```
  return res.status(405).json({ error: "Method not allowed" });
}

try {
  const profile = await currentProfilePages(req);
  const { content, fileUrl } = req.body;
  const { serverId, channelId } = req.query;

  if (!profile) {
    return res.status(401).json({ error: "Unauthorized" });
  }

  if (!serverId) {
    return res.status(400).json({ error: "Server ID missing" });
  }

  if (!channelId) {
    return res.status(400).json({ error: "Channel ID missing" });
  }

  if (!content) {
    return res.status(400).json({ error: "Content missing" });
  }

  const server = await db.server.findFirst({
    where: {
      id: serverId as string,
      members: {
        some: {
          profileId: profile.id
        }
      }
    },
    include: {
      members: true,
    }
  });

  if (!server) {
    return res.status(404).json({ message: "Server not found" });
  }

  const channel = await db.channel.findFirst({
    where: {
      id: channelId as string,
      serverId: serverId as string,
    }
  });
```

```
    if (!channel) {
      return res.status(404).json({ message: "Channel not found" });
    }

    const member = server.members.find((member) => member.profileId === profile.id);

    if (!member) {
      return res.status(404).json({ message: "Member not found" });
    }

    const message = await db.message.create({
      data: {
        content,
        fileUrl,
        channelId: channelId as string,
        memberId: member.id,
      },
      include: {
        member: {
          include: {
            profile: true,
          }
        }
      }
    });

    const channelKey = `chat:${channelId}:messages`;

    res?.socket?.server?.io?.emit(channelKey, message);

    return res.status(200).json(message);
  } catch (error) {
    console.log("[MESSAGES_POST]", error);
    return res.status(500).json({ message: "Internal Error" });
  }
}
```

#C:\Users\6truy\Downloads\FSD\lib\db.ts

```
import { PrismaClient } from "@prisma/client";

declare global {
  var prisma: PrismaClient | undefined;
};

export const db = globalThis.prisma || new PrismaClient();
```

```
if (process.env.NODE_ENV !== "production") globalThis.prisma = db
```

#C:\Users\6truy\Downloads\FSD\lib\initial-profile.ts

```
import { currentUser, redirectToSignIn } from "@clerk/nextjs";

import { db } from "@/lib/db";

export const initialProfile = async () => {
  const user = await currentUser();

  if (!user) {
    return redirectToSignIn();
  }

  const profile = await db.profile.findUnique({
    where: {
      userId: user.id
    }
  });

  if (profile) {
    return profile;
  }

  const newProfile = await db.profile.create({
    data: {
      userId: user.id,
      name: `${user.firstName} ${user.lastName}`,
      imageUrl: user.imageUrl,
      email: user.emailAddresses[0].emailAddress
    }
  });

  return newProfile;
};
```

#C:\Users\6truy\Downloads\FSD\lib\uploadthing.ts

```
import { generateComponents } from "@uploadthing/react";

import type { OurFileRouter } from "@/app/api/uploadthing/core";

export const { UploadButton, UploadDropzone, Uploader } =
  generateComponents<OurFileRouter>();
```

#C:\Users\6truy\Downloads\FSD\app\(auth)\(routes)\sign-in\[[...sign-in]]\page.tsx

```
import { SignIn } from "@clerk/nextjs";

export default function Page() {
  return <SignIn />;
}
```

#C:\Users\6truy\Downloads\FSD\app\(auth)\(routes)\sign-up\[[...sign-up]]\page.tsx

```
import { SignUp } from "@clerk/nextjs";

export default function Page() {
  return <SignUp />;
}
```

#C:\Users\6truy\Downloads\FSD\app\(main)\(routes)\servers\[serverId]\channels\[channelId]\page.tsx

```
import { redirectToSignIn } from "@clerk/nextjs";
import { redirect } from "next/navigation";
import { ChannelType } from "@prisma/client";

import { currentProfile } from "@/lib/current-profile";
import { ChatHeader } from "@/components/chat/chat-header";
import { ChatInput } from "@/components/chat/chat-input";
import { ChatMessages } from "@/components/chat/chat-messages";
import { MediaRoom } from "@/components/media-room";
import { db } from "@/lib/db";

interface ChannelIdPageProps {
  params: {
    serverId: string;
    channelId: string;
  }
}

const ChannelIdPage = async ({
  params
}: ChannelIdPageProps) => {
  const profile = await currentProfile();

  if (!profile) {
```

```
  return redirectToSignIn();
}

const channel = await db.channel.findUnique({
  where: {
    id: params.channelId,
  },
});

const member = await db.member.findFirst({
  where: {
    serverId: params.serverId,
    profileId: profile.id,
  }
});

if (!channel || !member) {
  redirect("/");
}

return (
  <div className="bg-white dark:bg-[#313338] flex flex-col h-full">
    <ChatHeader
      name={channel.name}
      serverId={channel.serverId}
      type="channel"
    />
    {channel.type === ChannelType.TEXT && (
      <>
        <ChatMessages
          member={member}
          name={channel.name}
          chatId={channel.id}
          type="channel"
          apiUrl="/api/messages"
          socketUrl="/api/socket/messages"
          socketQuery={{
            channelId: channel.id,
            serverId: channel.serverId,
          }}
          paramKey="channelId"
          paramValue={channel.id}
        />
        <ChatInput
          name={channel.name}
          type="channel"
          apiUrl="/api/socket/messages"
          query={{
```

```
              channelId: channel.id,
              serverId: channel.serverId,
            }}
          />
        </>
      )}
      {channel.type === ChannelType.AUDIO && (
        <MediaRoom
          chatId={channel.id}
          video={false}
          audio={true}
        />
      )}
      {channel.type === ChannelType.VIDEO && (
        <MediaRoom
          chatId={channel.id}
          video={true}
          audio={true}
        />
      )}
    </div>
  );
}

export default ChannelIdPage;
```

#C:\Users\6truy\Downloads\FSD\app\(setup)\page.tsx

```
import { redirect } from "next/navigation";

import { db } from "@/lib/db";
import { initialProfile } from "@/lib/initial-profile";
import { InitialModal } from "@/components/modals/initial-modal";

const SetupPage = async () => {
  const profile = await initialProfile();

  const server = await db.server.findFirst({
    where: {
      members: {
        some: {
          profileId: profile.id
        }
      }
    }
  });

  if (server) {
```

```
    return redirect(`/servers/${server.id}`);
  }

  return <InitialModal />;
}

export default SetupPage;
```

#C:\Users\6truy\Downloads\FSD\app\api\livekit\route.ts

```
import { AccessToken } from "livekit-server-sdk";
import { NextRequest, NextResponse } from "next/server";

export async function GET(req: NextRequest) {
  const room = req.nextUrl.searchParams.get("room");
  const username = req.nextUrl.searchParams.get("username");
  if (!room) {
    return NextResponse.json({ error: 'Missing "room" query parameter' }, { status:
400 });
  } else if (!username) {
     return NextResponse.json({ error: 'Missing "username" query parameter' }, {
status: 400 });
  }

  const apiKey = process.env.LIVEKIT_API_KEY;
  const apiSecret = process.env.LIVEKIT_API_SECRET;
  const wsUrl = process.env.NEXT_PUBLIC_LIVEKIT_URL;

  if (!apiKey || !apiSecret || !wsUrl) {
    return NextResponse.json({ error: "Server misconfigured" }, { status: 500 });
  }

  const at = new AccessToken(apiKey, apiSecret, { identity: username });

  at.addGrant({ room, roomJoin: true, canPublish: true, canSubscribe: true });

  return NextResponse.json({ token: at.toJwt() });
}
```

#C:\Users\6truy\Downloads\FSD\app\api\servers\[serverId]\invite-code\route.ts

```
import { v4 as uuidv4 } from "uuid";
import { NextResponse } from "next/server";

import { currentProfile } from "@/lib/current-profile";
import { db } from "@/lib/db";

export async function PATCH(
```

```
  req: Request,
  { params }: { params: { serverId: string } }
) {
  try {
    const profile = await currentProfile();

    if (!profile) {
      return new NextResponse("Unauthorized", { status: 401 });
    }

    if (!params.serverId) {
      return new NextResponse("Server ID Missing", { status: 400 });
    }

    const server = await db.server.update({
      where: {
        id: params.serverId,
        profileId: profile.id,
      },
      data: {
        inviteCode: uuidv4(),
      },
    });

    return NextResponse.json(server);
  } catch (error) {
    console.log("[SERVER_ID]", error);
    return new NextResponse("Internal Error", { status: 500 });
  }
}
```

#C:\Users\6truy\Downloads\FSD\app\api\uploadthing\core.ts

```
import { auth } from "@clerk/nextjs";
import { createUploadthing, type FileRouter } from "uploadthing/next";

const f = createUploadthing();

const handleAuth = () => {
  const { userId } = auth();
  if (!userId) throw new Error("Unauthorized");
  return { userId: userId };
}

export const ourFileRouter = {
  serverImage: f({ image: { maxFileSize: "4MB", maxFileCount: 1 } })
    .middleware(() => handleAuth())
    .onUploadComplete(() => {}),
```

```
  messageFile: f(["image", "pdf"])
    .middleware(() => handleAuth())
    .onUploadComplete(() => {})
} satisfies FileRouter;

export type OurFileRouter = typeof ourFileRouter;
```

#C:\Users\6truy\Downloads\FSD\app\api\uploadthing\route.ts

```
import { createNextRouteHandler } from "uploadthing/next";

import { ourFileRouter } from "./core";

// Export routes for Next App Router
export const { GET, POST } = createNextRouteHandler({
  router: ourFileRouter,
});
```

#C:\Users\6truy\Downloads\FSD\app\globals.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

html,
body,
:root {
  height: 100%;
}

@layer base {
  :root {
    --background: 0 0% 100%;
    --foreground: 20 14.3% 4.1%;

    --card: 0 0% 100%;
    --card-foreground: 20 14.3% 4.1%;

    --popover: 0 0% 100%;
    --popover-foreground: 20 14.3% 4.1%;

    --primary: 24 9.8% 10%;
    --primary-foreground: 60 9.1% 97.8%;

    --secondary: 60 4.8% 95.9%;
    --secondary-foreground: 24 9.8% 10%;

    --muted: 60 4.8% 95.9%;
```

```
    --muted-foreground: 25 5.3% 44.7%;

    --accent: 60 4.8% 95.9%;
    --accent-foreground: 24 9.8% 10%;

    --destructive: 0 84.2% 60.2%;
    --destructive-foreground: 60 9.1% 97.8%;

    --border: 20 5.9% 90%;
    --input: 20 5.9% 90%;
    --ring: 20 14.3% 4.1%;

    --radius: 0.5rem;
  }

  .dark {
    --background: 20 14.3% 4.1%;
    --foreground: 60 9.1% 97.8%;

    --card: 20 14.3% 4.1%;
    --card-foreground: 60 9.1% 97.8%;

    --popover: 20 14.3% 4.1%;
    --popover-foreground: 60 9.1% 97.8%;

    --primary: 60 9.1% 97.8%;
    --primary-foreground: 24 9.8% 10%;

    --secondary: 12 6.5% 15.1%;
    --secondary-foreground: 60 9.1% 97.8%;

    --muted: 12 6.5% 15.1%;
    --muted-foreground: 24 5.4% 63.9%;

    --accent: 12 6.5% 15.1%;
    --accent-foreground: 60 9.1% 97.8%;

    --destructive: 0 62.8% 30.6%;
    --destructive-foreground: 60 9.1% 97.8%;

    --border: 12 6.5% 15.1%;
    --input: 12 6.5% 15.1%;
    --ring: 24 5.7% 82.9%;
  }
}

@layer base {
  * {
```

```
    @apply border-border;
  }
  body {
    @apply bg-background text-foreground;
  }
}
```

#C:\Users\6truy\Downloads\FSD\app\layout.tsx

```
import './globals.css'
import type { Metadata } from 'next'
import { Open_Sans } from 'next/font/google'
import { ClerkProvider } from '@clerk/nextjs'

import { cn } from '@/lib/utils'
import { ThemeProvider } from '@/components/providers/theme-provider'
import { ModalProvider } from '@/components/providers/modal-provider'
import { SocketProvider } from '@/components/providers/socket-provider'
import { QueryProvider } from '@/components/providers/query-provider'

const font = Open_Sans({ subsets: ['latin'] })

export const metadata: Metadata = {
  title: 'DevX-Social Learning Network',
  description: 'Social learning Network For Students',
}

export default function RootLayout({
  children,
}: {
  children: React.ReactNode
}) {
  return (
    <ClerkProvider>
      <html lang="en" suppressHydrationWarning>
        <body className={cn(
          font.className,
          "bg-white dark:bg-[#313338]"
        )}>
          <ThemeProvider
            attribute="class"
            defaultTheme="dark"
            enableSystem={false}
            storageKey="discord-theme"
          >
            <SocketProvider>
              <ModalProvider />
              <QueryProvider>
```

```
                {children}
              </QueryProvider>
            </SocketProvider>
          </ThemeProvider>
        </body>
      </html>
    </ClerkProvider>
  )
}
```

#C:\Users\6truy\Downloads\FSD\components\chat\chat-header.tsx

```
import { Hash } from "lucide-react";

import { MobileToggle } from "@/components/mobile-toggle";
import { UserAvatar } from "@/components/user-avatar";
import { SocketIndicator } from "@/components/socket-indicator";

import { ChatVideoButton } from "./chat-video-button";

interface ChatHeaderProps {
  serverId: string;
  name: string;
  type: "channel" | "conversation";
  imageUrl?: string;
}

export const ChatHeader = ({
  serverId,
  name,
  type,
  imageUrl
}: ChatHeaderProps) => {
  return (
      <div  className="text-md  font-semibold  px-3  flex  items-center  h-12  border-
neutral-200 dark:border-neutral-800 border-b-2">
      <MobileToggle serverId={serverId} />
      {type === "channel" && (
        <Hash className="w-5 h-5 text-zinc-500 dark:text-zinc-400 mr-2" />
      )}
      {type === "conversation" && (
        <UserAvatar
          src={imageUrl}
          className="h-8 w-8 md:h-8 md:w-8 mr-2"
```

```
        />
      )}
      <p className="font-semibold text-md text-black dark:text-white">
        {name}
      </p>
      <div className="ml-auto flex items-center">
        {type === "conversation" && (
          <ChatVideoButton />
        )}
        <SocketIndicator />
      </div>
    </div>
  )
}
```

#C:\Users\6truy\Downloads\FSD\components\chat\chat-messages.tsx

```
"use client";

import { Fragment, useRef, ElementRef } from "react";
import { format } from "date-fns";
import { Member, Message, Profile } from "@prisma/client";
import { Loader2, ServerCrash } from "lucide-react";

import { useChatQuery } from "@/hooks/use-chat-query";
import { useChatSocket } from "@/hooks/use-chat-socket";
import { useChatScroll } from "@/hooks/use-chat-scroll";

import { ChatWelcome } from "./chat-welcome";
import { ChatItem } from "./chat-item";

const DATE_FORMAT = "d MMM yyyy, HH:mm";

type MessageWithMemberWithProfile = Message & {
  member: Member & {
    profile: Profile
  }
}

interface ChatMessagesProps {
  name: string;
  member: Member;
  chatId: string;
  apiUrl: string;
  socketUrl: string;
  socketQuery: Record<string, string>;
  paramKey: "channelId" | "conversationId";
  paramValue: string;
```

```
  type: "channel" | "conversation";
}

export const ChatMessages = ({
  name,
  member,
  chatId,
  apiUrl,
  socketUrl,
  socketQuery,
  paramKey,
  paramValue,
  type,
}: ChatMessagesProps) => {
  const queryKey = `chat:${chatId}`;
  const addKey = `chat:${chatId}:messages`;
  const updateKey = `chat:${chatId}:messages:update`

  const chatRef = useRef<ElementRef<"div">>(null);
  const bottomRef = useRef<ElementRef<"div">>(null);

  const {
    data,
    fetchNextPage,
    hasNextPage,
    isFetchingNextPage,
    status,
  } = useChatQuery({
    queryKey,
    apiUrl,
    paramKey,
    paramValue,
  });
  useChatSocket({ queryKey, addKey, updateKey });
  useChatScroll({
    chatRef,
    bottomRef,
    loadMore: fetchNextPage,
    shouldLoadMore: !isFetchingNextPage && !!hasNextPage,
    count: data?.pages?.[0]?.items?.length ?? 0,
  })

  if (status === "loading") {
    return (
      <div className="flex flex-col flex-1 justify-center items-center">
        <Loader2 className="h-7 w-7 text-zinc-500 animate-spin my-4" />
        <p className="text-xs text-zinc-500 dark:text-zinc-400">
          Loading messages...
```

```
        </p>
      </div>
    )
  }

  if (status === "error") {
    return (
      <div className="flex flex-col flex-1 justify-center items-center">
        <ServerCrash className="h-7 w-7 text-zinc-500 my-4" />
        <p className="text-xs text-zinc-500 dark:text-zinc-400">
          Something went wrong!
        </p>
      </div>
    )
  }

  return (
    <div ref={chatRef} className="flex-1 flex flex-col py-4 overflow-y-auto">
      {!hasNextPage && <div className="flex-1" />}
      {!hasNextPage && (
        <ChatWelcome
          type={type}
          name={name}
        />
      )}
      {hasNextPage && (
        <div className="flex justify-center">
          {isFetchingNextPage ? (
            <Loader2 className="h-6 w-6 text-zinc-500 animate-spin my-4" />
          ) : (
            <button
              onClick={() => fetchNextPage()}
               className="text-zinc-500 hover:text-zinc-600 dark:text-zinc-400 text-
xs my-4 dark:hover:text-zinc-300 transition"
            >
              Load previous messages
            </button>
          )}
        </div>
      )}
      <div className="flex flex-col-reverse mt-auto">
        {data?.pages?.map((group, i) => (
          <Fragment key={i}>
            {group.items.map((message: MessageWithMemberWithProfile) => (
              <ChatItem
                key={message.id}
                id={message.id}
                currentMember={member}
```

```
                member={message.member}
                content={message.content}
                fileUrl={message.fileUrl}
                deleted={message.deleted}
                timestamp={format(new Date(message.createdAt), DATE_FORMAT)}
                isUpdated={message.updatedAt !== message.createdAt}
                socketUrl={socketUrl}
                socketQuery={socketQuery}
              />
            ))}
          </Fragment>
        ))}
      </div>
      <div ref={bottomRef} />
    </div>
  )
}
```

#C:\Users\6truy\Downloads\FSD\components\modals\create-channel-modal.tsx

```
"use client";

import qs from "query-string";
import axios from "axios";
import * as z from "zod";
import { zodResolver } from "@hookform/resolvers/zod";
import { useForm } from "react-hook-form";
import { ChannelType } from "@prisma/client";

import {
  Dialog,
  DialogContent,
  DialogFooter,
  DialogHeader,
  DialogTitle,
} from "@/components/ui/dialog";
import {
  Form,
  FormControl,
  FormField,
  FormItem,
  FormLabel,
  FormMessage
} from "@/components/ui/form";
import { Input } from "@/components/ui/input";
import { Button } from "@/components/ui/button";
import { useParams, useRouter } from "next/navigation";
import { useModal } from "@/hooks/use-modal-store";
```

```
import {
  Select,
  SelectContent,
  SelectItem,
  SelectTrigger,
  SelectValue
} from "@/components/ui/select";
import { useEffect } from "react";

const formSchema = z.object({
  name: z.string().min(1, {
    message: "Channel name is required."
  }).refine(
    name => name !== "general",
    {
      message: "Channel name cannot be 'general'"
    }
  ),
  type: z.nativeEnum(ChannelType)
});

export const CreateChannelModal = () => {
  const { isOpen, onClose, type, data } = useModal();
  const router = useRouter();
  const params = useParams();

  const isModalOpen = isOpen && type === "createChannel";
  const { channelType } = data;

  const form = useForm({
    resolver: zodResolver(formSchema),
    defaultValues: {
      name: "",
      type: channelType || ChannelType.TEXT,
    }
  });

  useEffect(() => {
    if (channelType) {
      form.setValue("type", channelType);
    } else {
      form.setValue("type", ChannelType.TEXT);
    }
  }, [channelType, form]);

  const isLoading = form.formState.isSubmitting;

  const onSubmit = async (values: z.infer<typeof formSchema>) => {
```

```
    try {
      const url = qs.stringifyUrl({
        url: "/api/channels",
        query: {
          serverId: params?.serverId
        }
      });
      await axios.post(url, values);

      form.reset();
      router.refresh();
      onClose();
    } catch (error) {
      console.log(error);
    }
  }

  const handleClose = () => {
    form.reset();
    onClose();
  }

  return (
    <Dialog open={isModalOpen} onOpenChange={handleClose}>
      <DialogContent className="bg-white text-black p-0 overflow-hidden">
        <DialogHeader className="pt-8 px-6">
          <DialogTitle className="text-2xl text-center font-bold">
            Create Channel
          </DialogTitle>
        </DialogHeader>
        <Form {...form}>
          <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-8">
            <div className="space-y-8 px-6">
              <FormField
                control={form.control}
                name="name"
                render={({ field }) => (
                  <FormItem>
                    <FormLabel
                              className="uppercase  text-xs  font-bold  text-zinc-500
dark:text-secondary/70"
                    >
                      Channel name
                    </FormLabel>
                    <FormControl>
                      <Input
                        disabled={isLoading}
```

```
                          className="bg-zinc-300/50 border-0 focus-visible:ring-0
text-black focus-visible:ring-offset-0"
                        placeholder="Enter channel name"
                        {...field}
                      />
                    </FormControl>
                    <FormMessage />
                  </FormItem>
                )}
              />
              <FormField
                control={form.control}
                name="type"
                render={({ field }) => (
                  <FormItem>
                    <FormLabel>Channel Type</FormLabel>
                    <Select
                      disabled={isLoading}
                      onValueChange={field.onChange}
                      defaultValue={field.value}
                    >
                      <FormControl>
                        <SelectTrigger
                          className="bg-zinc-300/50 border-0 focus:ring-0 text-black
ring-offset-0 focus:ring-offset-0 capitalize outline-none"
                        >
                          <SelectValue placeholder="Select a channel type" />
                        </SelectTrigger>
                      </FormControl>
                      <SelectContent>
                        {Object.values(ChannelType).map((type) => (
                          <SelectItem
                            key={type}
                            value={type}
                            className="capitalize"
                          >
                            {type.toLowerCase()}
                          </SelectItem>
                        ))}
                      </SelectContent>
                    </Select>
                    <FormMessage />
                  </FormItem>
                )}
              />
            </div>
            <DialogFooter className="bg-gray-100 px-6 py-4">
              <Button variant="primary" disabled={isLoading}>
```

```
                Create
              </Button>
            </DialogFooter>
          </form>
        </Form>
      </DialogContent>
    </Dialog>
  )
}
```

#C:\Users\6truy\Downloads\FSD\components\modals\delete-channel-modal.tsx

```
"use client";

import qs from "query-string";
import axios from "axios";
import { useState } from "react";
import { useRouter } from "next/navigation";

import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogFooter,
  DialogHeader,
  DialogTitle,
} from "@/components/ui/dialog";
import { useModal } from "@/hooks/use-modal-store";
import { Button } from "@/components/ui/button";

export const DeleteChannelModal = () => {
  const { isOpen, onClose, type, data } = useModal();
  const router = useRouter();

  const isModalOpen = isOpen && type === "deleteChannel";
  const { server, channel } = data;

  const [isLoading, setIsLoading] = useState(false);

  const onClick = async () => {
    try {
      setIsLoading(true);
      const url = qs.stringifyUrl({
        url: `/api/channels/${channel?.id}`,
        query: {
          serverId: server?.id,
        }
      })
```

```
      await axios.delete(url);

      onClose();
      router.refresh();
      router.push(`/servers/${server?.id}`);
    } catch (error) {
      console.log(error);
    } finally {
      setIsLoading(false);
    }
  }

  return (
    <Dialog open={isModalOpen} onOpenChange={onClose}>
      <DialogContent className="bg-white text-black p-0 overflow-hidden">
        <DialogHeader className="pt-8 px-6">
          <DialogTitle className="text-2xl text-center font-bold">
            Delete Channel
          </DialogTitle>
          <DialogDescription className="text-center text-zinc-500">
            Are you sure you want to do this? <br />
             <span className="text-indigo-500 font-semibold">#{channel?.name}</span>
will be permanently deleted.
          </DialogDescription>
        </DialogHeader>
        <DialogFooter className="bg-gray-100 px-6 py-4">
          <div className="flex items-center justify-between w-full">
            <Button
              disabled={isLoading}
              onClick={onClose}
              variant="ghost"
            >
              Cancel
            </Button>
            <Button
              disabled={isLoading}
              variant="primary"
              onClick={onClick}
            >
              Confirm
            </Button>
          </div>
        </DialogFooter>
      </DialogContent>
    </Dialog>
  )
}
```

#C:\Users\6truy\Downloads\FSD\components\modals\delete-message-modal.tsx

```
"use client";

import qs from "query-string";
import axios from "axios";
import { useState } from "react";

import {
  Dialog,
  DialogContent,
  DialogDescription,
  DialogFooter,
  DialogHeader,
  DialogTitle,
} from "@/components/ui/dialog";
import { useModal } from "@/hooks/use-modal-store";
import { Button } from "@/components/ui/button";

export const DeleteMessageModal = () => {
  const { isOpen, onClose, type, data } = useModal();

  const isModalOpen = isOpen && type === "deleteMessage";
  const { apiUrl, query } = data;

  const [isLoading, setIsLoading] = useState(false);

  const onClick = async () => {
    try {
      setIsLoading(true);
      const url = qs.stringifyUrl({
        url: apiUrl || "",
        query,
      });

      await axios.delete(url);

      onClose();
    } catch (error) {
      console.log(error);
    } finally {
      setIsLoading(false);
    }
  }

  return (
    <Dialog open={isModalOpen} onOpenChange={onClose}>
```

```
      <DialogContent className="bg-white text-black p-0 overflow-hidden">
        <DialogHeader className="pt-8 px-6">
          <DialogTitle className="text-2xl text-center font-bold">
            Delete Message
          </DialogTitle>
          <DialogDescription className="text-center text-zinc-500">
            Are you sure you want to do this? <br />
            The message will be permanently deleted.
          </DialogDescription>
        </DialogHeader>
        <DialogFooter className="bg-gray-100 px-6 py-4">
          <div className="flex items-center justify-between w-full">
            <Button
              disabled={isLoading}
              onClick={onClose}
              variant="ghost"
            >
              Cancel
            </Button>
            <Button
              disabled={isLoading}
              variant="primary"
              onClick={onClick}
            >
              Confirm
            </Button>
          </div>
        </DialogFooter>
      </DialogContent>
    </Dialog>
  )
}
```

#C:\Users\6truy\Downloads\FSD\components\server\server-channel.tsx

```
"use client";

import {
  Channel,
  ChannelType,
  MemberRole,
  Server
} from "@prisma/client";
import { Edit, Hash, Lock, Mic, Trash, Video } from "lucide-react";
import { useParams, useRouter } from "next/navigation";

import { cn } from "@/lib/utils";
import { ActionTooltip } from "@/components/action-tooltip";
```

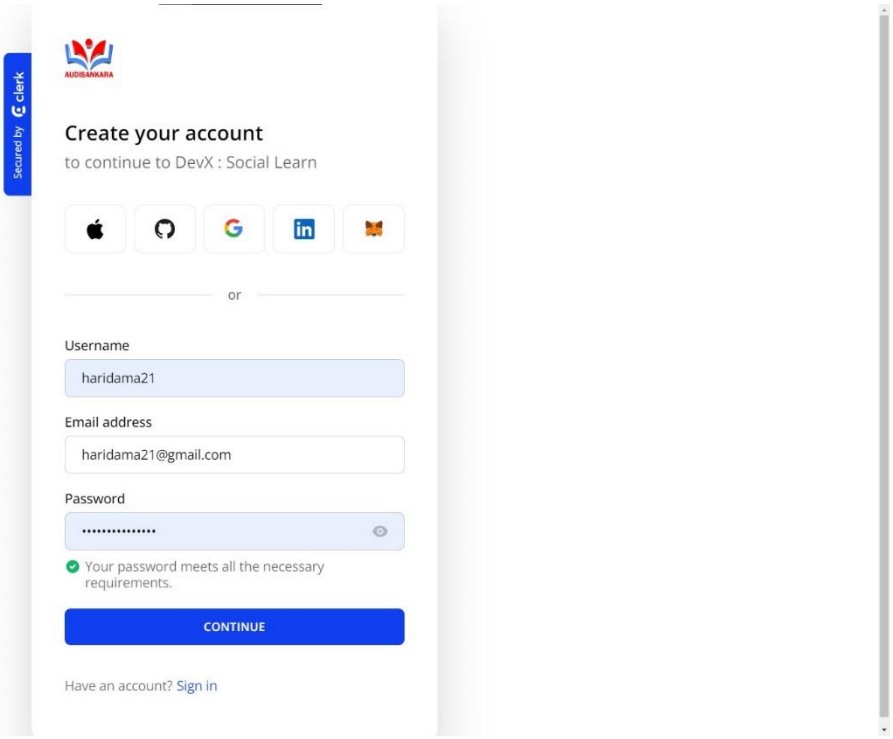
```
import { ModalType, useModal } from "@/hooks/use-modal-store";

interface ServerChannelProps {
  channel: Channel;
  server: Server;
  role?: MemberRole;
}

const iconMap = {
  [ChannelType.TEXT]: Hash,
  [ChannelType.AUDIO]: Mic,
  [ChannelType.VIDEO]: Video,
}

export const ServerChannel = ({
  channel,
  server,
  role
}: ServerChannelProps) => {
  const { onOpen } = useModal();
  const params = useParams();
  const router = useRouter();

  const Icon = iconMap[channel.type];

  const onClick = () => {
    router.push(`/servers/${params?.serverId}/channels/${channel.id}`)
  }

  const onAction = (e: React.MouseEvent, action: ModalType) => {
    e.stopPropagation();
    onOpen(action, { channel, server });
  }

  return (
    <button
      onClick={onClick}
      className={cn(
        "group px-2 py-2 rounded-md flex items-center gap-x-2 w-full hover:bg-zinc-
700/10 dark:hover:bg-zinc-700/50 transition mb-1",
        params?.channelId === channel.id && "bg-zinc-700/20 dark:bg-zinc-700"
      )}
    >
      <Icon className="flex-shrink-0 w-5 h-5 text-zinc-500 dark:text-zinc-400" />
      <p className={cn(
        "line-clamp-1 font-semibold text-sm text-zinc-500 group-hover:text-zinc-600
dark:text-zinc-400 dark:group-hover:text-zinc-300 transition",
```

```
            params?.channelId  ===  channel.id  &&  "text-primary  dark:text-zinc-200
dark:group-hover:text-white"
      )}>
        {channel.name}
      </p>
      {channel.name !== "general" && role !== MemberRole.GUEST && (
        <div className="ml-auto flex items-center gap-x-2">
          <ActionTooltip label="Edit">
            <Edit
              onClick={(e) => onAction(e, "editChannel")}
               className="hidden group-hover:block w-4 h-4 text-zinc-500 hover:text-
zinc-600 dark:text-zinc-400 dark:hover:text-zinc-300 transition"
            />
          </ActionTooltip>
          <ActionTooltip label="Delete">
            <Trash
              onClick={(e) => onAction(e, "deleteChannel")}
               className="hidden group-hover:block w-4 h-4 text-zinc-500 hover:text-
zinc-600 dark:text-zinc-400 dark:hover:text-zinc-300 transition"
            />
          </ActionTooltip>
        </div>
      )}
      {channel.name === "general" && (
        <Lock
          className="ml-auto w-4 h-4 text-zinc-500 dark:text-zinc-400"
        />
      )}
    </button>
  )
```

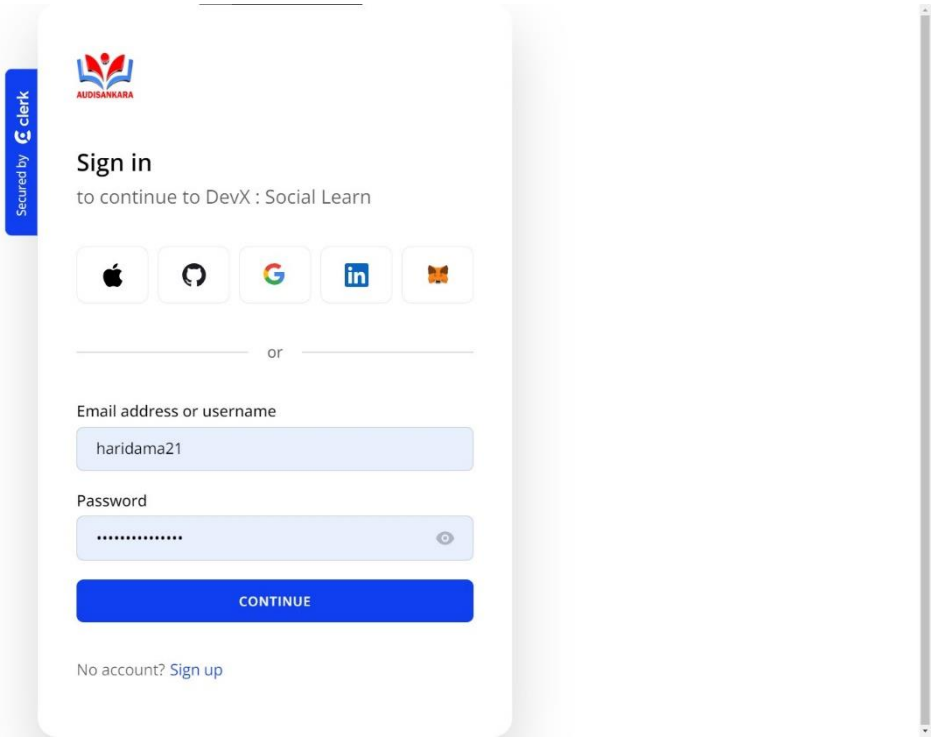
CHAPTER 10

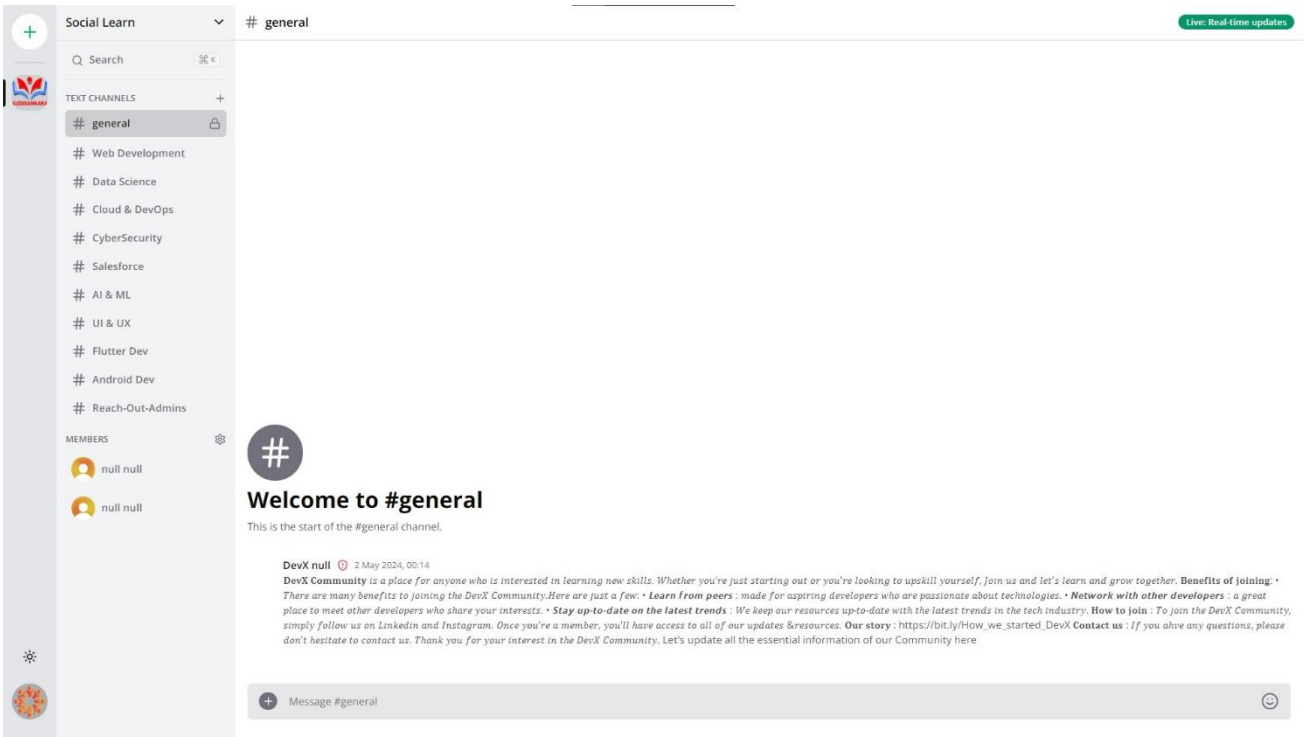
SCREENSHOTS

10.1 User Registeration

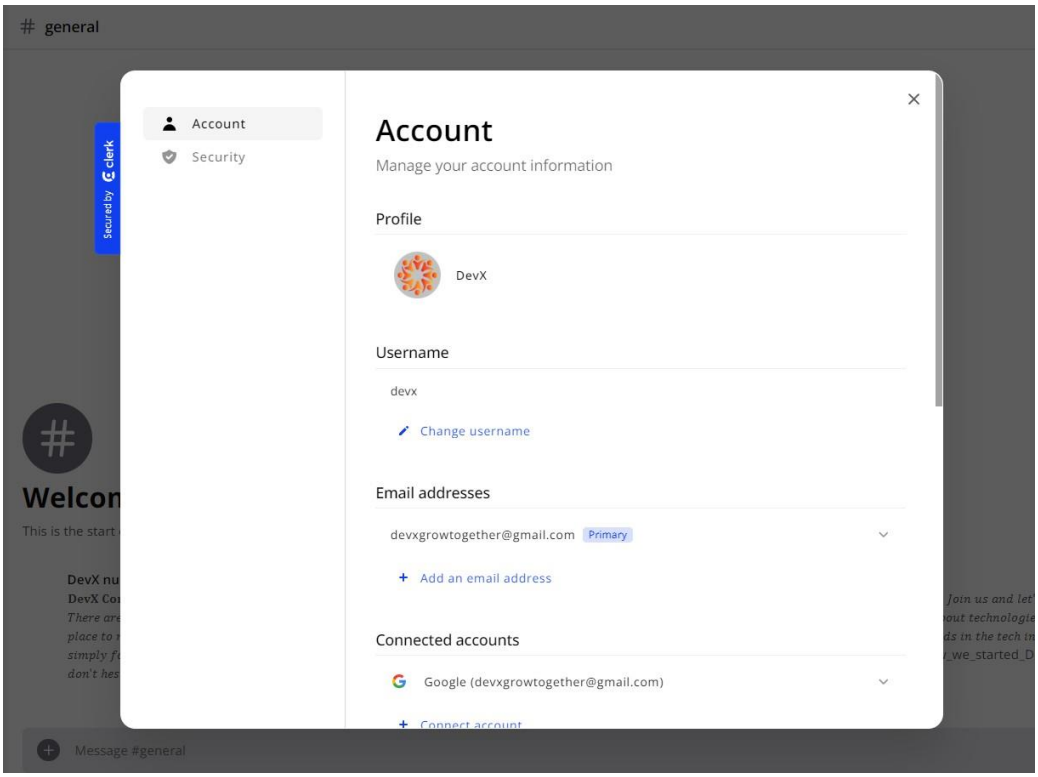


10.2 User Login



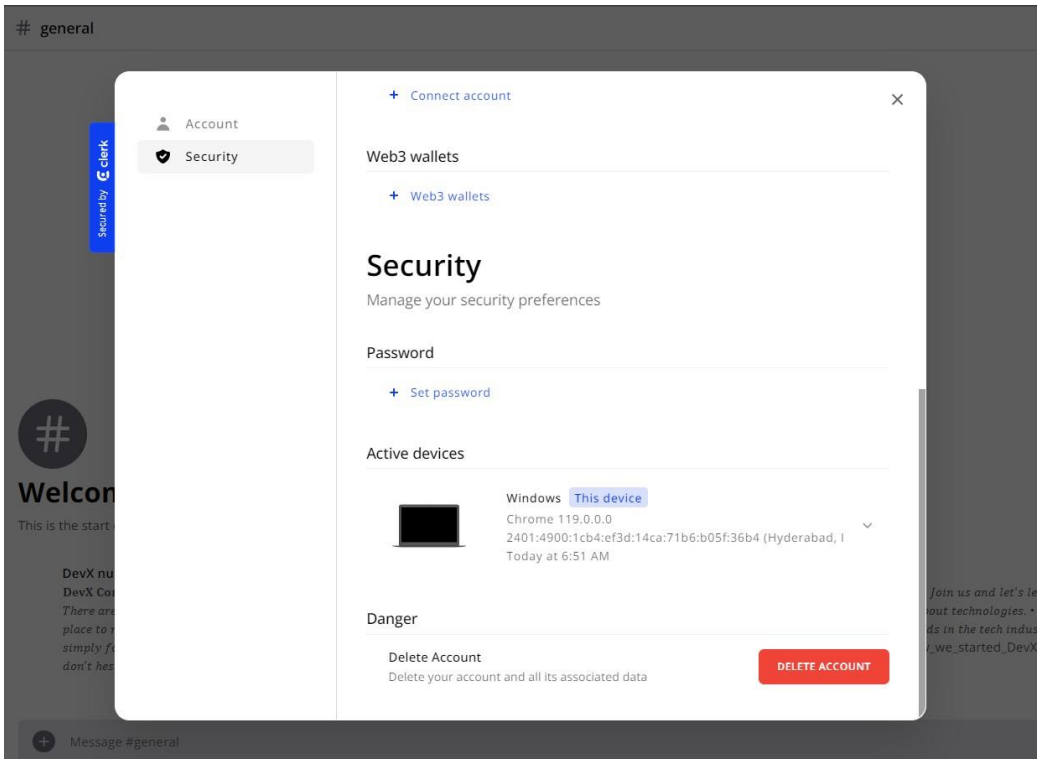


10.3 Social Learn : Landing Page

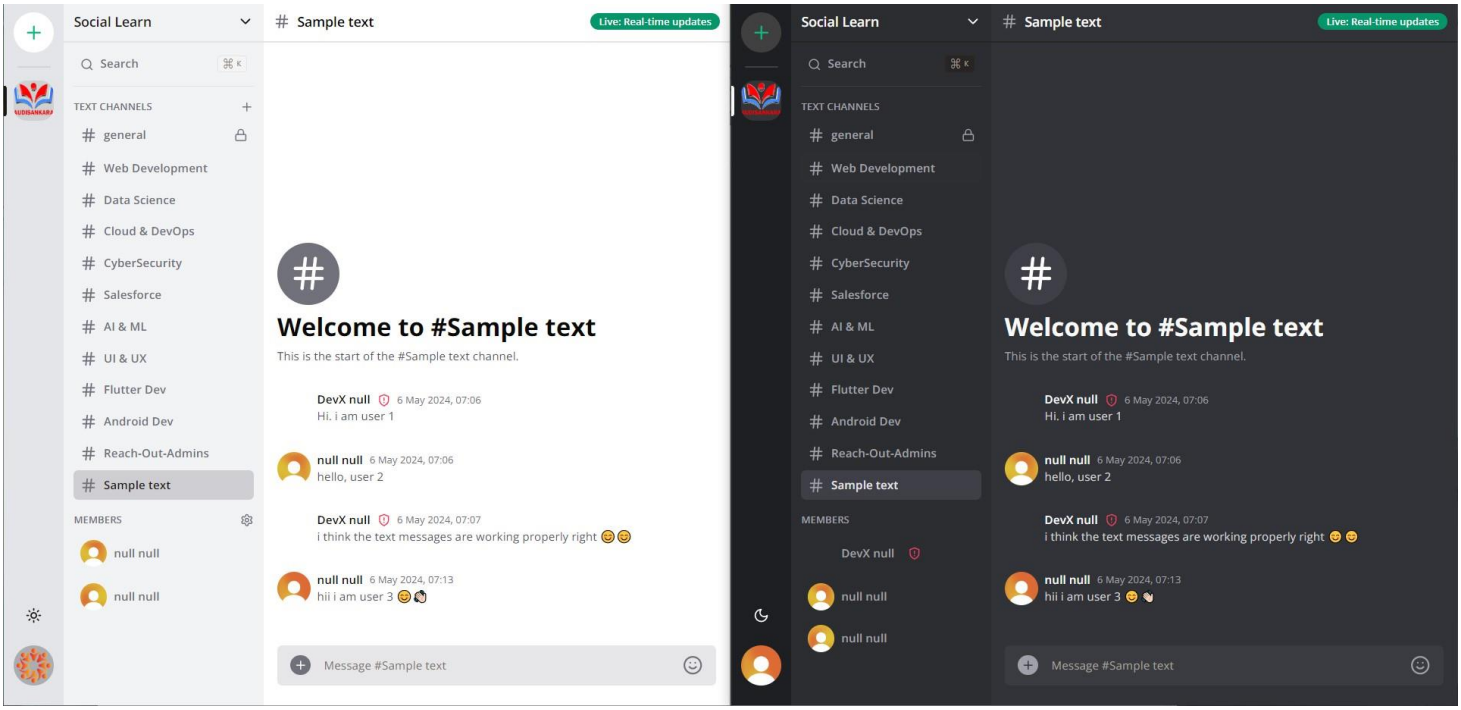


s

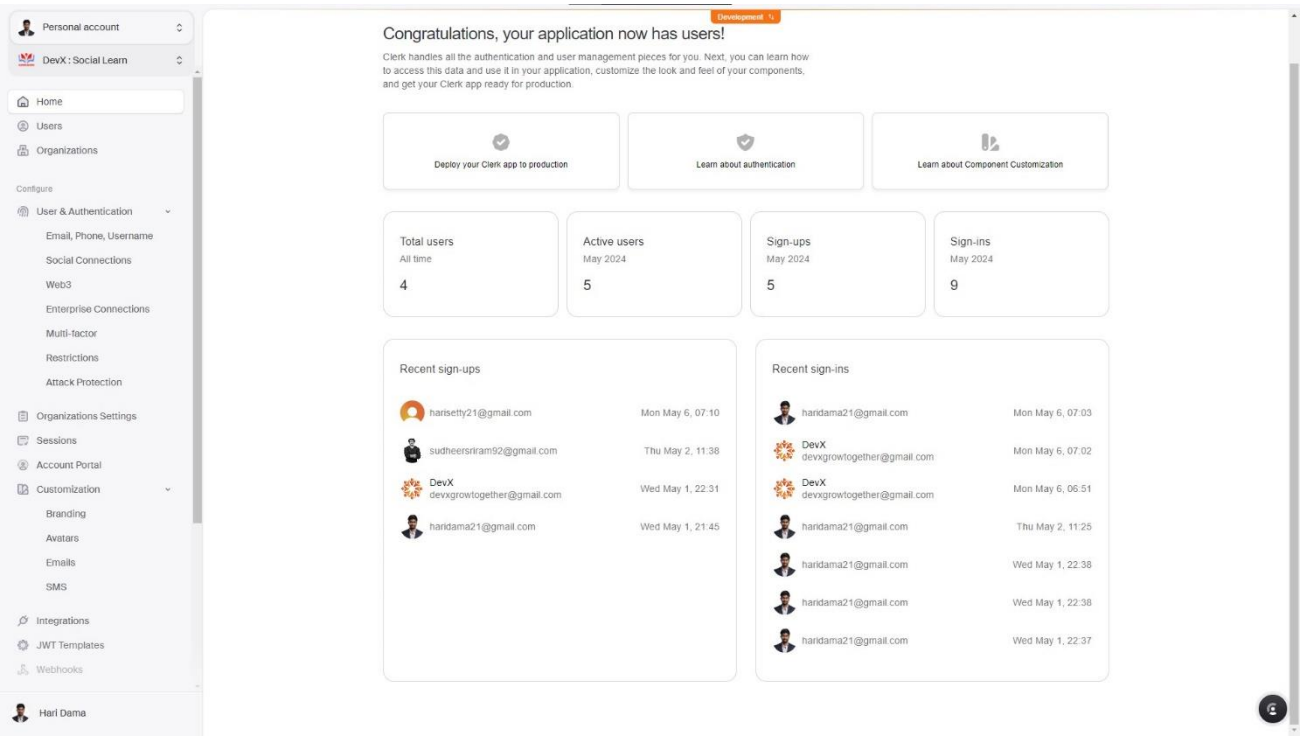
10.4 User Account Customization



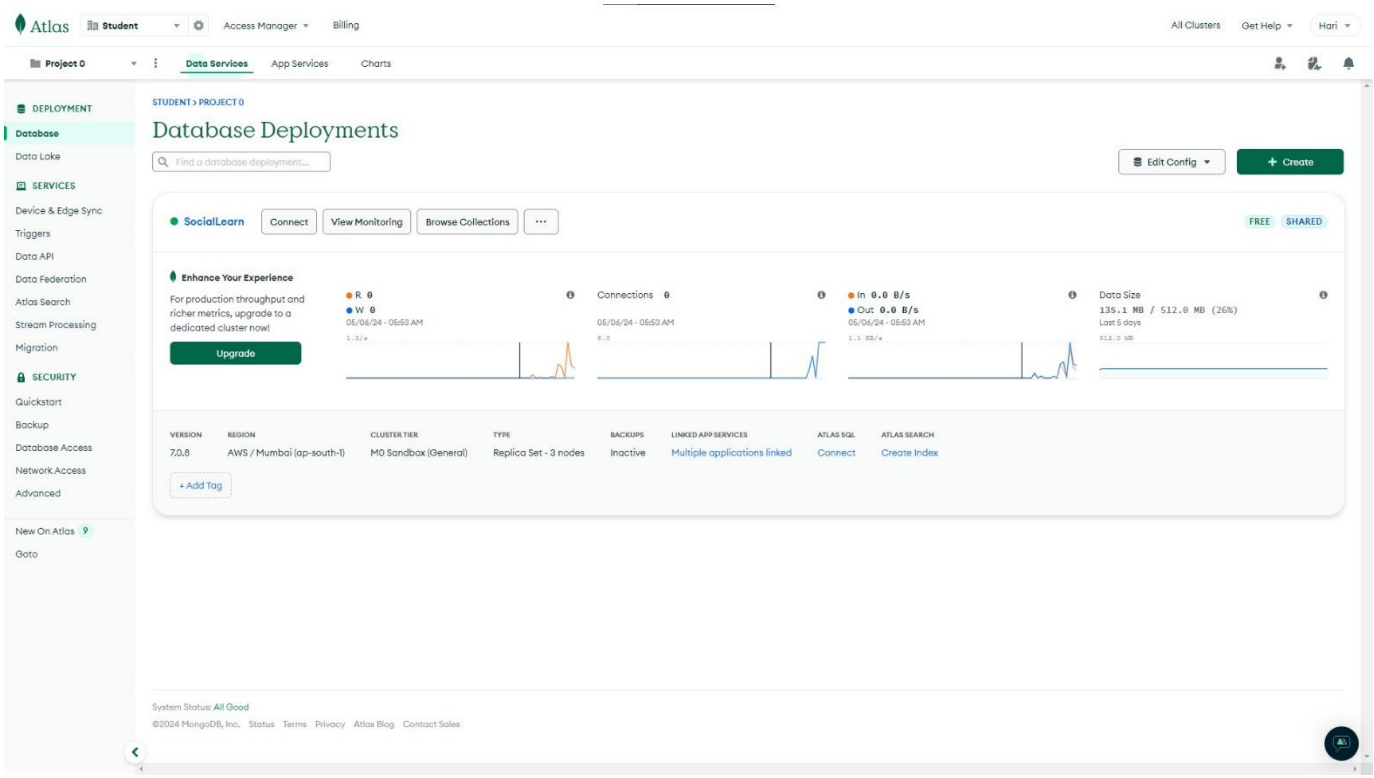
10.5 User Account Security



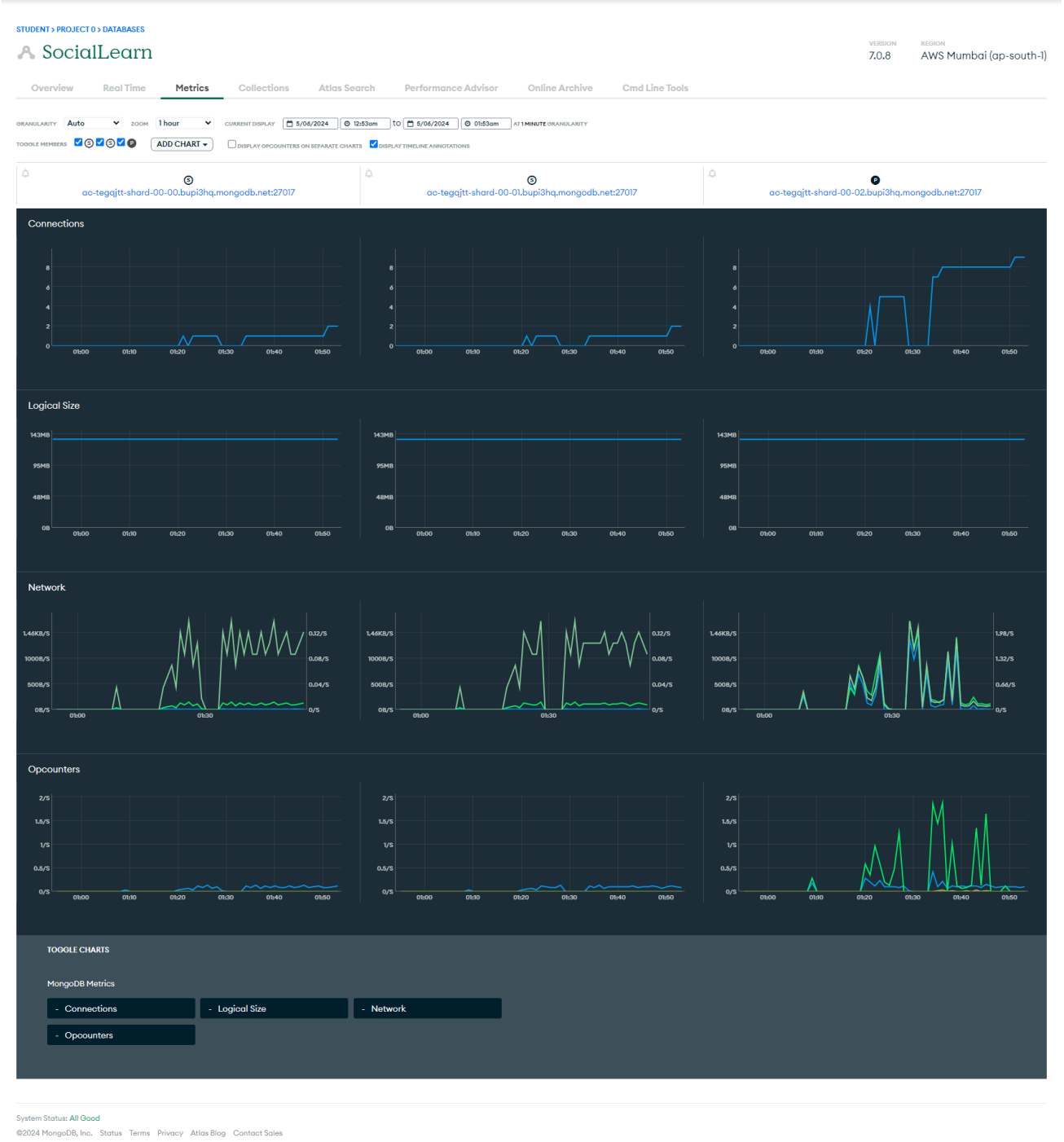
10.6 Discussion Sample Messages



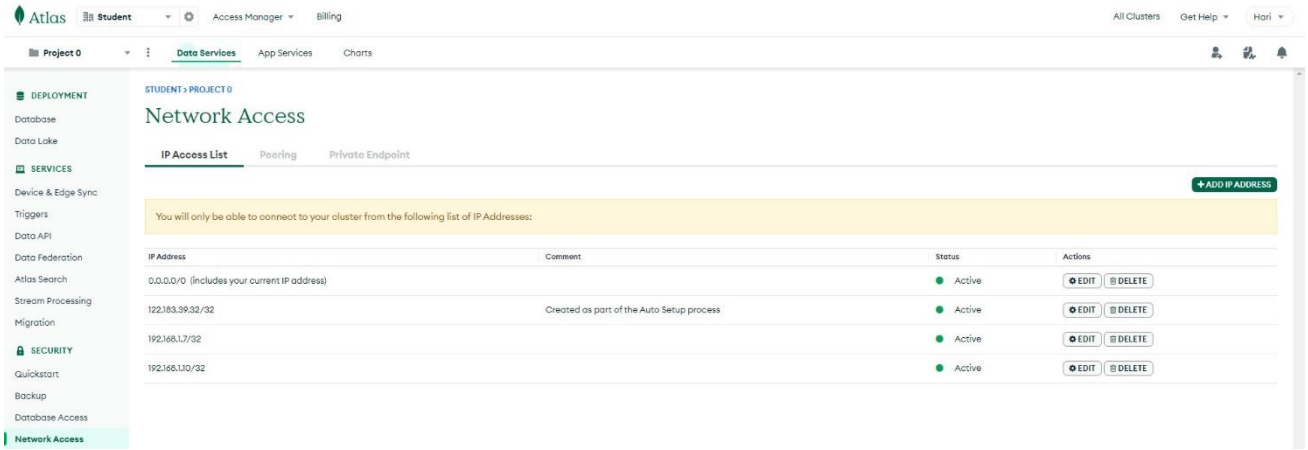
10.7 Clerk API : User Accounts Logs



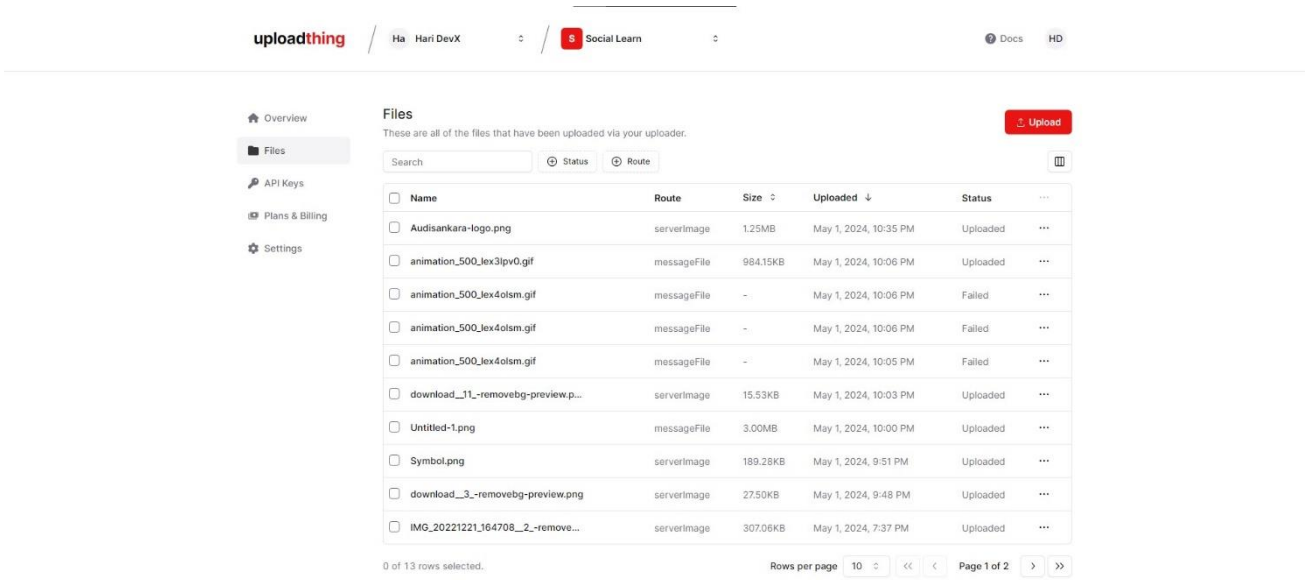
10.8 MongoDB : database Deployment



10.9 MongoDB : Server Usage Metrics in Cloud



10.10 MongoDB : Database Network Access



10.11 Uploadthing API : File Uploads

CHAPTER 11

CONCLUSION AND FUTURE SCOPE

Social Learn is a project dedicated to fostering a collaborative learning environment for students. It leverages the power of technology to bridge the gap between individual study and a vibrant learning community. Built upon the MERN stack (MongoDB, Express.js, React.js, and Node.js), Social Learn offers a flexible and efficient platform for knowledge sharing and interaction.

This project goes beyond traditional models where students passively receive information. Social Learn empowers them to actively contribute to a shared pool of knowledge. By integrating user management (Clerk API), content management (Uploadthing API), and a robust database (MongoDB Atlas), Social Learn establishes a foundation for a dynamic learning ecosystem.

Social Learn aims to move beyond rote memorization in isolated environments. It introduces a more engaging approach. Students can upload a variety of learning materials, fostering a sense of ownership and encouraging collaboration. The ability to create posts and comments ignites discussion and transforms learning into a two-way street where ideas can be exchanged and refined.

Social Learn emphasizes building connections alongside knowledge sharing. Features like user profiles and (potentially) the ability to follow classmates cultivate a sense of community. Students can learn from and provide support to one another, creating a network of peers united by their pursuit of knowledge.

Social Learn is more than just a technical achievement; it's an exploration of social learning dynamics. It challenges traditional learning paradigms and paves the way for a more engaging and collaborative future. It aspires to be not just a platform, but a springboard for a more connected generation of students, one that thrives on shared knowledge and a collaborative learning experience. The potential for growth and innovation is vast, offering the opportunity to revolutionize how students learn and grow together.

FUTURE SCOPE :

The "Social Learn: Social Learninh network For Students " project lays a strong foundation for future enhancements and expansions to further empower individuals in upskilling and community building. Some potential avenues for future development and improvement include the following things:

- **AI-powered Learning:** Integration of artificial intelligence could enable features like personalized learning paths, recommendation systems, and automated feedback for a more tailored learning experience.
- **Gamification:** Incorporating gamification elements like points, badges, and leaderboards could further incentivize student participation and engagement.
- **Social Learning Analytics:** Analyzing user interactions and content engagement can provide valuable insights into student learning patterns, allowing educators to tailor their teaching strategies and resources.
- **Expanded Social Features:** Building upon user profiles and following systems, the platform could integrate features for group projects, discussion forums, and real-time collaboration tools, further amplifying the collaborative learning aspect.
- **Granular Search:** Implement search functionalities that allow students to filter by content type (articles, videos, notes), upload date, specific keywords, or even user profiles (for finding content created by classmates).
- **Topic-Based Exploration:** Introduce the ability to browse content organized by topics or learning areas. This can involve user-generated tags or a curated taxonomy system.
- **Adaptive Learning Challenges:** Introduce interactive quizzes or challenges that adapt to the student's learning level, providing personalized feedback and adjusting difficulty for optimal learning progress.
- **Points and Badges:** Award points for completing learning activities, uploading content, or participating in discussions. Badges can be earned for achieving milestones or demonstrating mastery in specific areas.

CHAPTER 13

BIBLIOGRAPHY

1. Tufte, Edward R. "The Visual Display of Quantitative Information." Graphics Press, 1986.
2. Wickham, Hadley. "ggplot2: Elegant Graphics for Data Analysis." Springer, 2016.
3. Tukey, John W. "Exploratory Data Analysis." Addison-Wesley, 1977.
4. Few, Stephen. "Show Me the Numbers: Designing Tables and Graphs to Enlighten." Analytics Press, 2012.
5. Murray, Scott. "Interactive Data Visualization for the Web." O'Reilly Media, 2013.
6. Kandel, Sean, et al. "Wrangler: Interactive visual specification of data transformation scripts." ACM Human Factors in Computing Systems (CHI), 2011.
7. Heer, Jeffrey, and Ben Shneiderman. "Interactive dynamics for visual analysis." ACM Human Factors in Computing Systems (CHI), 2012.
8. MongoDB**:** Include the official documentation or a reputable online resource: "https://www.mongodb.com/docs/"
9. Express.js**:** Include the official documentation: "https://expressjs.com/"
10. React.js: Include the official documentation: "https://legacy.reactjs.org/"
11. Node.js: Include the official documentation: "https://nodejs.org/en"
12. Clerk API: If you have specific documentation for the Clerk API you're using, include it here. Otherwise, a general link to their documentation might suffice.https://clerk.com/
13. Uploadthing API: Similar to Clerk API, include their documentation if available. Otherwise, a general link might work."https://uploadthing.com/"
14. RESTful APIs: If your project heavily relies on RESTful principles, consider including a reference on the topic: "https://restfulapi.net/"