REAL-TIME CHAT APPLICATION USING THE MERN STACK

BY

OLUBAYO WISDOM

PSC1908918

DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF PHYSICAL SCIENCES,

UNIVERSITY OF BENIN,

BENIN CITY,

EDO STATE, NIGERIA.

MAY 2024

REAL-TIME CHAT APPLICATION USING THE MERN STACK

BY

OLUBAYO WISDOM

PSC1908918

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF PHYSICAL SCIENCES, UNIVERSITY OF BENIN, BENIN CITY

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF A BACHELOR OF SCIENCE (B.Sc.) DEGREE IN COMPUTER SCIENCE

JUNE 2024

CERTIFICATION

This is to certify that this project work was carried out by **OLUBAYO WISDOM** with Matriculation Number **PSC1908918** under my supervision. It is adequate and satisfactory, both in scope and content, for the award of Bachelor of Science (B.sc) Degree in Computer Science of the University of Benin

_____ _____

PROF A.A. IMAVIAN **DATE**

Project Supervisor

APPROVAL

This project work is hereby approved in partial fulfilment of the requirements for the award of Bachelor of Science (B.Sc.) Degree in Computer Science from the University of Benin.

_____ ._____

Prof. Godspower O. Ekuobase, PhD. **DATE**

Head of Department

DEDICATION

This project is dedicated to God Almighty for giving me the strength and wisdom to see it through to completion, and even throughout my stay in the University of Benin (UNIBEN). It is also dedicated to my parents; Mr and Mrs Olubayo and my Guardians Mr and Mrs Ekundayo; for their love, support and guidance throughout my academic journey.

ACKNOWLEDGEMENT

My utmost acknowledgement goes to God Almighty for giving me the strength, wisdom and direction throughout my academic journey. I would like to express my gratitude to my project supervisor who is also the Head of the Department Of Computer Science, Prof. A.A. Imavian for his consistent guidance towards ensuring the successful completion of this project.

I would also like to specially thank my project coordinator Dr. (Mrs.) A.R. Usiobaifo, and other lecturers in the Department of Computer Science who I have been opportune to cross paths with, and have impacted me immensely these past few years: Prof. G.O. Ekuobase, Dr. F.O. Oliha, Prof. K.C. Ukaoha, Prof. A.A. Imiavan, Prof. (Mrs.) F. Egbokhare, Prof. (Mrs.) V.V.N. Akwukwuma, Prof. F.I. Amadin, Prof. (Mrs.) S. Konyeha, Prof. (Mrs.) V.I. Osubor, Dr. (Mrs.) Aziken, Dr. F.O. Chete, Dr. (Mrs) R.O. Osaseri, Dr. J.C. Obi, Mr. P. E.B. Imiefoh, Mr. I.E. Obasohan, Mr. S.O.P. Oliomogbe, Mr. K.O. Otokiti, Mr. I.E. obayagbonna, Mrs. R.I. Izevbizua, Mr. E.C. Igodan, Miss L.O.Usiosefe, Mr J. Okhuoya, Prof. F.A.U. Imouokhome, Mrs. J.I. Adun, Dr. E. Nweli and Mr. D.N. Idehen.

Finally, I also want to appreciate those who contributed to the success of this project: Chaseideho Osegale, Olumese Obehighie and Enilama Godwin. I would also like to thank my family and friends for their support, words of encouragement, and consistent guidance throughout this project.

50
51
52
52
53
54
55
55
55
56
59
60
61
61
61
61
61
61
62
62
62
64
64
64
65
66
66
71
71
71
71
72
75
75

ABSTRACT

The MERN Chat Application for College Students is a contemporary web-based communication tool created to facilitate seamless interaction and collaboration among college students. In today's digital era, effective communication plays a crucial role in improving learning outcomes and building a sense of community among students.

Current chat applications often lack the specific features and security measures required for the unique needs of college students. Taking into consideration the constraints and the scope of the project, which is the University of Benin, Department of Computer Science, the method proposed by this project is a MERN Chat Application which aim to tackle these challenges by offering a secure, user-friendly, and feature-rich platform tailored to the needs of college students. By utilizing the MERN stack (MongoDB, Express.js, React.js, Node.js), the application provides a strong and scalable solution for real-time messaging, group discussions, and academic collaboration.

Key features of the MERN Chat Application include secure user authentication, dedicated group chat spaces, real-time messaging capabilities, and compatibility across various devices and web browsers. The application places a high priority on user privacy and data security, ensuring that student information remains confidential and always protected.

Through a combination of cutting-edge technology, user-friendly design, and comprehensive features, the MERN Chat Application aims to transform the way college students communicate and collaborate.

By offering a centralized platform for academic discussions, resource sharing, and group projects, the application aims to improve learning outcomes, encourage community engagement, and promote collaborative learning among college students.

CHAPTER ONE

INTRODUCTION

1.1 INTRODUCTION

The advancement in technologies and the fast-paced globalization has led to students in higher institutions either partially or fully in an online environment (Lapointe and Reisetter, 2008).

The recent growth in computer networks and more specifically, the World Wide Web (WWW) has given rise to communication, socialization and interaction between individuals via the internet. Sylvia (2007) explains that the internet is been used for several purposes this day ranging from education, business and so on, one major area that focuses on individual interaction and communication is online chat.

Online Chat is any form of communication over the internet that offers real-time transmission of text messages and files between participants. Messages are delivered and displayed almost instantly, mimicking a spoken conversation. This form of communication can occur in various formats, like one-on-one conversations, group chats, chat rooms, or embedded features within websites or applications.

Today, the world is brimming with technological advancements from cutting-edge innovations to familiar tools like the internet that have become seamlessly woven into our daily lives. The Android platform exemplifies this integration, empowering developers with freely available tools and user-friendly hardware platform, leading to a plethora of applications aimed at various needs. One prominent example lies in the social sphere, where web applications built on Android enable families and friends to connect on a deeper level through group chats, shared messages, and even location tracking. This highlights the dynamics interplay between technological progress and its impact on human relationships, showcasing hoe platforms like Android facilitate closer connections and foster stronger bond amidst our ever-evolving digital landscape.

1.2 BACKGROUND OF THE STUDY

The human desire to connect, to share thoughts and experiences with others, has always been a driving force. From the evocative cave paintings of our ancestors to

the intricate smoke signals used by indigenous cultures, we've employed various tools to bridge the physical gap and forge relationships. The 19th century witnessed a revolution in communication with the invention of the telegraph, telephone, and television. These technologies, for the first time, allowed for near-instantaneous communication across vast distances, forever altering the way we interacted. The telephone, in particular, experienced explosive growth, evolving from cumbersome fixed lines to the ubiquitous mobile devices we carry today, capable of not just voice calls but also seamless data transfer, blurring the lines between communication and information access.

But the story doesn't end there. The 20th century ushered in a new era of convergence, where computer and telecommunication technologies began to merge, driven by a shared goal: seamless human interaction, regardless of physical barriers. This convergence fueled efforts to consolidate communication tools into a single, versatile device, as envisioned by LaPointe and Reisetter (2008). Enter chatting, a new paradigm of communication using technology to bring people and ideas together, irrespective of geographical boundaries. While technology existed earlier, its widespread adoption is a recent phenomenon.

Online chat, defined by its real-time transmission of text messages via the internet, offers a unique and engaging experience. Unlike traditional email or forums, where messages might wait hours or even days for a response, online chat fosters a sense

of immediacy and dynamism. Short, quick messages create a conversational atmosphere, mirroring the back and forth of spoken communication. This real-time nature extends beyond simple texts, with platforms like web conferencing incorporating voice and video capabilities, further blurring the lines between the virtual and physical world.

But online chat isn't limited to one-on-one interactions. Platforms like instant messengers, Internet Relay Chat (IRC), and even virtual worlds like Multi-User Dimensions (MUDs) facilitate group conversations, enabling multiple users to engage in real-time discussions. Moreover, these platforms offer varying levels of anonymity, catering to both the desire for direct, addressed communication and freedom of more casual, anonymous interactions within large communities. The very name "chat" underscores this informality, evoking the casual exchange of ideas and feelings.

Web-based applications further expand the realm of online chat, allowing direct communication between users on various platforms. While often anonymous, these applications foster interaction and community building. However, web conferencing stands apart as a more specific service, often offered as a subscription model and hosted on dedicated servers, facilitating organized meetings and presentations with advanced features sharing and collaborative tools.

The evolution of communication from primitive smoke signals to the dynamic world of online chat reflects a relentless human desire for connection. Today, these technologies not only bridge physical distances but also create new forms of social interaction, blurring the lines between online and offline experiences. As we continue to innovate and explore, future communication promises to be even more immersive and interconnected, further enriching the way we connect and share our humanity with the world.

1.3 MOTIVATION

Witnessing the limitation of asynchronous communication in online learning delays, stalled discussions, and passive engagement ignited my passion to create a solution. This project proposes a real-time chat application integrated within learning management systems, not just for convenience, but to fundamentally transform the learning experience.

Imagine instant doubt clarification, lively discussions fueled by diverse perspectives, and active participation fostered by immediate feedback. This application fosters stronger relationships between students and educators, leading to a personalized and dynamic learning community. It aligns with the institution's goals of boosting engagement, promoting collaboration, and making learning accessible and inclusive for all. By unlocking real-time communication and collaboration, this project sparks synergy in learning empowering students,

educators, and the institution to thrive in a more connected and effective learning landscape.

1.4 PROBLEM DEFINITION

The reliance on asynchronous communication tools like email and discussion boards in traditional learning environments presents a significant barrier to effective student-educator interaction. These tools inherently introduce delays, hindering the dynamic flow of ideas and immediate clarification of doubts that are crucial for deep understanding and active learning.

The shortcomings of asynchronous communication are particularly detrimental in fostering group discussions, which require real-time engagement and collaborative brainstorming to spark insightful exchange and collective knowledge creation.

1.5 AIMS AND OBJECTIVES

This project proposes a solution to bridge this gap by developing a chat application integrated with enabling real-time communication between students and educators, enhancing the learning experience. This application will empower students and educators to:

- Clarify doubts instantly: Immediate question resolution fosters deeper engagement and prevents confusion from snowballing.

- Engage in dynamic discussions: Real-time interactions facilitate lively debates, spontaneous idea generation, and collaborative problem-solving.
- Promote active learning: Immediate feedback and interactive exchange create a more engaging learning environment, encouraging active participation and knowledge construction.

1.6 SIGNIFICANCE OF THE STUDY

This project holds tremendous significance for addressing a critical need in learning and fostering a more effective, engaging, and inclusive learning experience for students of all backgrounds. By promoting real-time communication and collaboration, this project has the potential to redefine the student-educator relationship and empower learners to thrive in the rapidly evolving educational landscape.

1.7 SCOPE OF THE STUDY

This study addresses the significant limitations of asynchronous communication tools in online learning environments, hindering student-educator interaction and ultimately impacting learning outcomes. It proposes the development and evaluation of a real-time chat application. The study's scope focuses on designing

and developing the application, and evaluating its impact on student understanding, engagement, communication skills, and relationship building with educators. While focusing on a specific institutional context, the study acknowledges potential limitations and lays the groundwork for future development and research into the application's long-term impact and effectiveness across diverse settings and student populations. This research holds significant potential to improve online learning by fostering real-time communication, promoting active participation, and facilitating a more dynamic and enriching learning

experience for all.

1.8 LIMITATION OF THE STUDY

While the study investigates the immediate impact of a real-tine chat application within the target audience which includes students in the institution, it acknowledges limitations such as specific institutional focus, potentially limiting generalizability, the need for further research on long-term effects and impact on diverse populations.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

A literature review is a fundamental component of any research project, serving several critical purposes. Here's an in-depth exploration of why a literature review is necessary and how it aids in understanding the existing landscape and guiding project development.

Understanding the Existing Landscape

- **Identifying Gaps in Knowledge:** A thorough literature review helps in identifying what has already been studied and what gaps remain in the field.

  This is crucial for ensuring that your project addresses a novel area or offers a new perspective on a well-researched topic. According to Machi and McEvoy (2016), a literature review is essential in highlighting the areas where knowledge is lacking or where further research is needed.

- **Contextualizing Research:** By examining the existing body of work, a literature review places your research within the broader academic conversation. This context helps to establish the relevance and significance of your study. Fink (2020) notes that understanding how your project fits within the larger framework of existing research can provide a foundation for your work, making it more robust and credible.

- **Avoiding Duplication:** Reviewing existing literature ensures that your work does not duplicate what has already been done, which saves time and resources and adds value to the field by contributing original insights or findings. Hart (2018) emphasizes that a literature review is essential for

ensuring that the research question is unique and has not been answered previously in the same manner.

Guiding Project Development

- **Refining the Research Question:** A well-conducted literature review can help refine your research question by revealing what aspects of a topic have already been explored and what questions still need answering. This refinement can lead to more precise and impactful research questions. As discussed by Creswell (2014), understanding the scope and depth of existing studies can guide the formulation of more targeted and meaningful research questions.

- **Methodological Insights:** Reviewing how other researchers have approached similar questions can provide valuable insights into effective methodologies and approaches. This can guide the design of your research, ensuring that you employ the best possible methods. According to Ridley (2012), a literature review can help identify the most appropriate research methods, data collection techniques, and analytical strategies used in the field.

- **Building a Theoretical Framework:** A literature review helps in constructing a solid theoretical framework for your project. By understanding the theories and models that have been applied to similar studies, you can develop a robust theoretical foundation for your own research. Merriam and Tisdell (2015) highlight the importance of using a literature review to inform the theoretical underpinnings of a study, ensuring that it is grounded in existing knowledge.

Enhancing Scholarly Contribution

- **Supporting Arguments:** A literature review provides an evidence base that supports the arguments and hypotheses of your research. It demonstrates that your work is grounded in a comprehensive understanding of the topic. According to Booth, Sutton, and Papaioannou (2016), a literature review is critical in providing the supporting context that strengthens the arguments made in the research.

- **Highlighting Research Impact:** By situating your research within the existing literature, you can more effectively demonstrate its impact and contribution to the field. This is important for justifying the significance of your study to peers, funding bodies, and other stakeholders. Cooper (2010) argues that a literature review is essential for showcasing the potential impact of new research by comparing it to existing studies and highlighting its unique contributions

2.2 HISTORICAL BACKGROUND OF CHAT APPLICATION

The history of online chat is a testament to the symbiotic relationship between technological advancements and human communication. From the Compatible Time-Sharing System (CTSS) developed at MIT in the early 1960s to the modern-day real-time chat applications built using the MERN stack, online chat has come a long way.

The emergence of online chat platforms like AOL Instant Messenger (AIM) and Yahoo! Messenger in the late 1990s revolutionized the way people communicated with each other. These platforms introduced features like user profiles, buddy lists, and chat rooms, which quickly gained widespread popularity and became emblematic of an era defined by the rapid proliferation of online chat culture.

Since then, online chat has evolved to become increasingly versatile and feature rich. It now serves various purposes, from casual conversations among friends to critical business communications and customer support interactions.

2.2.1 Early Chat Systems

Internet Relay Chat (IRC)

Internet Relay Chat (IRC) is one of the earliest forms of online chat systems, developed in 1988 by Jarkko Oikarinen. IRC allowed users to communicate in real-time via text-based messaging, creating channels for group discussions or private messaging between individuals.

Features of IRC:

- **Channels:** Users could join channels dedicated to specific topics or interests, where multiple users could participate in discussions.
- **Commands:** IRC introduced various commands for actions such as joining channels, sending private messages, and managing user privileges.
- **Bots:** IRC bots were commonly used for automated tasks, moderation, and providing information within channels.
- **Open Protocol:** IRC was based on an open protocol, allowing users to develop their own clients and servers, leading to a diverse ecosystem of IRC networks and software implementations.

IRC quickly gained popularity among early internet users and became a prominent platform for online communities, ranging from hobbyist groups to technical support channels.

AOL Instant Messenger (AIM)

AOL Instant Messenger (AIM) was launched in 1997 by America Online (AOL) and quickly became one of the most popular instant messaging platforms of its time. AIM allowed users to exchange messages in real-time, providing a user-friendly interface and a range of features that appealed to a broad audience.

Features of AIM:

- **Buddy Lists:** Users could create lists of friends and contacts, indicating who was online and available for chat.
- **File Transfer:** AIM supported file transfers, allowing users to share documents, images, and other files directly within conversations.
- **Customization:** Users could personalize their profiles with custom away messages, avatars, and user profiles.

- **Integration with AOL Services:** AIM seamlessly integrated with other AOL services, including email and news, providing a unified online experience for users.

AIM played a significant role in popularizing instant messaging among mainstream internet users and became an iconic symbol of the internet culture of the late 1990s and early 2000s. However, with the rise of social media and more advanced messaging platforms, AIM eventually declined in popularity and was discontinued in 2017.

2.2.2 The MERN Stack for Real-Time Chat Applications

With the development of real-time chat applications using modern technology stacks like MERN (MongoDB, Express.js, React, Node.js), online chat is poised to continue evolving and transforming the way people communicate with each other. The MERN stack is a powerful and flexible technology stack that is well-suited to building real-time chat applications. It consists of four technologies: MongoDB, Express.js, React, and Node.js.

These technologies work together seamlessly to offer a robust and scalable platform for building real-time chat applications that can handle large volumes of traffic and provide a seamless user experience.

2.2.3 Opportunites and Challenges

Building a real-time chat application using the MERN stack presents many opportunities and challenges. On the one hand, the MERN stack is a highly flexible and customizable technology stack that can be tailored to meet the specific needs of any chat application.

On the other hand, building a real-time chat application requires expertise in both front-end and back-end development, as well as a deep understanding of real-time communication protocols and user behavior.

Despite these challenges, building a real-time chat application using the MERN stack can be tremendously rewarding. It provides a unique opportunity to leverage the power of modern technology to transform the way people communicate with

each other. From casual conversations among friends to critical business communications and customer support interactions, the possibilities are endless. In conclusion, the history of online chat is a testament to the symbiotic relationship between technological advancements and human communication.

Building a real-time chat application using the MERN stack presents new opportunities and challenges, but it also offers a unique opportunity to leverage the power of modern technology to transform the way people communicate with each other.

2.3 EVOLUTION OF ONLINE CHAT

The evolution of online chat services has been a journey marked by continuous innovation and adaptation to changing user needs and technological capabilities. What began as simple text-based communication has evolved into a multifaceted platform offering diverse functionalities and catering to a wide range of use cases.

2.3.1 Text-Based Communication

Online chat first emerged as a primarily text-based communication platform, offering real-time messaging between users. This mode of communication became popularized thanks to platforms such as AOL Instant Messenger (AIM) and Yahoo! Messenger, which allowed users to engage in conversations with people from all around the world, whether they were friends, family, or strangers. The simplicity and immediacy of this text-based chat service laid the foundation for the widespread adoption of online chat services that we see today.

2.3.2 Multimedia Integration

Online chat services have come a long way since their inception. With the advancement of technology and improvement in internet bandwidth, these services now allow users to incorporate multimedia elements such as images, videos, and voice messages. Platforms like WhatsApp, Facebook Messenger, and Snapchat have completely transformed the way people communicate by enabling them to share rich media content in addition to text. This evolution has significantly enhanced the expressiveness and engagement of online conversations, enabling users to convey their emotions, share experiences, and express themselves more vividly.

2.3.3 Diverse Use Cases

Online chat services have become increasingly popular due to their versatility and adoption across various domains and use cases. They are now used for casual conversations among friends, business communications, and customer support, making them an indispensable tool for communication in the digital age. With chat rooms, private chats, and integrated chat features in social networking platforms and online games, online chat services have become even more useful, catering to the diverse needs and preferences of users.

2.3.4 Integration with Social Networking and Gaming

The incorporation of chat features on social networking platforms such as Facebook and Twitter have revolutionized the way people engage and connect online. These platforms not only facilitate text-based messaging but also allow users to share updates, photos, and videos with their social circles instantly. Similarly, online gaming platforms have capitalized on chat functionality to enhance multiplayer gaming experiences, enabling players to communicate, strategize, and establish virtual communities.

2.3.5 Ongoing Evolution and Future Directions

The evolution of online chat is ongoing, with continuous advancements in technology driving new features and applications. For example, the rise of AI-powered chatbots and virtual assistants has further diversified the use cases of chat services, offering automated customer support, personalized recommendations, and enhanced user engagement. Additionally, the integration of augmented reality (AR) and virtual reality (VR) into chat applications is on the horizon, promising even more immersive and interactive communication experiences.

2.3 TECHNOLOGIES USED IN CHAT APPLICATIONS

Creating and providing online chat applications requires a variety of essential tools and technologies. Each element plays a significant role in facilitating real-time communication and improving user experience. From the infrastructure in the background to the front-end interfaces and back-end processing, every aspect of online chat depends on a combination of tools and technologies to ensure smooth functionality and dependability.

Internet Infrastructure

Every online chat application relies on the Internet as the foundation for transmitting messages between users in real-time. The Internet creates the infrastructure necessary for establishing connections, routing data packets, and guaranteeing the timely delivery of messages across various devices and networks.

Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) are essential in designing and customizing the visual appearance and user interface of online chat applications. Developers can use CSS to modify and style chat interfaces, including the layout, colors, fonts, and animations. By applying CSS rules and stylesheets, developers can create visually appealing and user-friendly chat interfaces that improve user engagement and usability.

Databases

Online chat applications rely heavily on databases as a storage backbone to securely store and retrieve user information, message logs, and other relevant data. By utilizing databases like MySQL, PostgreSQL, or MongoDB, developers can ensure data integrity and accessibility, and keep user profiles and chat histories safe and secure.

JavaScript

JavaScript is a crucial element of online chat applications, as it provides dynamic functionality and interactivity on both the client and server sides. On the client side, JavaScript allows for real-time updates, message notifications, and user interactions within the chat interface. On the server side, frameworks such as Node.js utilize JavaScript to handle message routing, user authentication, database operations and server-side processing.

Frameworks and Libraries

Frameworks and libraries play a significant role in accelerating the development process and enhancing the functionality of online chat applications. React.js is a popular framework that provides a strong foundation for building interactive and responsive user interfaces. Bootstrap is a library that offers pre-designed components and stylesheets for speedy prototyping and design consistency. Other tools like Postman also aid in API development and testing, ensuring seamless integration and communication between front-end and back-end components.

Web Browsers and Text Editors

Web browsers and text editors are essential tools for accessing and developing online chat applications. With web browsers, users can access chat interfaces from any device that has internet connectivity. Text editors such as Visual Studio Code (VS Code) provide developers with a powerful environment to write, debug, and test chat application code. These tools work together to enable seamless communication and collaboration in the development and deployment of online chat applications.

2.4 SECURITY CONSIDERATIONS

Ensuring the confidentiality, integrity, and availability of user data and communications is crucial in the design and implementation of online chat applications. To achieve this, various security measures and protocols are employed to mitigate potential threats and vulnerabilities. These measures safeguard user privacy and protect against unauthorized access and malicious activities.

Encryption

Ensuring the confidentiality, integrity, and availability of user data and communications is crucial in the design and implementation of online chat applications. To achieve this, various security measures and protocols are employed to mitigate potential threats and vulnerabilities. These measures safeguard user privacy and protect against unauthorized access and malicious activities.

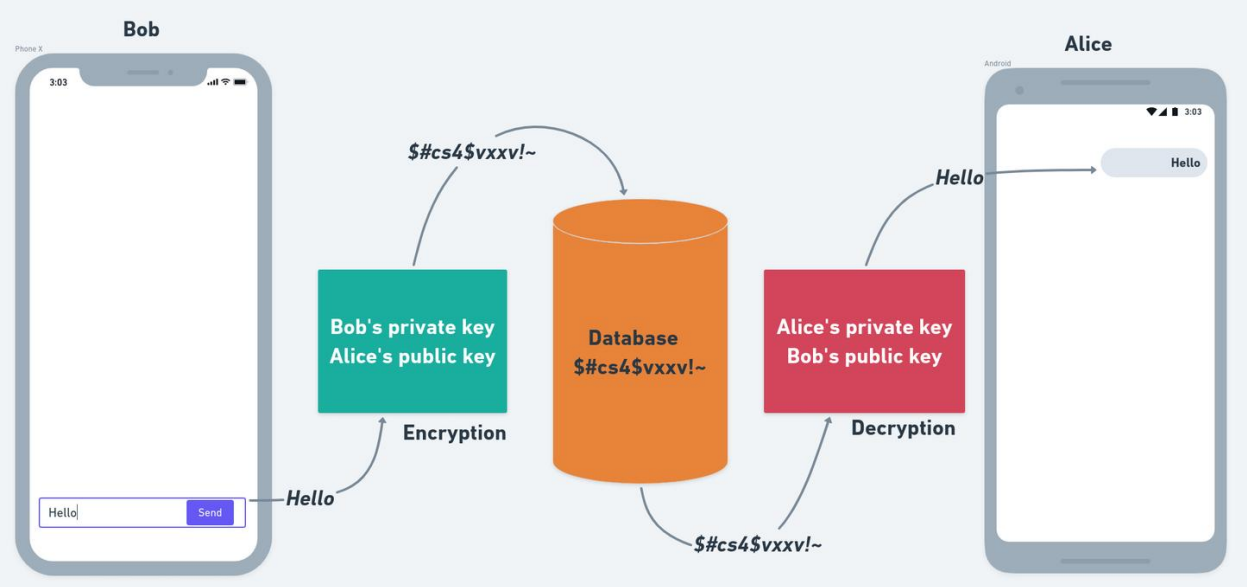

figure 2.1: A visual representation of Encryption

Secure Authentication

Having secure authentication mechanisms is crucial to confirm the identity of users and prevent unauthorized access to chat applications. To authenticate users securely and prevent account hijacking or unauthorized login attempts, robust password policies, multi-factor authentication (MFA), and OAuth (Open Authorization) protocols are used. In addition, biometric authentication methods like fingerprint or facial recognition provide an extra layer of security by verifying unique biological traits.

figure 2.2: Unique Authentication System in End-To-End Encrypted Chat App

Access Control

Access control mechanisms are used to limit user access to important features and functionalities within chat applications. This ensures that only authorized users can perform specific actions or view privileged information. There are different types of access control mechanisms such as role-based access control (RBAC), access control lists (ACLs), and permission settings. These mechanisms allow administrators to define user roles, privileges, and access levels based on their responsibilities and organizational hierarchy. By implementing granular access

controls, chat applications can reduce the risk of unauthorized access or manipulation of sensitive data.

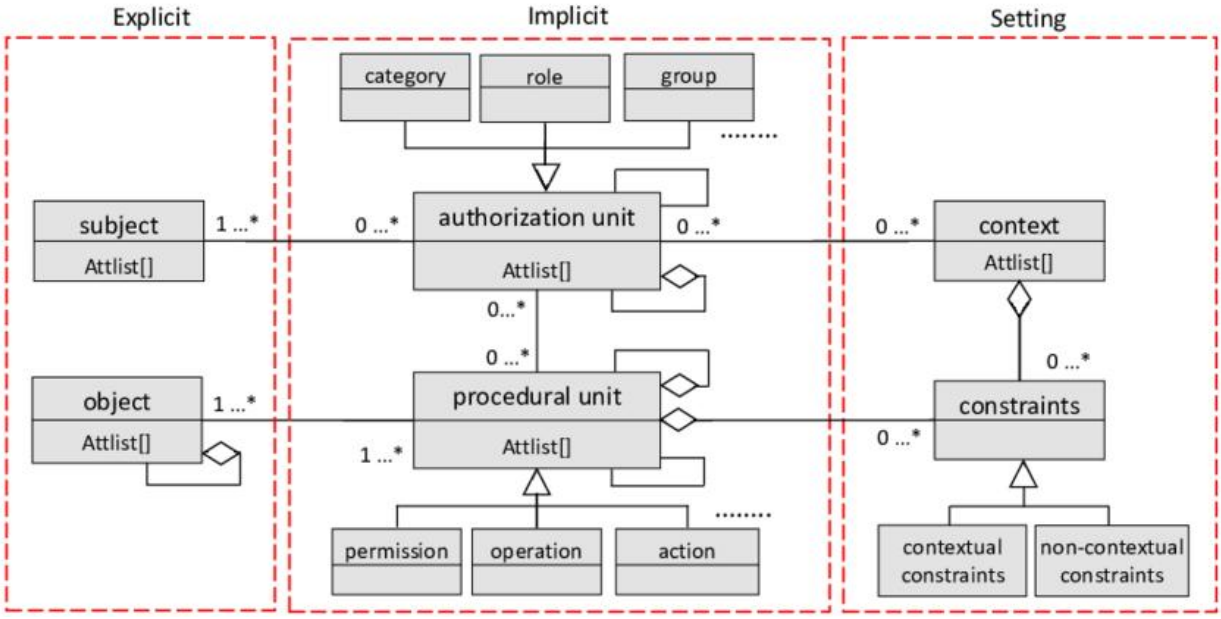

Figure 2.3: A visual representation of an Access Control Metamodel

Secure Transmission

It is essential to secure the transmission of data between clients and servers to prevent any interception or tampering of messages during transit. To ensure confidentiality and integrity of communications, Transport Layer Security (TLS) and Secure Socket Layer (SSL) protocols encrypt data transmitted over the network. Additionally, Secure Hypertext Transfer Protocol (HTTPS) encrypts HTTP traffic to protect sensitive information like user credentials and session tokens from eavesdropping or interception by malicious actors.

Threat Detection and Prevention

It is crucial to have continuous monitoring and threat detection mechanisms in online chat applications to identify and mitigate potential security threats and vulnerabilities. To achieve this, intrusion detection systems (IDS), anomaly detection algorithms, and real-time monitoring tools are used to analyze network traffic, user activities, and system logs and detect any suspicious behavior or unauthorized access attempts. Additionally, proactive measures such as regular

security audits, penetration testing, and vulnerability assessments help to identify and address security weaknesses before they can be exploited by attackers.

Data Privacy and Compliance

Ensuring compliance with data privacy regulations and standards is crucial in protecting user privacy and maintaining legal and regulatory compliance. Chat applications must comply with privacy laws, such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA), which govern the collection, processing, and storage of personal data. Incorporating privacy-enhancing features, such as data anonymization, pseudonymization, and consent management mechanisms, can help protect user privacy and reduce the risk of data breaches or compliance violations. Developers can create secure and resilient communication platforms that inspire trust, protect user privacy, and safeguard sensitive information against evolving cyber threats and vulnerabilities by incorporating robust security measures and best practices into the design and development of online chat applications.

Password Hashing with bcrypt.js

In addition to the security measures mentioned earlier, using bcrypt.js for password hashing can further enhance the security of online chat applications. Password hashing is a critical component of user authentication, ensuring that user passwords are securely stored and protected against unauthorized access or decryption. bcrypt.js is a JavaScript library that implements the bcrypt hashing algorithm, which is specifically designed for securely hashing passwords. Unlike traditional hash functions such as MD5 or SHA-1, bcrypt integrates salt and key stretching techniques to produce secure and robust password hashes.

Salting

Bcrypt.js uses a unique randomly generated value, called "salt", that is added to each password hash. This feature adds an extra layer of security by preventing the use of precomputed hash tables known as rainbow tables, which are employed by attackers to crack passwords. Each password hash produced by bcrypt.js is associated with a different salt value, which ensures that even identical passwords will have different hash values.

Key Stretching

bcrypt.js is designed to make it difficult for attackers to crack passwords. It does this by employing a technique called key stretching. This involves repeatedly applying the hash function to the password multiple times, thus increasing the computational cost of generating password hashes. By slowing down the hashing process through this method, bcrypt.js makes it computationally intensive for attackers to brute-force or crack passwords. The number of rounds of the hashing process is configurable, allowing developers to adjust the level of security according to their needs.

Resistance to Brute-Force Attacks

Bcrypt.js is a secure password hashing function that uses a combination of salt and key stretching to make it highly resistant to brute-force attacks. In such attacks, attackers try every possible combination of characters to guess passwords. Even if attackers have access to the password hashes, they face significant challenges in cracking bcrypt-hashed passwords due to the computational overhead introduced by salt and key stretching.

2.5 REVIEW OF RELATED WORKS

The landscape of online chat applications has been shaped by various research endeavors and practical implementations. This review examines several notable works and their associated methodologies and findings, shedding light on the evolution and current state of online chat technology.

WhatsApp - Real-time Messaging Platform

WhatsApp stands out as one of the most widely used real-time messaging platforms, enabling users to exchange text messages, images, videos, and voice recordings instantly. With a user base of over 2 billion, WhatsApp has revolutionized personal and business communication worldwide (WhatsApp).

Slack - Business Communication Tool

Slack offers a comprehensive platform for team communication and collaboration, featuring instant messaging, file sharing, and integration with various productivity tools. With customizable channels and robust search functionality, Slack enhances workplace communication efficiency (Slack).

Facebook Messenger - Social Messaging App

Facebook Messenger provides a versatile platform for social messaging, allowing users to connect with friends and family through text, voice, and video chats. With features like reactions, stickers, and group chats, Facebook Messenger offers a dynamic communication experience (Facebook Messenger).

Signal - Secure Messaging Application

Signal prioritizes user privacy and security, offering end-to-end encrypted messaging for individuals and groups. With a focus on data protection and anonymity, Signal ensures secure communication without compromising user privacy (Signal).

Discord - Community Chat

Discord caters to the gaming and community-building niche, providing users with text, voice, and video chat capabilities. With features like server customization and role-based permissions, Discord fosters vibrant online communities.

2.6 CASE STUDY OF EXISTING APPLICATIONS

Case Study 1:

WhatsApp

- **Background**: WhatsApp is a widely used messaging application with over 2 billion active users worldwide. It aims to provide a secure, reliable, and feature-rich platform for communication.
- **Implementation Details**: WhatsApp was developed using a combination of technologies, including Erlang for backend infrastructure and React Native for cross-platform mobile development.
- **Features and Functionality**: WhatsApp offers real-time messaging, voice and video calls, group chats, file sharing, and end-to-end encryption for privacy and security.
- **Challenges Faced**: WhatsApp has faced challenges related to handling a large user base, ensuring scalability, and addressing privacy concerns, particularly regarding data sharing with its parent company, Facebook.
- **Outcomes and Results**: WhatsApp's user-friendly interface, extensive feature set, and widespread adoption have made it one of the most popular messaging platforms globally, despite occasional controversies.

- **Lessons Learned**: WhatsApp's success highlights the importance of prioritizing user experience, security, and scalability in developing chat applications.

Case Study 2:

Slack

- **Background**: Slack is a collaboration hub that aims to streamline communication and workflow within teams and organizations. It is used by millions of users across various industries.
- **Implementation Details**: Slack is built using technologies such as PHP for backend services, JavaScript for frontend development, and Electron for desktop applications.
- **Features and Functionality**: Slack offers real-time messaging, channel-based communication, file sharing, integrations with third-party services, and customizable notifications.
- **Challenges Faced**: Slack has faced competition from other collaboration tools, as well as challenges related to maintaining performance and scalability as its user base continues to grow.
- **Outcomes and Results**: Slack has become a leading platform for team communication and collaboration, with its intuitive interface, extensive integrations, and focus on productivity.
- **Lessons Learned**: Slack's success underscores the importance of adaptability, innovation, and user engagement in the competitive landscape of chat applications.

2.7 COMPARATIVE ANALYSIS

Each of the reviewed platforms contributes unique features and functionalities to the realm of online chat applications. WhatsApp excels in personal communication, while Slack dominates the business communication space. Facebook Messenger offers a seamless social messaging experience, while Signal prioritizes privacy and security. Discord, on the other hand, specializes in community building within specific interest groups.

2.8 SYNTHESIS AND IMPLICATIONS

The synthesis of these real-world examples underscores the diverse challenges and opportunities in online chat application development. Future research and development efforts could focus on integrating advanced features such as multimedia support, enhanced security measures, and interoperability between different platforms.

By building upon the successes and innovations of existing platforms and leveraging emerging technologies, researchers and practitioners can continue to advance the state-of-the-art in online chat applications, ultimately enhancing user experiences and facilitating seamless communication in various contexts.

Table 2.1 Summary Table of Related Works

Author / Year	Title	Methods Adopted	Strengths	Research Gaps	Future Work

Ashutosh Kumar, Atul Singh (2022)	Group Chatting Application	The project utilizes Java, multi-threading, and client-server concepts to develop an online chat system. It enables both private and public messaging and facilitates the sharing of resources like files, images, and videos.	Utilizes Java, a widely used programming language. Incorporates multi-threading for efficient handling of multiple clients. Implements client-server architecture.	The paper does not delve deeply into security measures implemented to safeguard user data. It lacks discussion on real-world testing scenarios or user feedback regarding usability and performance. There's no mention of how the application handles potential issues like server downtime or network failures.	Future enhancements include adding features such as file transfer and voice/video messaging, improving security with encryption and authentication, conducting user studies for UI/UX feedback, exploring scalability options, and enhancing reliability with fault tolerance and automated testing.
Avinash Bamane (2012)	Enhanced Chat Application	The researchers aimed to design an enhanced chat application that enables users to communicate using text, figures, and	The application provided features such as predictive texting, painting and coloring, instant messaging, drawing tools, and	While the paper focused on designing and describing the features of the enhanced chat application, there was limited	Future research opportunities include evaluating the usability and effectiveness of the enhanced chat application

		diagrams. The application was intended for professional institutions and operated on a client-server architecture within a LAN.	image transfer capabilities. It addressed the limitation of traditional chat applications by allowing users to communicate using diagrams and figures.	discussion on user feedback, usability testing, scalability, and security considerations.	through user studies, improving security measures, enhancing scalability for larger user bases, and integrating additional collaboration tools to enhance communication within professional organizations.
Noor Sabah (2017)	Developing an End-to-End Secure Chat Application	Qualitative synthesis	The proposed application offers end-to-end encryption, social key exchange, and resistance to server attacks.	The paper doesn't mention how the application would handle message delivery if a user were not online.	The authors propose to implement and test the proposed chat application.
Mr.Sachin Bansal(2023)	Development of Real-Time Chat Application Using ReactJS and Firebase	The project utilizes ReactJS with Firebase to create a real-time chat application supporting features such	The project leverages modern web development tools like ReactJS and Firebase to deliver a seamless,	The literature review overlooks security measures, user feedback collection, and scalability	Future enhancements involve adding multi-language support, improving UI based on feedback,

		as user registration, login, chat room creation, messaging, user blocking, notifications, message deletion, and profiles, implemented with HTML, CSS, and JavaScript.	user-friendly chat experience with real-time updates, advanced features, and responsive design across various devices, utilizing Firebase Realtime Database for efficient message storage and retrieval.	considerations beyond Firebase hosting.	implementing file transfer and voice/video messaging, addressing security with encryption and authentication, conducting usability testing for UX improvements, exploring scalability beyond Firebase, and enhancing reliability through fault tolerance and automated testing.

BR. Gayathri, C.(2020)	Multi-User Chat Application	The Research paper outlines a method using client-server model and multi-	The proposed system offers two-way communication, group chat, file transfer, and	The research paper does not specify how the system would address security	Improving the security concerns like network threats.

		threading for their multi-user chat application.	security features	concerns like network threats	
Jhalak Mittal (2020)	Secure Real-Time Communication Application Development using Android and Firebase	The study presents a software application developed on the Android platform with backend support from Google Firebase. It employs modern encryption algorithms and lightweight storage techniques to ensure security and efficiency in real-time communication.	The study employs the Android platform alongside Firebase for real-time database and cloud services, integrating the XSalsa20 encryption algorithm for message security, Firebase Cloud Messaging for efficient message delivery, and outlines a comprehensive architecture ensuring secure storage and transmission of data.	Lacks discussion on user feedback collection methodologies or usability testing, potentially overlooking user experience aspects. Does not address scalability considerations beyond Firebase, limiting discussion on accommodating larger user bases. Focuses primarily on secure messaging between friends, with limited exploration of additional features or broader application scenarios.	Enhancements include adding voice messaging, group calling, live streaming, auto message deletion, and personalized message tones, alongside implementing chat room creation during conversations, while continuously improving security measures through advanced encryption algorithms for enhanced data protection.

Aditya Yadav (2007)	Real-Time Chat Applications	The literature review examines real-time chat applications, encryption technologies, and their implementation methodologies.	Provides insights into the significance, application, and encryption technologies employed in real-time chat applications.	Lacks in-depth discussion on user feedback collection methodologies, scalability considerations, and usability testing.	Future research could focus on addressing usability issues, enhancing scalability beyond current technologies, and exploring advanced encryption algorithms.
Khushi Srivastava (2023)	Real-Time Chat Box System using MERN: A Comprehensive Review	Utilized qualitative analysis and literature review to explore the evolution, features, and impact of chat applications, focusing on the MERN stack development approach.	Offers insights into the design and implementation of real-time chat applications, highlighting key technologies like React.js, Socket.io, Node.js, Express.js, and MongoDB.	Limited discussion on user feedback collection methodologies, scalability considerations, and usability testing. Future works could explore these areas further.	Potential future research could focus on enhancing user experience through improved interface design, conducting extensive scalability testing, and incorporating advanced security features.
Dr. Ashish Sharma. (2023)	Development of a Real-Time Chat Application with N-TEA Encryption	Utilizing Django framework for backend, HTML/CSS for frontend,	Integration of real-time chat functionality with robust encryption for secure	Lack of detailed discussion on user experience, scalability	Potential enhancements include adding features like file sharing

	Algorithm	and implementing N-TEA encryption algorithm for secure real-time messaging.	communication, clear explanation of frontend and backend technologies used.	considerations, and potential performance issues under heavy loads.	and voice chat to enhance user experience and expanding the discussion on scalability and performance optimization.
Prof. Amrita. A. Shirode. (2022)	Literature Review of Secure Chat Applications	The literature review involved analyzing existing chat applications in the market, researching encryption-decryption algorithms, integrity algorithms, key-exchange algorithms, and authentication methods.	Comprehensive analysis of existing systems, identification of security vulnerabilities, and implementation of advanced encryption techniques.	Identified shortcomings in existing chat applications, particularly in terms of security and user interface design.	Enhancements include incorporating features like file transfer, video messaging, audio/video calls, group calls, authorization services, database management, web support, and voice-based chat to improve the effectiveness and usability of the application.
Diksha Makode. (2023)	Literature Review of Real-Time Chat Applications.	The review involved analyzing existing real-time chat applications,	Comprehensive understanding of the significance of chat	Identified gaps in existing chat applications regarding privacy,	Proposed enhancements include incorporating features such as

		studying technologies like ReactJS and Firebase, and collecting data on their features, functionalities, and user experiences.	applications in modern communication, identification of key features and functionalities desired by users, and recognition of the importance of privacy, security, and user-friendliness.	security, and user experience, emphasizing the need for improved solutions that prioritize anonymity, secure communication, and intuitive design.	multimedia sharing, cross-platform communication, end-to-end encryption, group chats, notifications, and a user-friendly interface to improve the effectiveness and usability of real-time chat applications.
Swanand Joshi. (2014)	Literature Review of Sentiment Analysis in Mobile Chat Applications	The review outlines sentiment analysis systems in mobile chat apps, employing logistic regression for sentiment classification from text features and utilizing PUSH technology for efficient sentiment updates dissemination	The proposed system provides automated sentiment analysis in mobile chat apps, leveraging text message data without user intervention and utilizing PUSH technology for efficient distribution.	The proposed system offers automated sentiment analysis without user intervention, leveraging text message data. It introduces a novel approach to sentiment analysis in mobile chat applications and utilizes PUSH technology for efficient	Future research could focus on integrating the system into existing chat apps or developing it as a standalone platform, with potential enhancements in sentiment analysis features and accuracy, as well as integration with a broader array

		.		distribution.	of chat applications.

Shreya Bisht (2022)	News Chat APP: Connecting People During Crisis	Utilized Java, XML, AES Algorithm, and Firebase for analysis, development, design, and validation, incorporating two servers for user chat and news.	Provides real-time communication during crises, integrates news updates, offers end-to-end encryption, and includes various media sharing features.	Lacks detailed discussion on data analysis methodology and user feedback integration for continuous improvement.	Potential enhancements could focus on improving user engagement through personalized news feeds, enhancing user interface, and expanding platform compatibility.
Mrs. P. A. Satarkar (2022)	Realtime Chat Application Using PHP with MySQL	Employed HTML, CSS, JavaScript, PHP with MySQL for server-side scripting, and database management.	Offers real-time communication, user-friendly interface, efficient data management, and secure user authenticatio	Limited discussion on scalability and handling large numbers of users, lacks exploration of advanced chat features and	Potential enhancements could focus on scalability improvements, integration with additional features like file sharing and

			n.	integration with other platforms.	voice/video calling, and enhancing user experience through UI/UX enhancements.
Mr. Karan Mandve (2024).	Chat Application with Hate Speech Recognition	The paper proposes the development of a chat application equipped with a hate speech detection system. It incorporates machine learning and natural language processing algorithms to analyze messages in real-time for hate speech and offensive language.	The application aims to provide a safe and inclusive environment for online communication by proactively detecting and filtering out hate speech and abusive content. It offers features such as user registration, real-time messaging, hate speech detection, content filtering, and warning/blocking mechanisms to promote respectful dialogue and	While the paper discusses the importance of addressing hate speech and abusive language online, it does not delve deeply into the specific challenges and limitations associated with hate speech detection, such as the nuances of language, cultural context, and evolving forms of online harassment.	Future research could focus on refining the hate speech detection algorithms to improve accuracy and reduce false positives/negatives. Additionally, exploring methods to address cultural and contextual variations in hate speech, enhancing user reporting mechanisms, and implementing user feedback loops for continuous improvement would be

			community engagement.		valuable avenues for future work.
Nikhil Chaudhari (2018)	Chatting Application with Real-Time Translation	The paper presents the development of a chat application with real-time translation capabilities, utilizing Google Cloud API and Google's Machine Learning for language translation. The application is built using Ionic Framework 3.6.0, Angular 4.3.*, Firebase (MongoDB) for the database, and Express (Node.js) for the backend.	The application provides real-time translation of messages across multiple languages along with image processing features like face detection, expression analysis, image backup, theft alert, landmark detection, and OCR, utilizing Google's technology in a hybrid environment for cross-platform compatibility.	While the paper discusses the motivation behind the project and its goals, it lacks a detailed literature review on existing chat applications with real-time translation capabilities. Additionally, there is limited discussion on the specific challenges faced in implementing real-time translation, such as language accuracy, speed, and resource consumption.	Future research could focus on enhancing the real-time translation algorithms to improve accuracy and reduce latency. Additionally, exploring methods to integrate voice translation, dialect recognition, and context-aware translation would enhance the user experience.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1 NTRODUCTION

In this chapter, I will outline the research methodology and project activities that were undertaken, highlighting the process from conception to conclusion. Developing an application is like constructing a house; while having the right tools is crucial, a solid foundation is of utmost importance.

Consequently, this chapter will provide a detailed account of the tools and approaches utilized in this project, akin to laying a robust foundation. Furthermore, this chapter will delve into the research design, data collection, analysis, and presentation techniques that were employed throughout the project. It will also explain why certain methods were chosen over others, providing a clear understanding of the research process for anyone who wishes to replicate this project or conduct similar research. The methods used in this project were carefully selected to ensure that the research was conducted ethically and transparently.

This chapter will provide a comprehensive explanation of the steps taken to ensure data privacy and security, along with the measures taken to minimize any potential risks and biases.

Overall, this chapter aims to give readers a thorough understanding of the research process and its outcomes in a clear and concise manner. By providing a detailed account of the research methodology and project activities, this chapter will enable readers to comprehend the methods used in the project, the rationale behind the choices made, and the implications of the research findings. This information will be of great value to researchers, practitioners, and anyone interested in the research field.

3.2 SYSTEM ANALYSIS

System analysis is a comprehensive process that requires a thorough understanding of an existing or proposed system. By evaluating a system's objectives, functions, and features, system analysts can identify areas where improvements can be made. The process involves examining the system's constraints and conditions, as well as its strengths and weaknesses, to determine its desirability and effectiveness. Once potential areas for improvement have been identified, system analysts can suggest enhancements that will help the system meet its goals and objectives more efficiently and effectively. The goal of system analysis is to ensure that a system is working optimally and can meet the needs of its users.

3.3 ANALYSIS OF EXISTING SYSTEM

When it comes to designing a new system, analysis is an essential step. Analysis requires a thorough examination of the existing system to identify its strengths and weaknesses to create a new system that addresses the limitations of the current one.

In other words, it involves delving deep into the processes and operations of the system and understanding how they interrelate. During this analysis, data is collected on the available files, decision points and transactions handled by the present system.

This data is then used to create specifications for the new system. By conducting a comprehensive analysis, stakeholders can ensure that the new system is more efficient, effective, and able to meet the needs of the users better.

By following the steps provided, it is possible to accurately outline the perimeter of the new system that is being considered:

a) Considering the challenges and changing needs.

b) Evaluating the advantages and disadvantages of the system and identifying any new areas that require attention.

3.3.1 Problems of Existing System

- Limited Platform Compatibility: Some chat applications are limited to specific platforms or operating systems, reducing their flexibility and usability.

- Inadequate Authentication Mechanisms: Existing systems often lack robust authentication mechanisms, making them vulnerable to unauthorized access and security breaches. For instance, some chat applications allow users to sign up with simple email addresses, usernames or phone numbers, without any verification of their identity. This lack of secure authentication methods can lead to security vulnerabilities and unauthorized access to the system.

- Lack of Online Gathering Spaces: Students do not have dedicated online spaces to gather and discuss related topics, limiting their opportunities for collaborative learning and discussion.

3.3.2 Solution to Problems of the Existing System

The solution proffered is to develop an online chat application that can perform the below:

a) Web-Based Application Using MERN Technology: MERN technology ensures that the chat application is accessible from any device with a web browser and internet connectivity. Users can access the chat application from various platforms, including desktop computers, laptops, tablets, and smartphones. The web-based approach ensures cross-platform compatibility, allowing users to access the chat application seamlessly regardless of their device or operating system.

b) Enhanced Authentication Mechanisms: Allow only registered students to log in using their unique matriculation numbers. Implement a robust authentication system that verifies the identity of users based on their matriculation numbers. Ensure that only authorized students have access to the chat application, enhancing security and preventing unauthorized access.

c) Dedicated Online Spaces for Discussions: Provide dedicated online spaces for students to gather and discuss related topics. Create chat rooms or discussion forums within the application where students can engage in academic discussions with their instructors. Promote collaborative learning

and discussion among the student community by facilitating easy access to online discussion spaces.

It is important to carefully examine and record all rules and steps involved in a process using tools such as data flow diagrams, data dictionary, logical data structures, and miniature specifications. The new system being introduced is a more sophisticated and versatile version of the existing system that offers greater accessibility to users.

3.4 OVERVIEW PROPOSED NEW SYSTEM

The proposed new system aims to address the shortcomings of the existing chat application by introducing several enhancements and new features to improve user experience, security, and functionality.

1) **Enhanced Authentication and Security:**

- Implement a robust authentication system to verify and authenticate user information.

2) **Group Chat Management:**

- Allow users to create and manage group chats.
- introduce the ability for group chat creators to appoint administrators who can add, remove, or manage group members.
- Enable users to leave group chats at any time.

3) **Improved User Interface and Experience:**

- Redesign the user interface to make it more intuitive and user-friendly.
- Implement a responsive design to ensure the application is accessible across different devices and screen sizes.

4) **Enhanced Messaging Features:**

- Implement end-to-end encryption to ensure the security and privacy of user messages.

5) **User Profile Enhancements:**

- Allow users to view and update their profiles with additional information such as profile pictures, status messages, and contact details.
- Introduce a status indicator to show when users are online, offline, or away.

6) **Improved Notification System:**

- Implement real-time notifications for new messages and other important events.

7) **Advanced Search and Filter Options:**

- Introduce advanced search and filter options to help users find specific messages, contacts, or group chats quickly.
- Implement search filters based on date, sender, keyword, and other criteria.

By implementing these changes, we believe that the new system will provide users with a more secure, intuitive, and feature-rich chatting experience.

3.5 FEASIBILITY STUDY

A Project Feasibility Study is a critical document that helps organizations determine the viability of a proposed project before committing resources to it. The study helps project teams identify potential obstacles and assess risks associated with a project.

It is an essential part of the project planning process as it helps project managers and decision-makers to make informed decisions about whether to proceed with a project or not.

Feasibility Studies can be used to evaluate a range of projects, from small-scale initiatives to large complex projects. During the study, team members need to

gather information about the project's technical feasibility, economic viability, legal requirements, operational feasibility, and scheduling feasibility. This information is then analyzed to produce a feasibility report that outlines the recommended course of action.

The report typically includes a summary of the project, an overview of the project's objectives, a review of the alternatives, a detailed analysis of the proposed solution, and a conclusion that outlines the recommended course of action.

The feasibility report is a critical document that helps stakeholders understand the project's viability and provides a clear path forward.

3.6 REQUIREMENT ANALYSIS

Requirements analysis is a crucial step in the process of developing a new product or modifying an existing one. It involves identifying and evaluating the expectations of the end-users and stakeholders.

The requirements, which are the features that the product must possess, should be measurable, relevant, and detailed. In software engineering, these requirements are often referred to as functional specifications.

The purpose of the requirements analysis is to ensure that all stakeholders agree about what the product should do and what it should not do. The system requirement specification is a document that outlines the requirements of a product. It is a structured document that provides a detailed description of the product's features and functions.

The system requirement specification is created by gathering information from various stakeholders, including end-users, developers, and business analysts.

The document includes information on the product's purpose, its functionality, performance requirements, and any constraints or limitations that may impact the development process.

By creating a comprehensive system requirement specification, the development team can ensure that everyone is aligned on the product's objectives, and the development process can progress smoothly.

3.6.1 User Requirements

This section aims to outline the key features and capabilities that the proposed system must possess to meet its intended purpose. Specifically, the focus will be on the functional requirements, and these include:

i. The application ought to have the capability to verify and authenticate any information entered by the user and react suitably to any irregularities.

ii. The application must ensure that only one user is allowed to access an account through standard authentication procedures.

iii. The application must provide users with the ability to see the profiles of other users and their current status of activity.

iv. The application must have the ability to stop or block any unauthorized attempts to gain access.

v. The data entry system must possess the capability to identify errors or duplications in the entered data.

vi. The application should enable both the Administrator and Users to log in and perform different functions as per their respective roles.

vii. The application must enable all users to engage in a private conversation with any other user of their choosing.

viii. Users who create a group chat must be the only ones, or other administrators chosen by the user, able to add, remove, or manage group members.

ix. Users should have the freedom to leave group chats at their discretion.

3.6.2 System Requirements

In this part, we will discuss the conditions that the target system must fulfill to ensure that the software application works as intended. We will delve into the software and hardware requirements of the target computer system. A system requirement is a document that outlines the specifications of a system by gathering relevant information. This section will primarily concentrate on the software and hardware requirements.

Component	Requirement
RAM	2GBof RAM
Hard disk / Solid State Drives	10gb of hard disk space
Processor	2.0GHz or higher
OS	Windows 10 0r higher / Linux/ Ubuntu / Android / ios

Table 3.1 System Requirements

3.7 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the flow of data through a system or process. It illustrates how data is input, processed, stored, and output in a clear and concise manner.

DFDs are useful tools for understanding complex systems and identifying potential areas for improvement.

They are also used in the design and development of software systems to ensure that all data flows and interactions are properly accounted for. In simpler terms, a data flow diagram is a visual representation of how information moves through a system or process.

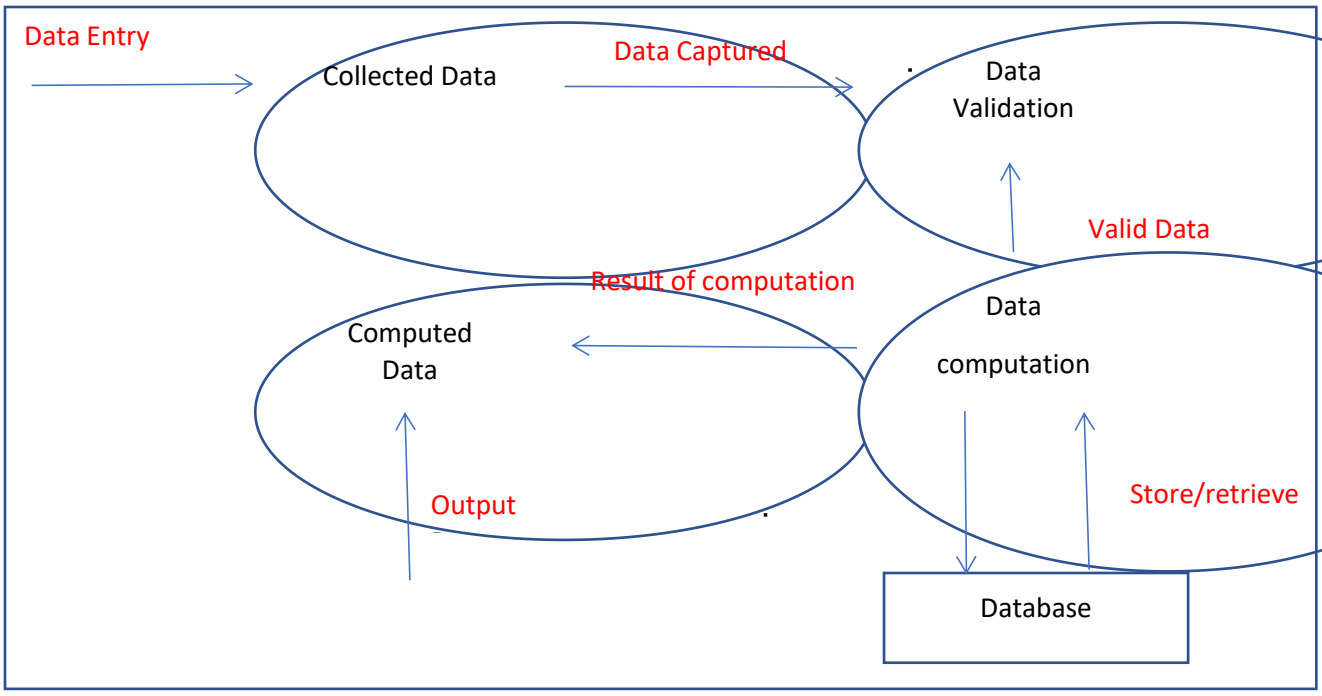

Fig 3.1: Data Flow Diagram

3.8 SYSTEM ARCHITECTURE

These models or diagrams help define the structure and relationships among the components of a system. They provide a high-level overview of the software elements, such as entities and their attributes, and the relationship between them. By having a clear understanding of the conceptual models, developers can design software systems that are efficient, scalable, and easy to maintain.

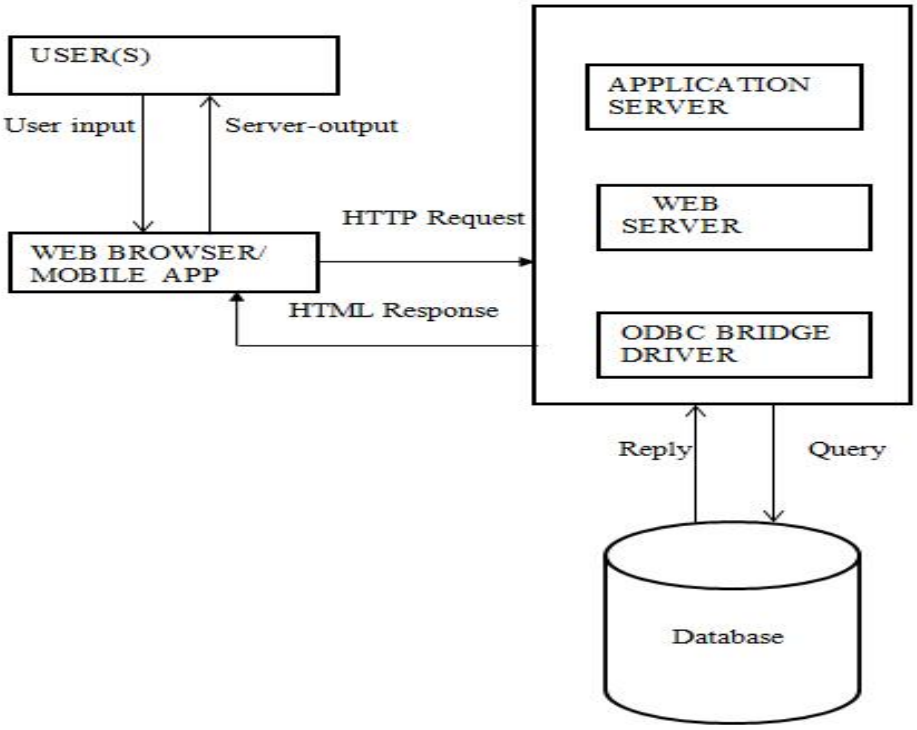

Figure 3.2: System Architecture

3.9 SYSTEM DESIGN

System design is an essential step in the process of creating a functional and efficient system.

It involves defining the various components, modules, interfaces, and data that are necessary to meet the specified functional requirements of the system.

The design phase is critical to the success of any system because it sets the foundation for the development process. By carefully analyzing user requirements and making informed design decisions, a well-designed system can meet the needs of users and perform its intended functions reliably. Documenting the design decisions made during this phase is vital to ensure that the system can be maintained, updated, and scaled in the future.

3.9.1 User Interface Design

The proposed system leverages the use of React components, Redux for state management, and Context API for global state sharing, we can achieve a robust and responsive UI that meets the needs of modern chat users.

React Components:

React components serve as the building blocks of the UI, encapsulating specific functionalities and visual elements. By breaking down the UI into reusable components, we can create a modular and maintainable UI architecture that promotes code reusability and scalability.

- **Chat Room Component**: Displays the list of chat rooms available for users to join. Utilizes Redux to fetch chat room data from the server and update the UI in real-time.
- **Message List Component**: Renders the list of messages within a selected chat room. Utilizes Redux to fetch message data and manage message state (e.g., loading, sending, receiving).
- **Message Input Component**: Allows users to input and send messages in the active chat room. Utilizes Redux to manage message sending and updates message state accordingly.
- **User Profile Component**: Displays user profile information and settings. Utilizes Context API to access user authentication state globally across the application.

Redux for State Management:

Redux provides a centralized state management solution, enabling predictable state updates and efficient data flow within the application. By maintaining application state in a single store and using actions and reducers to update state, Redux facilitates a clear and structured approach to managing UI state.

- **Global State**: Stores application-wide state such as user authentication status, chat room data, and message history.
- **Actions**: Define action types and creators to trigger state updates based on user interactions (e.g., sending messages, joining chat rooms).

- **Reducers**: Define reducers to specify how state changes in response to actions, ensuring that state updates are predictable and consistent across components.

Context API for Global State Sharing:

Context API provides a lightweight and efficient mechanism for sharing state across components without the need for prop drilling. By creating context providers and consumers, we can make application-wide state accessible to any component within the UI hierarchy.

- **User Authentication Context**: Manages user authentication state globally, allowing components to access user authentication status and user profile data.
- **Theme Context**: Allows users to customize the UI theme (e.g., light mode, dark mode) and persists theme preferences across sessions.
- **Notification Context**: Handles notification messages and alerts, providing a consistent way to display important information to users.

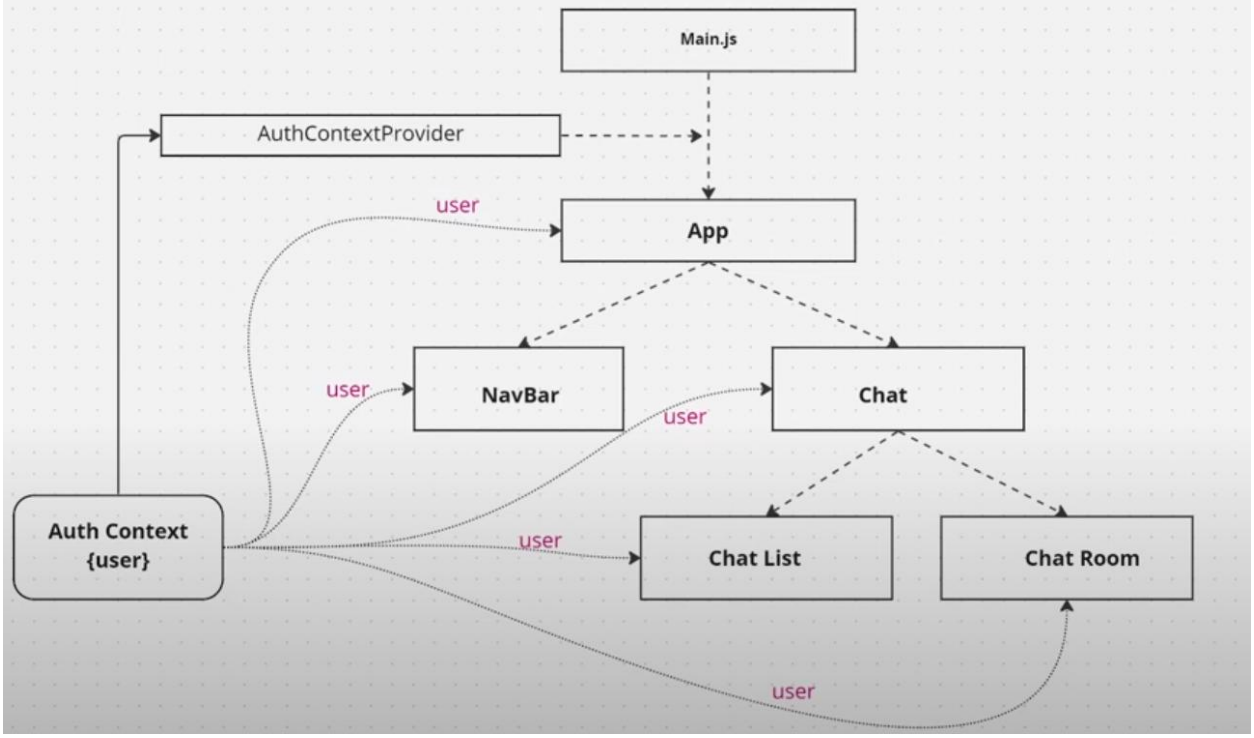

Figure 3.3: Authentication Context API

3.9.2 Database Design

The proposed system utilizes a MongoDB database to store and manage records. Designing a database involves planning and structuring data objects along with their relationships within the database. The database is set up to meet the information needs of the organization.

According to Beynon-Davies (2004), a database is a collection of related data items with a regular structure. It is organized in such a way that any design information contained in the collection can be easily accessed for viewing, editing, or deleting.

When implementing this system using MongoDB and Mongoose, the database will be structured to efficiently store and retrieve data in accordance with the organization's information requirements.

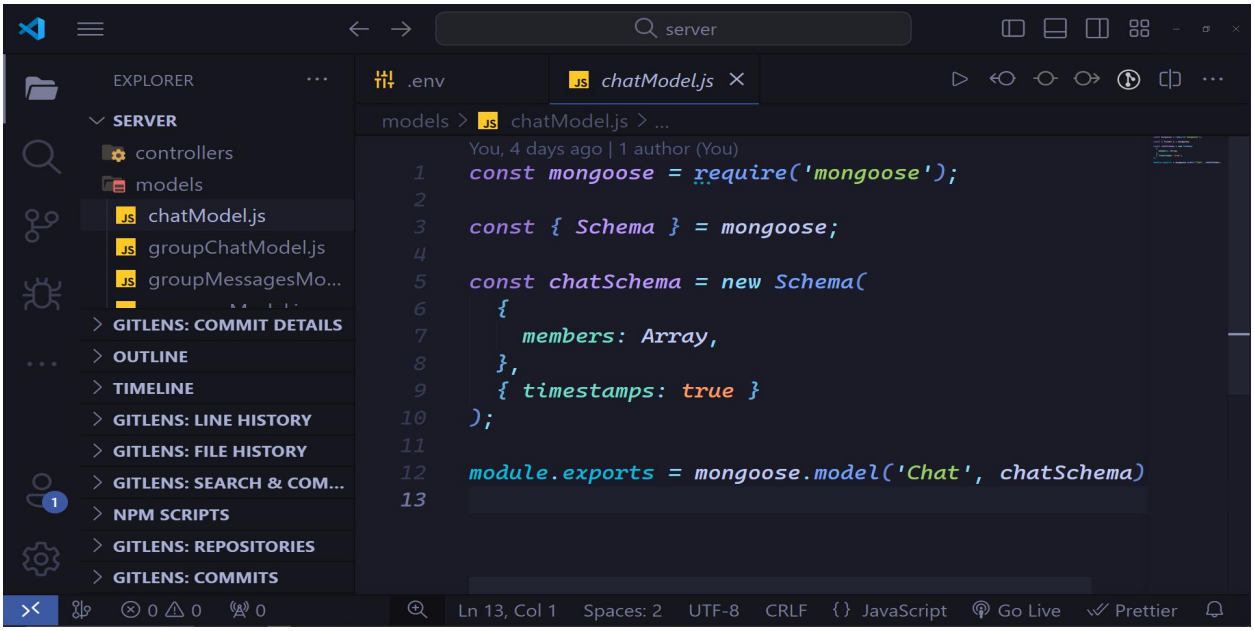

Figure 3.4: User Chat Model

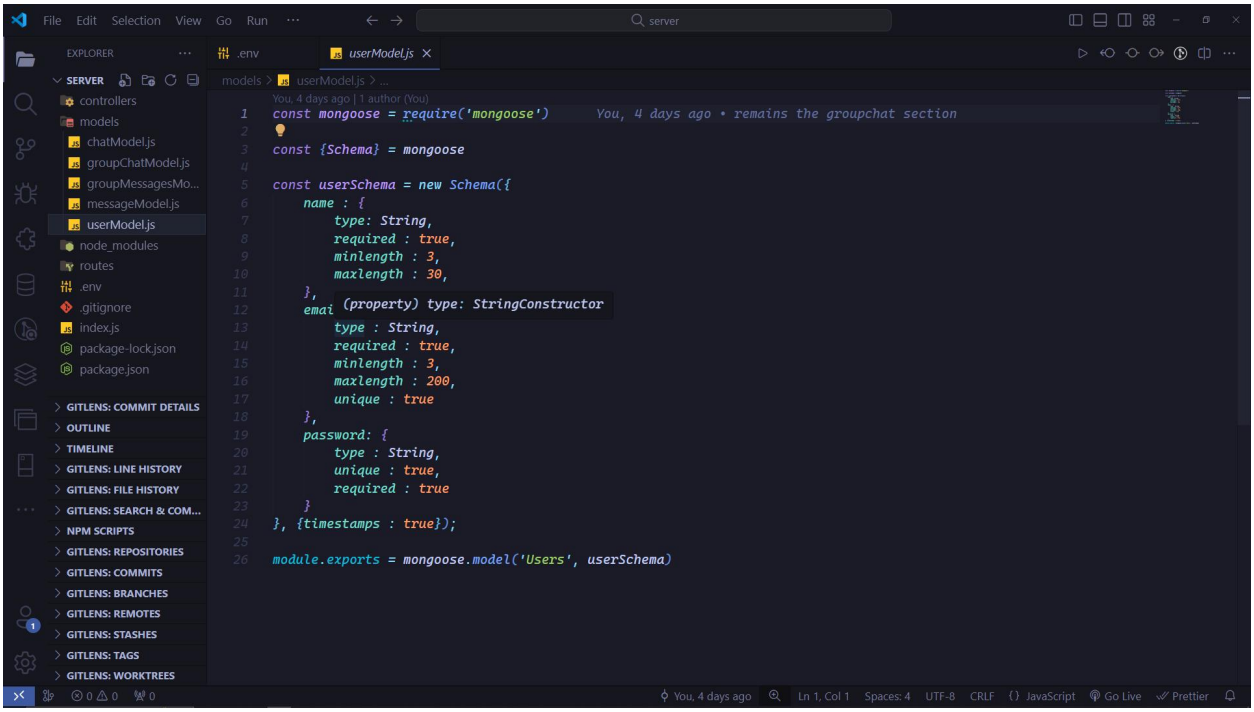

Figure 3.5: User Authentication Model

3.10 SYSTEM FLOWCHART

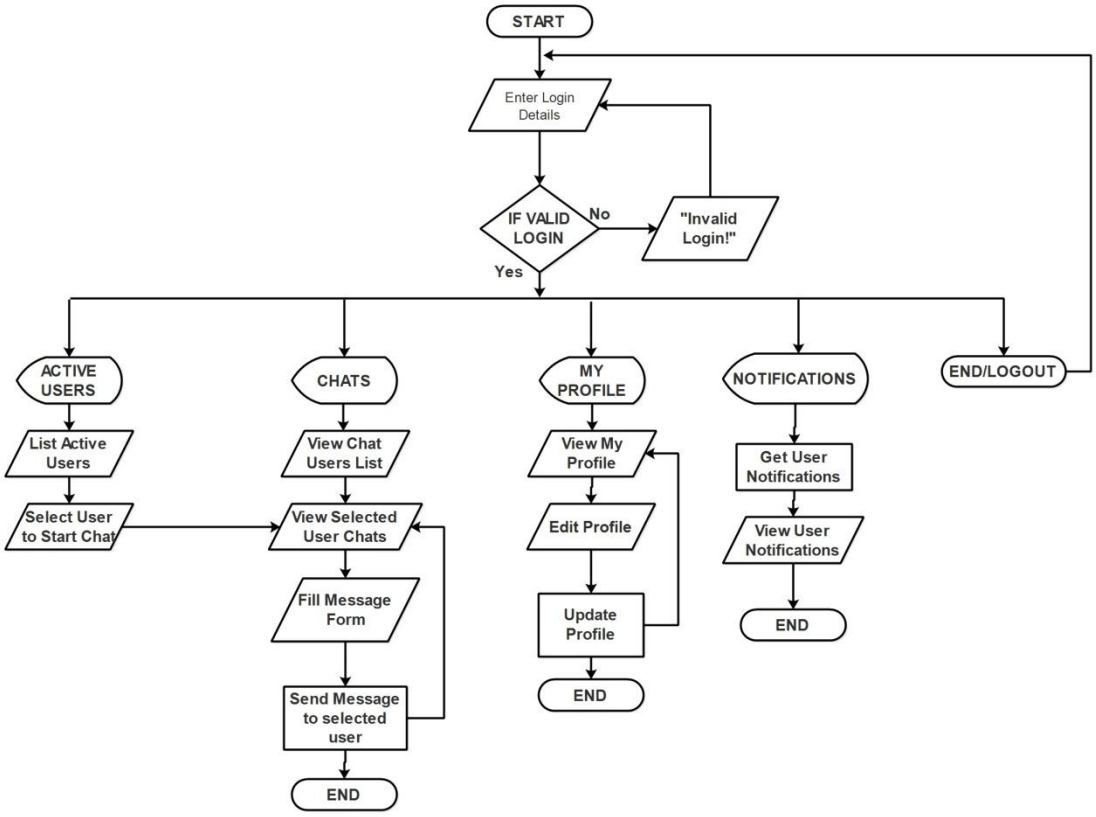

Figure 3.6: System Flowchart

3.11 PROGRAM FLOWCHART

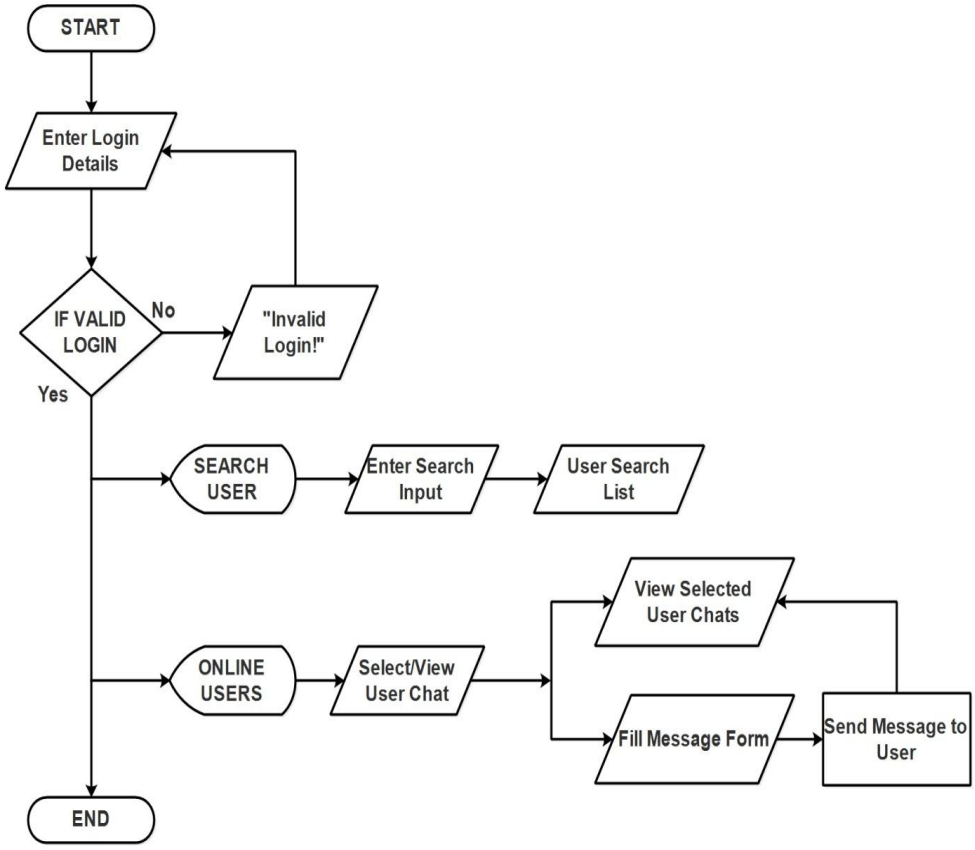

Figure 3.7: Program Flowchart

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.0 INTRODUCTION

This chapter emphasizes the execution of the system. It will delve deeply into the characteristics of the implementation languages, frameworks, and packages employed in this study, such as CSS, HTML, JavaScript, Node.js, Express, MongoDB, Mongoose, and React.

Furthermore, it will cover system testing approaches, the necessary computer specifications, and potential system maintenance challenges.

4.1 SYSTEM DEVELOPMENT

This phase involves the production of the actual code that will be delivered as the running system. Individual modules are well-designed, developed, and tested before being delivered to the next phase. The development of this system involves the use of modern web development technologies, including:

4.1.1 Node.js with Express.js (The Server-Side Scripting Phase)

Node.js with Express.js was used in the development of the system. It provides a platform for building fast and scalable server-side applications using JavaScript. Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

4.1.2 MongoDB (The Database Phase)

MongoDB is a NoSQL document database used for high volume data storage. It is a scalable, flexible, and high-performance open-source database designed to handle document-oriented storage. MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

4.1.3 React.js (The Frontend Phase)

React.js is a JavaScript library used for building user interfaces. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. It focuses on the components of the application, allowing the creation of reusable UI components.

4.1.4 HTML, CSS, and JavaScript (The Markup Phase)

Hyper-Text Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript were used during the development of this system. HTML provides the structure of the web pages, CSS is used for styling and layout, and JavaScript is used for interactivity and dynamic content.

4.2 SYSTEM IMPLEMENTATION

The implementation phase involves deploying the developed system into a production environment, making it accessible to users. This phase encompasses several key steps to ensure a successful deployment:

1. Deployment Environment Setup:

Before deploying the system, the deployment environment must be properly configured. For your MERN chat application deployed on Vercel, you need to set up the Vercel project environment. This involves creating a Vercel account, connecting your Git repository (where your application code resides), and configuring any necessary environment variables.

2. Backend Deployment:

Since Vercel is primarily designed for hosting frontend applications, you will need to deploy your Node.js backend separately. You can host your backend on platforms like Heroku, AWS, or DigitalOcean. Once deployed, configure your frontend to communicate with the backend API endpoints.

3. Database Configuration:

Using MongoDB for database management, ensure that your database is properly configured and accessible from your backend server. If you are using MongoDB Atlas or any other cloud-based MongoDB service, configure network access settings to allow connections from your backend server's IP address.

4. Frontend Deployment:

With Vercel, deploying the front end of the MERN application is straightforward. Simply connect your Git repository containing the React frontend code to Vercel, and Vercel will automatically build and deploy your frontend application. Ensure that your fron-tend code is configured to fetch data from the correct backend API endpoints.

5. Testing and Quality Assurance:

After deployment, conduct thorough testing of both the front-end and back-end components of your application. Test for functionality, performance, and compatibility across different devices and browsers. Use tools like Postman for API testing and browser developer tools for frontend debugging.

6. Monitoring and Maintenance:

Set up monitoring tools to track the performance and availability of your deployed application. Vercel provides built-in monitoring and analytics tools that you can use to monitor your front-end application. For backend monitoring, consider using tools like New Relic or Datadog. Regularly perform maintenance tasks such as updating dependencies, optimizing database queries, and applying security patches.

By following these steps and leveraging Vercel for frontend deployment and other cloud services for backend hosting and database management, one can successfully implement and deploy their MERN chat application, providing users with a reliable and scalable communication platform.

4.3 SYSTEM TESTING

This section focuses on testing and debugging programs to achieve the system requirements. System testing evaluates the integrated system's compliance with specified requirements. It falls under black box testing, meaning testers need no knowledge of the code's inner design or logic. System testing assesses software design, behavior, and customer expectations. Hence, it's also known as the investigatory testing phase of the software development life cycle. The system testing strategies employed here include unit testing and integration testing.

4.3.1 Unit Test

The primary goal of unit testing is to take the smallest piece of testable software in the application, isolate it from the remainder of the code and determine whether it behaves exactly as it is expected to behave. Each unit is tested separately before integrating them into modules to test the interfaces between modules. Unit testing has proven its value in that a large percentage of defects are identified during its use.

4.3.2 Integration Testing

Integration testing is a logical extension of unit testing. In its simplest form, the units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. In a realistic scenario, many units are combined

into components, which are in turn aggregated into even larger parts of the program. The idea is to test combination of pieces and eventually expand the process to test your modules with those of other groups. Integration testing can be done in a variety of ways which include top-down approach, bottom-up approach and the umbrella approach.

In the integration testing of the software, satisfactory results were obtained from the test using the bottom-up approach.

4.3.3 System Testing

System testing of software or hardware is a test conducted on a completed, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic, (Sommerville, 2007)

As a rule, system testing takes, as its input, all of the "integrated" software components that have successfully passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated togetheror between any of the assemblage and the hardware. System

testing is more limiting type of testing; it seeks to detect defects both within the "inter-assemblage" and also within the system as a hole.

4.3.4 User Acceptance Test

This is the phase at which a user is introduced to the system and allowed to surf, browse, and even use the contents or components of the system to get accurate and valid feedback of excellent performance of the system. Therefore, a user was introduced to the system and allowed to browse through the system; where excellent and good feedback was gotten for the user.

This is the phase at which a user is introduced to the system and allowed to surf, browse, and even use the contents or components of the system to get accurate and valid feedback of the excellent performance of the system. Therefore, a user was introduced to the system and allowed to browse through the system; where excellent and good feedback was gotten from the user.

4.4 INTERFACE FORMS

This phase deals with the various forms of interfaces of the system. Based on this project, I designed JSX pages that provide a user with friendly interfaces to work upon. The system has two major categories of users in this project; they are Students and Staffs.

1. Login Page:

- This page allows users to log in to the system.
- It should have input fields for matriculation number or username and password.
- A "Forgot Password?" Link.

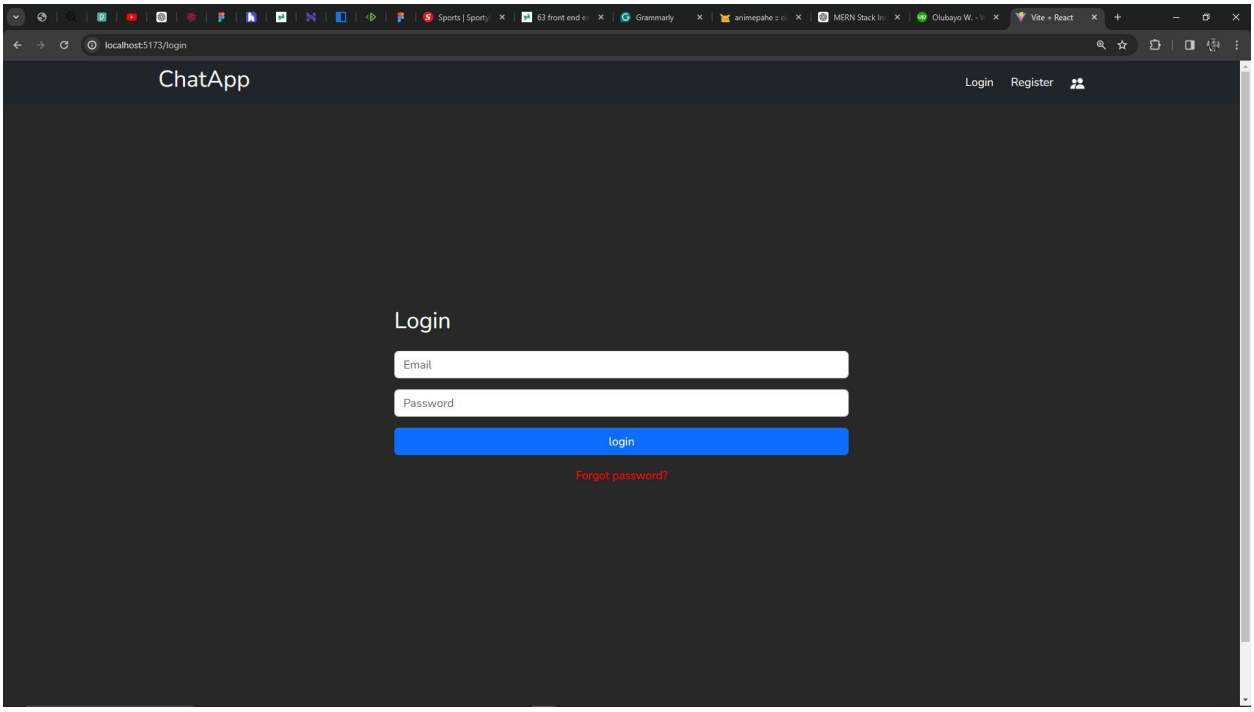

Figure 4.1: Screenshot of the Login page

2. User Registration Page:

- This page allows new users to register for the system.
- It should have input fields for name, email, password, and other required information.

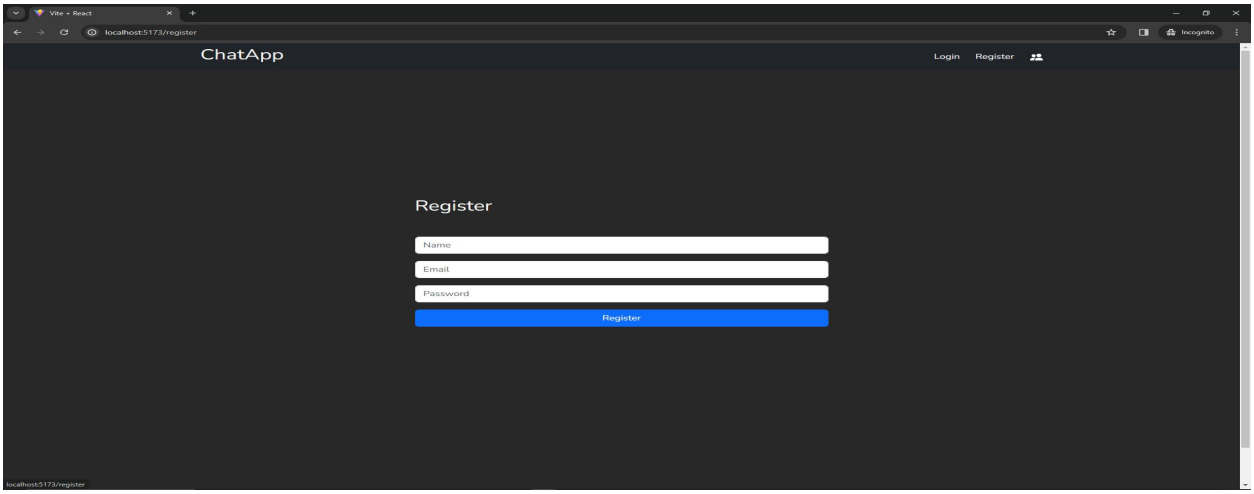

Figure 4.2: Screenshot of the Register page

3. Users' Messages / Chat Page:

- This page allows users to view their messages.
- Messages can be categorized based on sender, date, or other relevant criteria.

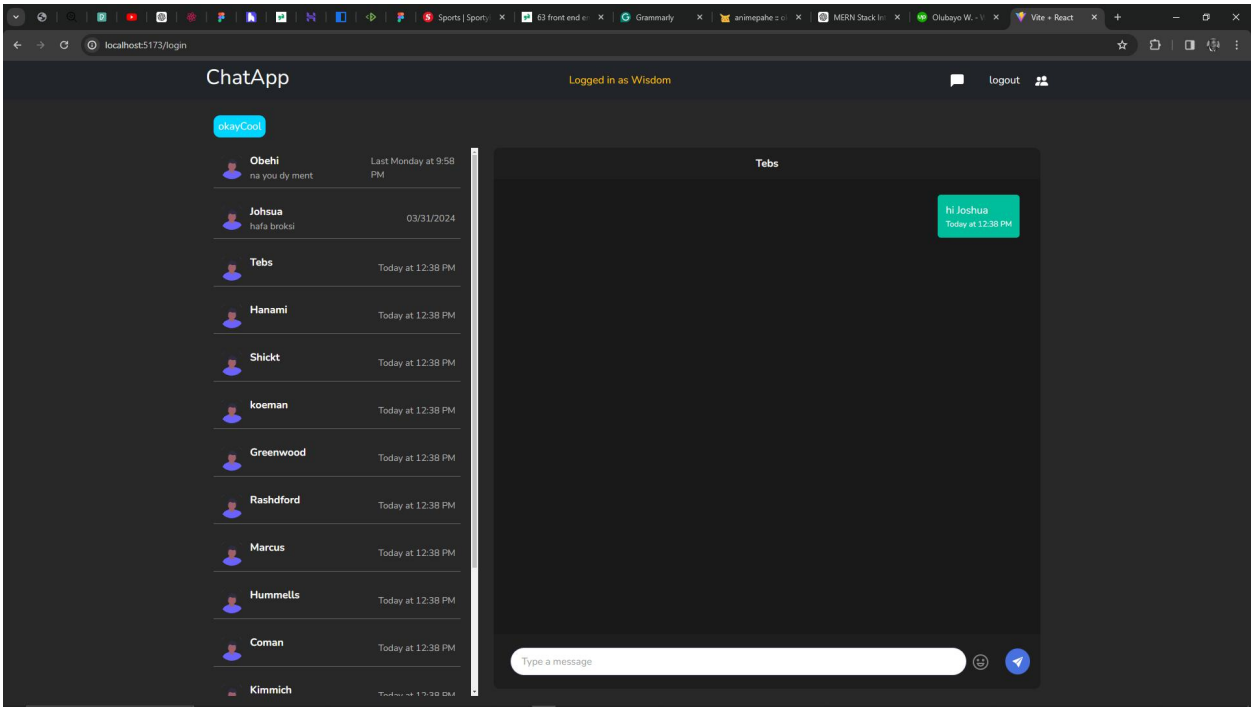

Figure 4.3: Screenshot of the chat / messages page

5. Chat Conversation Page:

- This page facilitates real-time communication between users.
- Users can send and receive messages in a chat interface.
- You can implement this using web sockets for real-time updates.

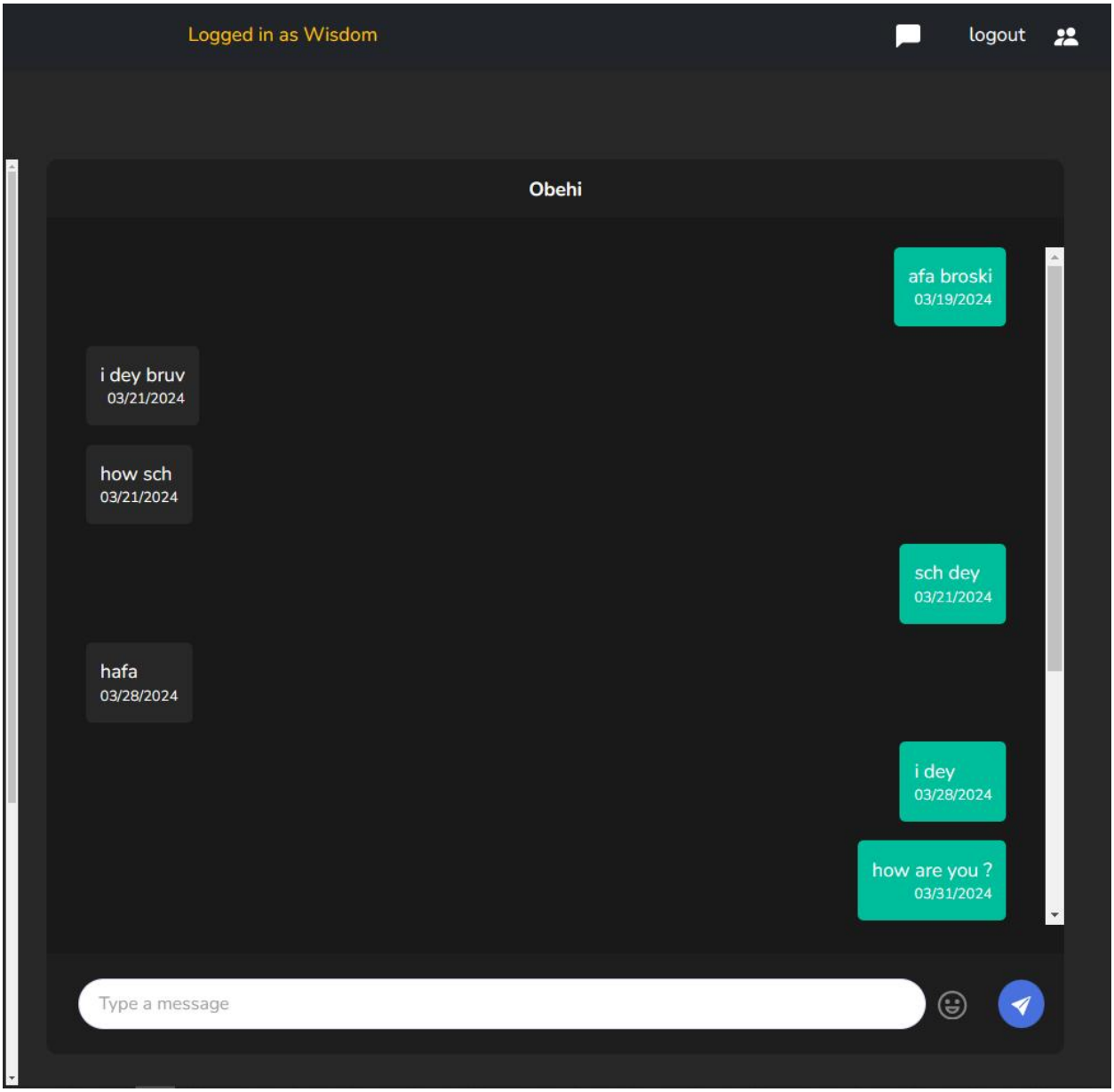

Figure 4.4: Screenshot of the conversation page

6. Group Chats Page:

- This page allows users to chat in chat rooms or groups.

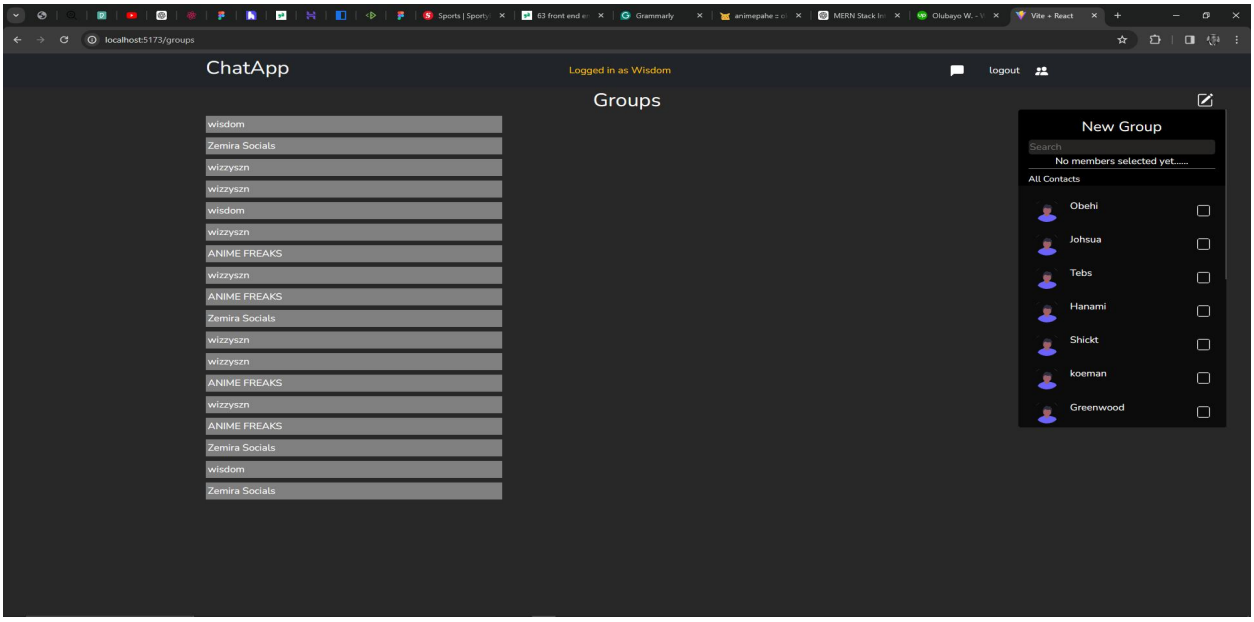

Figure 4.4: Screenshot of the Group chats page

Using the MERN stack, you'll need to implement these interfaces using React for the front end (HTML/CSS/JSX), Node.js with Express for the backend, and MongoDB for the database. You'll use React Router for navigation between pages and Axios for making HTTP requests to your backend.

CHAPTER FIVE

SUMMARY AND CONCLUSION

5.1 Summary

The MERN Chat Application for College Students is a comprehensive solution designed to address the existing communication challenges among college students. By providing a secure, efficient, and user-friendly platform for real-time messaging, group discussions, and sharing academic resources, the application aims to enhance collaboration and information sharing among students. Leveraging the MERN stack (MongoDB, Express.js, React.js, Node.js), the app ensures a modern, scalable, and robust solution that is compatible across various devices and web browsers.

The existing problem of fragmented communication channels and lack of an organized platform for students to engage in meaningful discussions and resource sharing will be effectively addressed by the MERN Chat Application. With its emphasis on security and real-time messaging, the app will create a safe and efficient environment for students to communicate and collaborate. Furthermore, the seamless compatibility across devices and web browsers will ensure that students can easily access the platform, thereby fostering a stronger sense of community and knowledge exchange.

5.2 Conclusion

In conclusion, the MERN Chat Application for College Students is poised to revolutionize the way students interact and collaborate. By offering a secure, modern, and user-friendly communication platform, the application will effectively solve the existing problem of fragmented communication channels and lack of a centralized platform for meaningful engagement. This will ultimately lead to enhanced collaboration and information sharing among college students, contributing to a more enriched academic experience.

REFERENCES

AOL. (n.d.). *AOL Instant Messenger*. Retrieved from https://cancel.aol.com/

Ashutosh,K. Atul,S.(2022). *Group Chatting Application.*
https://www.researchgate.net/publication/360483603

Avinash.B., Parikshit,B., Lineesh A. (2012). Enhanced Chat Application. Global journal of computer science and technology. https://www.semanticscholar.org/paper/Enhanced-Chat-Application-Bamane-Bhoyar/847e510c3699c235c92139b95ce3666d903723ad#citing-papers

Beynon-Davies, P. (2004). *Database systems* (3rd ed.). Palgrave Macmillan.

Booth, A., Sutton, A., & Papaioannou, D. (2016). *Systematic approaches to a successful literature review*. SAGE Publications.

Cooper, H. (2010). *Research synthesis and meta-analysis: A step-by-step approach*. SAGE Publications.

Creswell, J. W. (2014). *Research design: Qualitative, quantitative, and mixed methods approach*. SAGE Publications.

Discord. (n.d.). Retrieved May 20, 2024, from https://discord.com/

Facebook Messenger. (n.d.). Retrieved May 20, 2024, from https://www.messenger.com/

Fink, A. (2020). *Conducting research literature reviews: From the internet to paper*. SAGE Publications.

Hart, C. (2018). *Doing a literature review: Releasing the research imagination*. SAGE Publications.

IRC Help. (n.d.). IRC History. Retrieved from https://www.irchelp.org/irchelp/misc/history.html

Kline, D. (1996). *The Internet: What is it?* Que.

Lapointe, L., & Reisetter, M. (2008). The transition from face-to-face to online learning: Students' understanding of the changing expectations. *Journal of Online Learning and Teaching, 4*(2), 195-202.

Machi, L. A., & McEvoy, B. T. (2016). *The literature review: Six steps to success*. Corwin Press.

Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative research: A guide to design and implementation*. Jossey-Bass.

Noor,S.,Jamal,M. & Ban,N. (2017). Developing an End-to-End Secure Chat Application. *International Journal of Computer Science and Network Security.* http://paper.ijcsns.org/07_book/201711/20171114.pdf

Ridley, D. (2012). *The literature review: A step-by-step guide for students*. SAGE Publications.

Signal. (n.d.). Retrieved May 20, 2024, from https://signal.org/

Sommerville, I. (2007). *Software Engineering* (8th ed.). Addison-Wesley.

Sylvia, C. (2007). Uses of the internet: Then and now. *Communications of the ACM, 50*(5), 25-31.

WhatsApp. (n.d.). Retrieved May 20, 2024, from https://www.whatsapp.com/

APPENDIX

SOURCE CODE

APP Component

```
import {Route,Routes, Navigate, BrowserRouter} from 'react-router-dom'
import Chat from './pages/Chat'
import Login from './pages/Login'
import Register from './pages/Register'
import "bootstrap/dist/css/bootstrap.min.css"
import {Container} from 'react-bootstrap'
import Navbar from './components/NavBar'
import { useContext } from 'react'
import { AuthContext } from './context/AuthContext'
import { ChatContextProvider } from './context/chatContext'
import Group from './components/groupChat/Group'
import GroupChatContext from './context/groupChatContext'
function App() {
 const {user} = useContext(AuthContext)
 return (
  <ChatContextProvider user={user}>
    <GroupChatContext user ={user}>



  <BrowserRouter>
  <Navbar />
  <Container>
  <Routes>
    <Route path='/'  element={user ? <Chat />: <Login />}/>
    <Route path='/login'  element={user ? <Chat /> : <Login/>}/>
```

```
    <Route path='/register'  element={user ? <Chat /> : <Register />}/>
    <Route path='/groups'  element={<Group />}/>
    <Route path='*'  element={<Navigate to='/'/>}/>

  </Routes>
  </Container>
  </BrowserRouter>
  </GroupChatContext>
  </ChatContextProvider>
 )
}

export default App
```

Login Page

```
import { useContext, useEffect } from "react";
import { Alert,Button,Form,Row,Col,Stack } from "react-bootstrap";
import { AuthContext } from "../context/AuthContext";
import { ToastContainer, toast } from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';
// minified version is also included
// import 'react-toastify/dist/ReactToastify.min.css';

const Login = () => {
  const {loginUser, updateLoginInfo, loginInfo, loginError, isLoginLoading } = useContext(AuthContext);
  return ( <>
  <Form onSubmit={(e)=>{
     e.preventDefault();
```

```
    loginUser()


}}>
    <Row style={{
        height : "100vh",
        justifyContent : "center",
        paddingTop : "20%"
    }}>
        <Col xs={6}>
        <Stack gap={3}>
            <h2>Login</h2>
            <Form.Control type="email" placeholder="Email" onChange={(e)=>{
                updateLoginInfo({
                    ...loginInfo,
                    email : e.target.value
                })
            }}/>
            <Form.Control type="password" placeholder="Password" onChange={(e)=>{
                updateLoginInfo({
                    ...loginInfo,
                    password : e.target.value
                })


            }}/>
            <Button variant="primary" type ="submit">
                {
                    isLoginLoading ? "Getting you in .." : "login"
                }
            </Button>
```

```
            <div style={{
               color : "red",
               textAlign : 'center'
            }} >Forgot password?</div>
            {
               loginError?.error && (
                  <Alert variant="danger"> <p>{loginError.message}</p></Alert>

               )
            }

         </Stack>
         </Col>
      </Row>
   </Form>
   <ToastContainer
   position="top"
   autoClose={5000}
   hideProgressBar={false}
   newestOnTop={false}
   closeOnClick
   rtl={false}
   pauseOnFocusLoss
   draggable
   pauseOnHover
   theme="dark"
   />
   </> );
}
```

```
export default Login;
```

Chat Section

```
import { useContext } from "react";
import { ChatContext } from "../context/chatContext";
import { Container, Stack } from "react-bootstrap";
import UserChat from "../components/chat/UserChat";
import { AuthContext } from "../context/AuthContext";
import PotentialChats from "../components/chat/PotentialChats";
import ChatBox from "../components/chat/ChatBox";

function Chat() {
   const {user} = useContext(AuthContext)
   const {userChats, isUSerChatsLoading, updateCurrentChat} = useContext(ChatContext);
   return ( user && <Container>
      <PotentialChats user ={user}/>
      {
         userChats?.length < 1 ? null : <Stack direction="horizontal" gap={4} className="align-items-
start">
            <Stack className="flex-grow-0 messages-box pe-3" gap={3} style={{
               scrollbarWidth : 'thin',
               scrollbarColor: 'black',
            }} >
               {
                  isUSerChatsLoading && <p>Loading chats ....</p>
               }
               {
```

```
                userChats.map((chat ,index) => {
                    return <div key={index} onClick={() => {updateCurrentChat(chat)}}>
                        <UserChat  chat = {chat} user = {user}/>
                    </div>
                })
            }
        </Stack>
        <ChatBox />
    </Stack>
    }

     </Container>);
}

export default Chat;
```

Register page

```
import { useContext } from "react";
import { Alert,Button,Form,Row,Col,Stack } from "react-bootstrap";
import { AuthContext } from "../context/AuthContext";
const Register = () => {
  const user = useContext(AuthContext)
  const {registerInfo, updateRegisterInfo, registerUser, registerError, isRegisterLoading} =
useContext(AuthContext)
  return ( <>
  <Form onSubmit={(e)=>{
    e.preventDefault();
    registerUser()}}>
    <Row style={{
```

```
        height : "100vh",
        justifyContent : "center",
        paddingTop : "20%"
      }}>
        <Col xs={6}>
        <Stack gap={3}>
          <h2>Register</h2>
          <h1>{user.name}</h1>
          <Form.Control type="text" placeholder="Name" onChange={(e)=>{updateRegisterInfo({
            ...registerInfo,
            name : e.target.value,
          })}} />
          <Form.Control type="email" placeholder="Email" onChange={(e)=>{updateRegisterInfo({
            ...registerInfo,
            email : e.target.value
          })}}/>
          <Form.Control type="password" placeholder="Password"
onChange={(e)=>{updateRegisterInfo({
            ...registerInfo,
              password : e.target.value
          })}}/>
          <Button variant="primary" type ="submit">
          {isRegisterLoading ? "Creating your account" : "Register"}
          </Button>
          {
          registerError?.error && (<Alert variant="danger"> <p>{registerError?.message}</p></Alert>)
          }
```

```
          </Stack>
          </Col>
       </Row>
   </Form>
   </> );
}

export default Register;
```

Cascading Style Sheets (CSS)

```
@import
url('https://fonts.googleapis.com/css2?family=Josefin+Sans:wght@100;300;400;500&family=Montserrat:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;1,100;1,200;1,300;1,400;1,500;1,600;1,700&family=Nunito:ital,wght@0,200;0,300;0,400;0,700;1,300;1,700&display=swap');

body{
   color :white;
   background: rgb(40,40,40);
font-family: 'Nunito', sans-serif;
}

.all-users {
   display: flex;
   margin-bottom: 1rem;
 }

 .single-user {
   background: rgb(0, 213, 255);
   color: white;
   cursor: pointer;
   padding: 5px 7px;
```

```
  border-radius: 10px;
  margin-right: 0.5rem;
  position: relative;
}

/* User Card */
.user-card {
  color: white;
  position: relative;
  border-bottom: 1px solid rgb(100, 100, 100);
  width: 380px;
}

.user-card .text-content {
  width: 170px;
}

.user-card .name {
  font-weight: 700;
}

.user-card .text,
.user-card .date {
  font-size: 14px;
  color: rgb(180, 180, 180);
}

.user-online {
  display: inline-block;
```

```
  height: 12px;
  width: 12px;
  border-radius: 50%;
  background: rgb(0, 219, 0);
  position: absolute;
  top: -3px;
  right: -3px;
  z-index: 2;
}

/* Chat Box */
.chat-box {
  max-height: calc(100vh - 10rem);
  overflow-y: auto;
  background: rgb(25, 25, 25);
  border-radius: 10px;
}

.messages-box {
  height: 85vh;
  overflow-y: auto;
}


/* Chat Header */
.chat-header {
  display: flex;
  align-items: center;
  justify-content: center;
```

```
  padding: 0.75rem;
  color: white;
  background: rgb(30, 30, 30);
}

/* Messages */
.messages {
  max-height: calc(100vh - 15rem);
  overflow-y: auto;
  padding: 0 2rem;
}

.message {
  color: white;
  background: #282828;
  padding: 0.75rem;
  border-radius: 5px;
  max-width: 50%;
}

.message.self {
  color: white;
  background: #00bd9b;
}

.message-footer {
  font-size: 12px;
  align-self: flex-end;
  font-weight: 400;
```

```
}

/* Chat Input  */
.chat-input {
  width: 100%;
  background: rgb(30, 30, 30);
  padding: 1rem;
}
.send-btn {
  border: none;
  background: rgba(72, 112, 223, 1);
  color: white;
  height: 40px;
  width: 40px;
  border-radius: 50%;
}

/* Notification */
.notifications {
  position: relative;
}

.notifications-icon {
  color: white;
  cursor: pointer;
  position: relative;
  margin-right: 1rem;
}
```

```
.notifications-box {
  max-height: 50vh;
  width: 300px;
  overflow: auto;
  position: absolute;
  top: 2rem;
  right: 0;
  box-shadow: rgba(50, 50, 93, 0.25) 0px 2px 5px -1px,
    rgba(0, 0, 0, 0.3) 0px 1px 3px -1px;
  background: #181d31;
  color: white;
  z-index: 5;
}

.notifications-header {
  padding: 1rem;
  padding-bottom: 0;
  display: flex;
  justify-content: space-between;
}

.notifications-box h3 {
  font-weight: 700;
  font-size: 20px;
}

.mark-as-read {
  cursor: pointer;
  font-weight: 700;
```

```
  opacity: 0.8;
}

.notification {
  font-size: 14px;
  margin: 1rem ;
  padding-bottom: 0.2rem;
  border-bottom: 1px solid rgb(207, 207, 207);
  display: flex;
  flex-direction: column;
  cursor: pointer;
  padding: 0.5rem 1rem;
}

.notification.not-read {
  background: #263159;
}

.notification-time {
  margin-top: 0.2rem;
  font-size: 12px;
  color: #e0e0e0;
}
.no-convo{
  margin-right: auto;
  margin-left: auto;
}
.notification-count {
  display: flex;
```

```
  background: #00bd9b;
  height: 25px;
  width: 25px;
  font-size: 14px;
  font-weight: 700;
  border-radius: 50%;
  align-items: center;
  justify-content: center;
  position: absolute;
  top: -10px;
  right: -15px;
}

.this-user-notifications {
  display: flex;
  background: #00bd9b;
  height: 20px;
  width: 20px;
  font-size: 12px;
  font-weight: 700;
  border-radius: 50%;
  align-items: center;
  justify-content: center;
}
.this-user-notifications-off {
  display: flex;
  background: transparent;
  height: 20px;
  width: 20px;
```

```
  font-size: 12px;
  font-weight: 700;
  border-radius: 50%;
  align-items: center;
  justify-content: center;
}
/*scrollbar selector */
.scroll::-webkit-scrollbar{
  width: 0.1vw !important;
}
.scroll::-webkit-scrollbar-thumb{
  background-color: grey ;
}
.scroll::-webkit-scrollbar-track{
  background-color: inherit;
}
.scroll:hover{
  overflow: auto;
}
.scroll{
  overflow: hidden;
}

.scale:hover{
  transform: scale(1.2);
  cursor: pointer;
}

/* Customize placeholder text */
```

```
.underline-input input::placeholder {
  color: gray; /* Set placeholder text color */
  font-style: italic; /* Set placeholder text style *//* Set placeholder text opacity */
}
.people:hover{
  background-color: #222020;
}
```