**I.K.G Punjab Technical University**

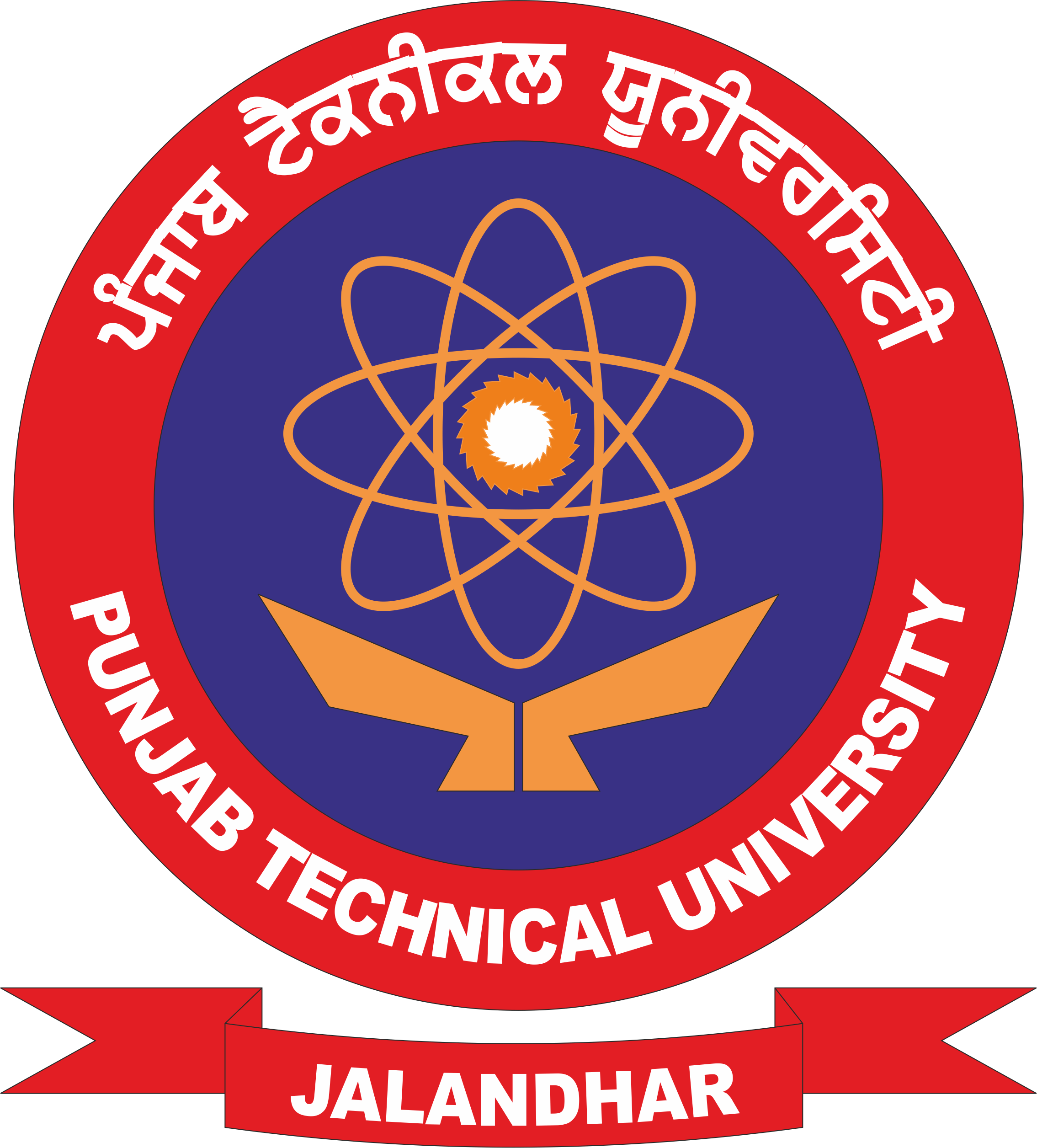
**B-Tech Batch 2021 Onward**

**Six Month Training Report**

ON

“**Chat App (Industrial Training)”**

**Submitted to**



**I.K GUJRAL PUNJAB TECHNICAL UNIVERSITY**

**KAPURTHALA**

In Partial Fulfillment of The Requirement for the Award of degree

**Of**

**Bachelor of Technology (B. Tech)**

**SUBMITTED BY**: **Submitted To:**

**NAME: Satinderpal Kaur** **Name**: **Tarun Bhalla**

**SEMESTER:** **8TH**

**ROLL NO**: **2119559**



**Computer science & Engineering**

**Anand College of Engineering & Management**

**Kapurthala**

**BATCH (2021 to 2024)**

**CERTIFICATE**

|  |
| --- |
|  |

**TO WHOM IT MAY CONCERN**

I hereby certify that Satinderpal kaur, Roll No 2119559 of B. Tech (CSE), Anand College of Engineering & Technology, has undergone six months industrial training from Future Finder at our organization to fulfill the requirements for the award of degree of B.Tech. (CSE). He worked on Social Echo project during the training under the supervision of Ayush Sharma. During his tenure with us we found him sincere and hard working. We wish him a great success in the future.

Dated: 15-01-2024 Training In-charge: Archana Sharma

**DECLARATION**

I hereby declare that the Project report titled **Chat App** is my original work and has not been published or submitted for any degree, diploma or other similar titles elsewhere. This has been undertaken for the purpose of partial fulfilment of B-Tech (Computer science engineering)

SATINDERPAL KAUR

ROLL NO:2119559

|  |
| --- |
|  |
|  |

**PREFACE**

This project report attempts to bring under one cover the entire hard work and dedication put in by me in the completion of the project work on “Chat App**”.**

I have expressed my experiences in my own simple way. I hope who goes through it will find it interesting and worth reading. All constructive feedback is cordially invited.

**ACKNOWLEDGMENT**

It is really a matter of pleasure for me to get an opportunity to thank all the persons who contributed directly or indirectly for the successful completion of the project report, “**Chat App**”.

First of all, I am extremely thankful to my college

**Anand college of Engineering & Management** for providing me with this opportunity and for all its cooperation and contribution. I am highly thankful to our respected project guide for giving me the encouragement and freedom to conduct my project.

I am also grateful to all my faculty members for their valuable guidance and suggestions for my entire study.

I would also like to thank the Future Finder team for extending their valuable time and cooperation.

Satinderpal Kaur

Roll No: 2119559

**TABLE OF CONTENTS**

|  |
| --- |
| DESCRIPTION |
| Certificate |
| Declaration |
| Acknowledgement |
| List of Tables |
| List of Figures |
| Company Profile |
| Chapter 1: Introduction Project |
| 1.1ProjectAim &Objectives |
| 1.2ProjectRequirements |
| 1.3KeyFeatureof Project |
| Chapter 2: Software and Hardware Used |
| 2.1 Software Requirement |
| 2.2 Hardware Requirement |
| Chapter 3: Results and Discussion |
| 3.1Screenshots and its Description |
| Chapter 4: Conclusion and Future Scope |
| 4.1Conclusion |
| 4.2FutureScope |
| REFERENCES |

**ABOUT COMPANY:**



Future Finders is a group of professionals who love to think outside the box. Everything we do to refine and achieve absolute excellence is our motto. We value your unbeatable talent, your innovative ideas and turn them into reality. Let us get together and head for the successful brand!

**PHILOSOPHY**

* To impart hardcore practical quality training among students/developers about latest technologies trending today.
* To share knowledge of information security and create awareness in the market. The solution to clients' as per the International standard practices and governance.
* To support good business practices through continual employee training and education
* To equip a local team with a strong knowledge of international best practices and international expert support so as to provide practical advisories in the best interests of our clients

**COMPANY’S VISION & MISSION**

FUTURE FINDERS only vision is to provide with cutting edge practical skills so that students can easily cope with and quickly adapt to the ever- changing technologies in the corporate environment. Our mission at FUTURE FINDERS is to create the highest standards in education through improvisation of quality and practical skills.

**SERVICES**

* Software Testing
* Mobile Application Testing
* [Web Development](https://futurefinders.in/web-development/)
* [Web Designing](https://futurefinders.in/web-desiging/)
* [Mobile App Development](https://futurefinders.in/mobile-app-development/)
* Embedded System Services

**Why Choose Us?**

* Hundreds of Clients & Nearly a Decade of Experience
* Goal Oriented, ROI-Driven Focus
* A Streamlined / Quality-Driven Process
* Talented Designers & Expert Developers
* Our Websites & E-marketing Platforms are Easy to Manage
* We Are Dedicated to Our Clients’ Success
* **We focus on imparting practical skills to the trainees & not just theoretical knowledge. The courses are designed in this way at FUTURE FINDERS correspond to the standards of the corporate divisions and industries. Only through the acquisition of practical skills; you can handle the everlasting technologies that venture out in real-time situations.**
* At **FUTURE FINDERS,** we have competence to expand and adjust as per client specific requirements.
* **Skilled Workforce:** At **FUTURE FINDERS,** you deal with the highly professional and proficient employees.
* **Cost Efficiency:** We help you to reduce the unnecessary investment and ask for the reasonable amount of money.
* **Quality Of the Product:** Our software service sector has been maintaining the highest international standards of quality.
* **Infrastructure:** Well organized team and tools to handle the projects with responsible approach Hardware, Software, Networking, Voice, Conferencing, disaster recovery all infra all you need for international projects.
* **Ongoing Involvement:** **FUTURE FINDERS**, products are “built for change” as we are well responsive that the necessity to improve a Web solution generally arises even before the solution is out of the door.
* **Partnership:** **FUTURE FINDERS**, considers every client a partner. From the initial stages, you are closely involved into the procedure of technical classification, development, and testing.

**KEY PROFESSIONALS**

In addition to a panel of eminent consultants and advisors, we have a dedicated pool of trained Developers and Trainer, investigators, working under the guidance of professional managers. **“A Ship is as good as the crew who sail her**.**”** Our Technical team of professionals handing, designing & delivering of projects has a strong presence in the North India & the US. Our engineers are already working on the latest technologies like **I-Phone & Android** Applications, **Robotics**, **VLSI-VHDL**, Embedded System, Networking and **Cloud computing.** Our key professionals and advisors are listed below:

**MR. BONISH SINGLA:** (**DIRECTOR**)

He is the backbone of FUTURE FINDERS, and a man with more than 9 years rich practical experience who believes in taking up new ventures and projects. He has been awarded many times for his exemplary work in process improvement for IT Service Delivery Domains. MASTERS in Computer applications and Certified from CU Certification. Holds total of 9 Years of rich experience including 5 Years in Information Security Implementation, Maintenance and Auditing and initial over 4 years of experience in Project Management, Client Relationship Management and Server, Desktop and IT Service Delivery/

**MR. AYUSH SHARMA: (FULL STACK DEVELOPER)**

He has more than 2 years’ solid industrial experience in software companies & he is very innovative in his technical approach. MCA, diploma in information technology,expertise in search engine optimization and web designing.

**MISS.HARJIT KAUR: (BRANCH MANAGER)**

She has more than 5 years solid industrial experience in software companies & is very innovative in her technical approach. MCA, Diploma in Information Technology, expertise in search engine optimization and web designing.

**MISS. ISHA BALA: (HUMAN RESOURCES)**

A very committed team leader and multi-talented person, who has been professionally attached with company.

**MISS. DIVYA: (HEAD COUNCILOR)**

She is determinate and a team player. She is good at problem solving skills and a fast learner. She can do multitasking.

**MR. SUKHBIR SINGH: (TECHNICAL HEAD)**

He has more than 3 years of experience in PLC and Scada.). He has been awarded many times for his brilliant services.

## MR. JASPAL SINGH: (CIVIL &MECHANICAL HEAD)

He has more than 38years of experience in industrial field. He is providing services as a technical trainer for more than 8 years. He did his B. Tech in Mechanical Engineering from PEC (Punjab Engineering College). He has been awarded many times for his brilliant services.

## MR. CHETAN KALRA: (DIGITAL MARKETING HEAD)

B. Tech (CSE) – IKG-PTU, Expertise in Python Programming, Full Stack Development, Presenting ideas for web development software, including Application software, Working closely with analysts, Senior Developers, Programmers, designers and staff, Producing detailed specifications and writing the program codes, Testing the product in controlled, real situations before going live, Preparation of training manuals for users and Maintaining the systems once they are up and running.

**Lists of Figures**

|  |  |
| --- | --- |
| DESCRIPTION | FIGURE NO. |
| HTML Logo | Figure 2.1 |
| CSS Logo | Figure 2.2 |
| JavaScript Logo | Figure 2.3 |
| React Logo | Figure 2.4 |
| Node. JS Logo | Figure 2.5 |
| MongoDB Logo | Figure 2.6 |
| Visual Studio Code Logo | Figure 2.7 |
| VS Code Interface | Figure 2.8 |
| Signup and login page | Figure 3.1 |
| login page | Figure 3.2 |
| Signup | Figure 3.3 |
| Community Post | Figure 3.4 |
|  |  |

**CHAPTER-1**

**INTRODUCTION TO PROJECT**

# 1.1 Project Aim & Objectives

Project Aim:

Welcome to [Chat App], your go-to platform for seamless and secure chatting! Whether you're catching up with friends, collaborating with colleagues, or making new connections, [Chat App] is designed to enhance your communication experience.

With its user-friendly interface and robust features, [Chat App Name] offers a diverse range of functionalities to cater to your messaging needs. From individual chats to group discussions, from text messages to multimedia sharing, [Chat App Name] ensures that you stay connected with ease.

But [Chat App Name] is more than just a messaging platform. We prioritize your privacy and security, employing state-of-the-art encryption protocols to safeguard your conversations. Your data remains confidential, and you have full control over your account settings.

Additionally, [Chat App Name] is built for versatility. Whether you're accessing it on your smartphone, tablet, or desktop, [Chat App Name] seamlessly syncs your conversations across devices, ensuring you never miss a beat.

Join millions of users worldwide who trust [Chat App Name] for their chatting needs. Download the app today and experience a new era of communication.

Stay connected, stay secure, with [Chat App].

Top of Form

**Literature Review**

A literature survey for a chat app focused on chatting, you'll want to explore various aspects related to messaging platforms, user experience, technology, security, and more. Here's a general outline of what you might include in your literature survey:

1. **Messaging Platforms and User Experience**:
   * Explore existing messaging platforms such as WhatsApp, Messenger, Telegram, Signal, We Chat, etc.
   * Analyze user experiences, including interface design, ease of use, and user engagement strategies.
   * Examine the features that contribute to user satisfaction and retention.
2. **Technological Aspects**:
   * Investigate the underlying technologies powering messaging apps, such as real-time messaging protocols, push notifications, and cloud infrastructure.
   * Review advancements in chat app development, including cross-platform compatibility, integration with APIs, and support for multimedia messaging.
3. **Security and Privacy**:
   * Investigate security measures employed by existing messaging platforms to protect user data and communications.
   * Review encryption standards (e.g., end-to-end encryption) and authentication mechanisms.
   * Examine privacy policies and user data handling practices.
4. **User Behavior and Adoption**:
   * Explore studies on user behavior in messaging apps, including messaging frequency, preferred features, and communication patterns.
   * Analyze factors influencing the adoption of messaging apps among different demographics and regions.
5. **Business Models and Monetization**:
   * Investigate various monetization strategies employed by messaging platforms, such as subscription models, in-app purchases, advertising, and enterprise solutions.
   * Review case studies of successful monetization approaches in the messaging app industry.
6. **Emerging Trends and Future Directions**:
   * Explore emerging trends in messaging apps, such as chatbots, voice messaging, integration We Chat, etc.
   * Analyze user experiences, including interface design, ease of use, and user engagement strategies.
   * Examine the features that contribute to user satisfaction and retention.
7. **Technological Aspects**:
   * Investigate the underlying technologies powering messaging apps, such as real-time messaging protocols, push notifications, and cloud infrastructure.
   * Review advancements in chat app development, including cross-platform compatibility, integration with APIs, and support for multimedia messaging.
8. **Security and Privacy**:
   * Investigate security measures employed by existing messaging platforms to protect user data and communications.
   * Review encryption standards (e.g., end-to-end encryption) and authentication mechanisms.
   * Examine privacy policies and user data handling practices.
9. **User Behavior and Adoption**:
   * Explore studies on user behavior in messaging apps, including messaging frequency, preferred features, and communication patterns.
   * Analyze factors influencing the adoption of messaging apps among different demographics and regions.
10. **Business Models and Monetization**:
    * Investigate various monetization strategies employed by messaging platforms, such as subscription models, in-app purchases, advertising, and enterprise solutions.
    * Review case studies of successful monetization approaches in the messaging app industry.
11. **Emerging Trends and Future Directions**:
    * Explore emerging trends in messaging apps, such as chatbots, voice messaging, integration with virtual assistants, and augmented reality features.
    * Discuss potential future developments in messaging technology and user experience.
12. **Regulatory and Ethical Considerations**:
    * Review regulatory frameworks relevant to messaging apps, such as data protection laws, telecommunications regulations, and censorship policies.
    * Consider ethical implications related to user privacy, content moderation, and platform responsibility.
13. **Comparative Analysis**:
    * Conduct a comparative analysis of existing messaging platforms based on key criteria such as security, user experience, features, and business models.
14. **Gap Analysis**:
    * Identify gaps or opportunities for innovation in the existing landscape of messaging apps, based on the literature review.
15. **Conclusion**:
    * Summarize key findings from the literature survey.
    * Highlight potential areas for further research or development in the context of your chat app project.

**CHAPTER -2**

**HARDWARE AND SOFTWARE USED**

**2.1. Hardware Requirements:**

The development of the Chat App Website necessitates specific hardware components to ensure optimal performance and functionality. The essential hardware prerequisites for this project are as follows:

X86 or 64-bit CPU (Intel / AMD architecture): A central processing unit with x86 or 64-bit architecture, preferably from Intel or AMD, is a fundamental requirement to support the execution of software and processing of data for the CHAT APP Website.

* + 1. GB RAM: To facilitate smooth multitasking and efficient handling of data, a minimum of 4 gigabytes of access memory (RAM) is essential. This allows the system to run resource-intensive applications with ease.

* + 1. GB free disk space: Adequate storage space is crucial to accommodate the development tools, code repositories, and data files associated with the Chat App project. A minimum of 5 gigabytes of free disk space ensures ample room for these requirements.

**2.2. Software Requirements:**

The software stack for the development of the CHAT APP encompasses a range of tools and technologies. These software components are essential for creating a robust and functional website. The software requirements include

Modern Operating System:

1. Windows 8.1 or 10: The website development environment is compatible with modern Windows operating systems, ensuring a seamless experience for developers using this platform.
2. Mac OS X 10.11 or higher, 64-bit: Mac users are accommodated with compatibility for Mac OS X versions 10.11 and above, specifically tailored for 64-bit architecture

VS Code: Visual Studio Code (VS Code) is a versatile and user-friendly source code editor. It offers a conducive development environment for writing, editing, and debugging code, enhancing productivity throughout the web development process.

The combination of these hardware and software requirements forms the foundation for the successful development of the CHAT APP.

# HTML5 (Hyper Text Markup Language-5)



# Figure 2.1 (HTML Logo)

HTML5 is a markup language used for structuring and presenting content on the World Wide

Web. It is the fifth and current version of the HTML standard.

It was published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc. HTML5 is intended to subsume not only HTML 4 but also XHTML 1 and DOM Level 2 HTML.

HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves, and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a candidate for cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

Many new syntactic features are included. To natively include and handle multimedia and graphical content, the new<video>, <audio>, and <canvas> elements were added, and support for scalable vector graphics (SVG) content and MathML for mathematical formulas. To enrich the semantic content of documents, new page structure elements such as <main>, <section>, <article>, <header>, <footer>, <aside>, <nav>, and <figure>, are added. New attributes are introduced, some elements and attributes have been removed, and others such as<a>, <cite>, and <menu>have been changed, redefined, or standardized. The APIs and Document Object Model (DOM) are now fundamental parts of the HTML5 specification and HTML5 also better defines the processing for invalid documents.

**CSS3 (Cascading Style Sheet)**



Figure 2.2 (CSS Logo)

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG, and XUL, and applies to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology most websites use to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.

CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share formatting by specifying the relevant CSS in a separate .CSS file, and reduce complexity and repetition in the structural content. Separation of formatting and content makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. It can also display the web page differently depending on the screen size or viewing device.

# JavaScript

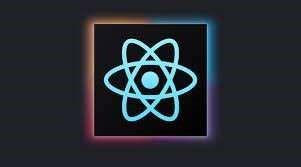


# Figure 2.3 (JavaScript Logo)

JavaScript, often abbreviated as JS, is a high-level, dynamic, weakly typed, object-based, and interpreted client-side programming language. Alongside HTML and CSS, JavaScript is one of the three core technologies of World Wide Web content production. It is used to make webpages interactive and provide online programs, including video games. The majority of websites employ it, and all modern web browsers support it without the need for plug-ins using a built-in JavaScript engine.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has an API for working with text, arrays, dates, regular expressions, and basic manipulation of the DOM, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded. Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, in non-web programs such as word processors and software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets

**REACT**



# Figure 2.4(React Logo)

ReactJS is a JavaScript library used for building reusable UI components.

According to React official documentation, following is the definition −

React is a library for building composable user interfaces. It encourages the creation of reusable UI components, which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native. React implements a one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.

React Features

JSX − JSX is a JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended. Components − React is all about components. You need to think of everything as a component. This will help you maintain the code when working on large-scale projects. Unidirectional data flow and Flux−React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keep your data unidirectional. License − React is licensed under Facebook Inc. Documentation is licensed under CC BY 4.0.

React Advantages

Uses virtual DOM which is a JavaScript object. This will improve app performance since JavaScript virtual DOM is faster than the regular DOM. Can be used on the client and server side as well as with other frameworks.

Component and data patterns improve readability.

# NODE.js

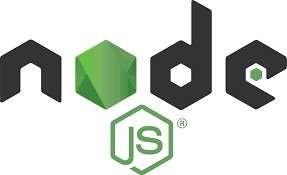


Figure 2.5 Node.js Logo)

Node.js, a groundbreaking server-side platform introduced by Ryan Dahl in 2009, has undergone continuous evolution, with its latest version, v0.10.36, showcasing advancements in performance and functionality. The core essence of Node.js, as articulated in its official documentation, revolves around harnessing Chrome's JavaScript runtime to empower developers in building high-performance and scalable network applications with ease.

Its event-driven architecture, coupled with a non-blocking I/O model, epitomizes Node.js' lightweight and resource-efficient nature, making it a preferred choice for developing real-time applications across distributed systems. Beyond its technical prowess, Node.js stands as an emblem of collaboration and innovation within the open-source community, with contributions from developers worldwide enriching its ecosystem and expanding its capabilities.

As an adaptable and cross-platform runtime environment, Node.js provides developers with a versatile toolkit for crafting server-side and networking applications tailored to diverse use cases. By leveraging JavaScript—a ubiquitous language in web development—Node.js simplifies the development process and accelerates time-to-market for applications spanning various domains, from e-commerce platforms to social networking services.

With seamless compatibility across major operating systems—including OS X, Microsoft Windows, and Linux—Node.js facilitates the creation of robust and scalable applications that cater to the evolving needs of modern enterprises. Its versatility, combined with a vibrant community and robust ecosystem, positions Node.js as a cornerstone of innovation in the realm of server-side.

# MongoDB



Figure 2.6 (MongoDB Logo)

Like any other database management language, MongoDB is founded on a NoSQL database model, which efficiently stores data in a key-value pair. Its functionality revolves around the concept of documents and collections, making it highly versatile for various data storage needs. MongoDB is renowned for its open-source nature, document-oriented architecture, and cross-platform compatibility, making it accessible and adaptable for a wide range of applications.

Developed by 10gen in October 2007, MongoDB has garnered widespread adoption due to its high availability, robust performance, and automatic scaling capabilities. It is licensed under the General Public License (GPL) and is also available under a Commercial license from the manufacturer, offering flexibility in usage. MongoDB was specifically designed to operate seamlessly with commodity servers, catering to the needs of companies of all sizes and industries worldwide.

MongoDB's scalability and flexibility make it an ideal choice for businesses looking to manage large volumes of data efficiently. Its ability to handle diverse data types and support for complex queries make it well-suited for modern applications that require agile and responsive database solutions. With a vibrant community of developers and extensive documentation, MongoDB continues to evolve and adapt to meet the ever-changing demands of the digital landscape.

Its schema-less design allows for seamless integration with evolving data structures, enabling developers to iterate quickly and adapt to changing business requirements. Additionally, MongoDB's support for geospatial queries and indexing further extends its utility across a wide range of use cases, from geolocation-based applications to data analytics platforms.

# Visual Studio Code



# Figure 2.7 (VS Code Logo)

Visual Studio Code (famously known as VS Code) is a free open-source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times.

VS Code supports a wide array of programming languages from Java, C++, and Python to CSS, Go, and Docker file. Moreover, VS Code allows you to add on and even create new extensions including code linkers, debuggers, and cloud and web development support.

The VS Code user interface allows for a lot of interaction compared to other text editors. To simplify user experience, VS Code is divided into five main regions:

* The activity bar in Visual Studio Code provides quick access to various features and functionalities, including navigating between different views such as Explorer, Search, Source Control, and Extensions.
* The sidebar, located on the left-hand side of the interface, displays additional navigation options, such as file explorer, search, and source control, allowing users to manage their projects efficiently.
* Editor groups in Visual Studio Code allow users to work with multiple

files simultaneously by splitting the editor window into different sections, enabling seamless multitasking and code comparison.

* The panel, situated at the bottom of the interface, offers access to additional tools and features such as output, debug console, and integrated terminal, enhancing productivity and workflow management.
* The status bar provides essential information such as file encoding, line endings, and Git status, keeping users informed about the current state of their projects and enabling quick actions.
* The integrated terminal feature in Visual Studio Code allows developers to execute commands, run scripts, and interact with the command-line interface directly within the editor, streamlining the development process and facilitating tasks such as debugging and testing.

The image below shows how these regions are displayed:

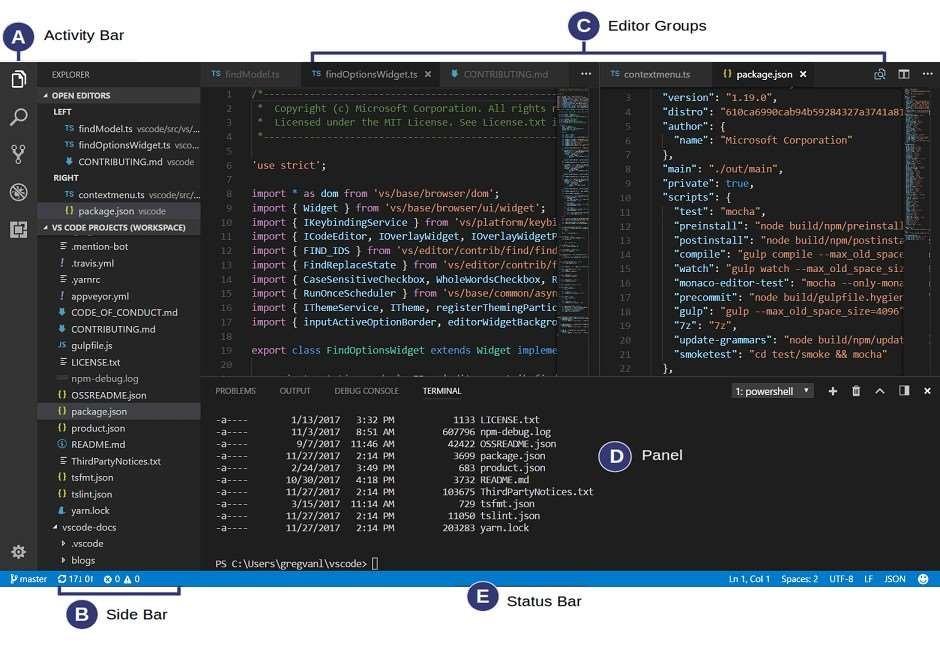


Figure 2.8 (VS Code Interface)

After installation, you can run the editor by entering the code -n command into the terminal. This will prompt VS Code to start and display a fresh instance. If you want to continue working from where you left off in the previous coding session, enter the code command without the -n flag. If you want to open VS Code in a certain directory, navigate to the directory and enter the code -r command. You are now on your way to writing powerful code in your VS Code environment

**CHAPTER 3**

**RESULT AND DISCUSSION**

**RESULT:-**

* To provide you with a specific result for a "Chat App” project built with the MERN stack (MongoDB, Express.js, React.js, Node.js), I'll outline the typical features and components you might include:
* **Authentication**: Implement user authentication and authorization, allowing users to sign up, log in, and securely access their accounts. You can use libraries like Passport.js or JSON Web Tokens (JWT) for this.
* **User Profiles**: Allow users to create profiles, update their information, and upload a profile picture. This could include basic information like name, bio, and contact details.
* **Post Creation and Viewing**: Implement functionality for users to create, edit, and delete posts. Users should also be able to view posts created by themselves and others. Posts can include text, images, or multimedia content.
* **Comments and Likes**: Enable users to comment on posts and like or react to them. Real-time updates for likes and comments can enhance the user experience.
* **News Feed**: Design a news feed that displays posts from users the current user follows. This involves fetching and displaying posts in a chronological order.
* **Friend/Follow System**: Allow users to follow or befriend other users. Implement functionality to view followers and followed users, and to manage friend requests.
* **Notifications**: Implement a notification system to notify users about new followers, likes, comments, or other relevant activities.
* **Messaging**: Optionally, implement a messaging system to enable private communication between users.
* **Search Functionality**: Provide users with the ability to search for other users or specific posts based on keywords or tags.
* **Responsive Design**: Ensure the application is responsive and works well on various devices, including desktops, tablets, and smartphones.
* For the backend, you would use Node.js with Express.js for creating RESTful APIs to handle data manipulation and authentication. MongoDB would serve as the database to store user information, posts, comments, etc.
* On the frontend, you'd use React.js to build the user interface, manage state, and interact with the backend APIs. Libraries like Redux or Context API can be used for state management, and frameworks like Material-UI or Bootstrap can aid in designing the UI components.
* To get started, you might want to set up your development environment, design the database schema, and then gradually implement each feature while testing along the way.

**DISCUSSION:-**

Certainly! Let's discuss the potential aspects and considerations for a "Chat App" project built with the MERN stack.

1. **Purpose and Audience**: Before diving into development, it's crucial to define the purpose of the Social Echo platform and understand the target audience. Is it a platform for general social networking, focused on a specific niche, or aimed at a particular demographic? Understanding the target audience will help in designing features and user experience tailored to their needs.
2. **Feature Set**: Based on the purpose and audience, you'll need to determine the core features to include in the platform. These may include user authentication, profile management, post creation and interaction (likes, comments, shares), news feed, messaging, friend/follow system, notifications, search functionality, and possibly more advanced features like groups or events.
3. **Database Design**: With MongoDB as the database in the MERN stack, you'll need to design a schema that efficiently stores user data, posts, comments, relationships (followers/following), and other relevant information. Consider the relationships between different entities and how they will be queried and accessed.
4. **Authentication and Security**: Implement robust authentication mechanisms to ensure user accounts are secure. This may involve using JWT tokens, OAuth, or other authentication methods. Additionally, enforce proper authorization to restrict access to certain features or data based on user roles and permissions.
5. **Real-time Functionality**: Adding real-time capabilities can enhance the user experience by providing instant updates on likes, comments, and notifications. Technologies like Web Sockets or libraries such as Socket.io can be used to implement real-time features efficiently.
6. **Scalability and Performance**: Plan for scalability from the outset, considering factors such as database indexing, caching, and load balancing. As the platform grows, it should be able to handle increasing numbers of users and data without sacrificing performance.
7. **User Interface and Experience**: Design an intuitive and visually appealing user interface using React.js. Consider factors such as responsiveness, accessibility, and ease of navigation to ensure a positive user experience across devices and screen sizes.
8. **Testing and Quality Assurance**: Implement thorough testing practices, including unit tests, integration tests, and end-to-end tests, to ensure the reliability and stability of the application. Continuous integration and deployment (CI/CD) pipelines can help automate the testing and deployment process.
9. **Community Management**: Consider how you will manage user-generated content, moderate discussions, and handle user feedback and reports. Implementing moderation tools and community guidelines can help maintain a positive and safe environment for users.
10. **Legal and Compliance**: Be aware of legal considerations such as data privacy regulations (e.g., GDPR), terms of service, and copyright issues. Ensure that the platform complies with relevant laws and regulations in the regions where it operates.

By carefully considering these aspects and planning the development process accordingly, you can create a successful and engaging CHAT APP platform in the MERN stack.

**Specifications:**

Font Style: Times New Roman

Chapter Name: 16pts, bold, Block Letters

Font Size: Headings (14pts. Bold)

Sub Headings: (12pts. Bold)

Text: 12pts.

Content: Justify

Chapter Name: Centre, Bold, Block Letters

Margins: Top: 1

Bottom: 1

Left: 1.7

Right: 1

Line Spacing: 1.15

IMPORTANT

Mention: Figure Number and Name

Table Number and Name

Note: Figures and Tables should be canter Align

Insert page number on Bottom Left Side

Sample Figure



dashboard.

**Table of Contents**

* [Project Overview](https://github.com/nz-m/SocialEcho#project-overview)
* [Features](https://github.com/nz-m/SocialEcho#features)
* [Technologies](https://github.com/nz-m/SocialEcho#technologies)
* [Schema Diagram](https://github.com/nz-m/SocialEcho#schema-diagram)
* [Getting Started](https://github.com/nz-m/SocialEcho#getting-started)
* [Usage](https://github.com/nz-m/SocialEcho#usage)
* [License](https://github.com/nz-m/SocialEcho#license)

**Project Overview**

The project is a social networking platform built using the MERN (MongoDB, Express.js, React.js, Node.js) stack. It incorporates two major features: an automated content moderation system and context-based authentication. These features are accompanied by common functionalities found in social media applications, such as profile creation, post creation and sharing, liking and commenting on posts, and following/unfollowing users.

**Automated Content Moderation**

The platform's automated content moderation system utilizes various NLP (Natural Language Processing) APIs. These APIs include:

* Perspective API: Used for filtering spam, profanity, toxicity, harassment etc.
* Text Razor API: Integrated for content categorization.
* Hugging Face Interface API: Utilized with BART Large MNLI for content categorization.

A Flask application has been developed to provide similar functionality as the Hugging Face Interface API's classifier. The Flask app utilizes the BART Large MNLI model. It operates as a zero-shot classification pipeline with a Pay Torch framework.

The system allows flexibility in choosing different services for API usage or disabling them without affecting overall functionality by using a common interface for interacting with the APIs.

When a user posts content, it undergoes a thorough filtering process to ensure compliance with the community guidelines. Additionally, users have the ability to report posts that they find inappropriate, which triggers a manual review process.

**Context-Based Authentication**

The platform implements context-based authentication to enhance user account security. It takes into consideration user location, IP address, and device information for authentication purposes. Users can conveniently manage their devices directly from the platform. To ensure data privacy, this information is encrypted using the AES algorithm and securely stored in the database.

In case of a suspicious login attempt, users are promptly notified via email and are required to confirm their identity to protect against unauthorized access.

**User Roles**

There are three distinct user roles within the system:

1. Admin: The admin role manages the overall system, including moderator management, community management, content moderation, monitoring user activity, and more.
2. Moderators: Moderators manage communities, manually review reported posts, and perform other moderation-related tasks.
3. General Users: General users have the ability to make posts, like comments, and perform other actions within the platform.

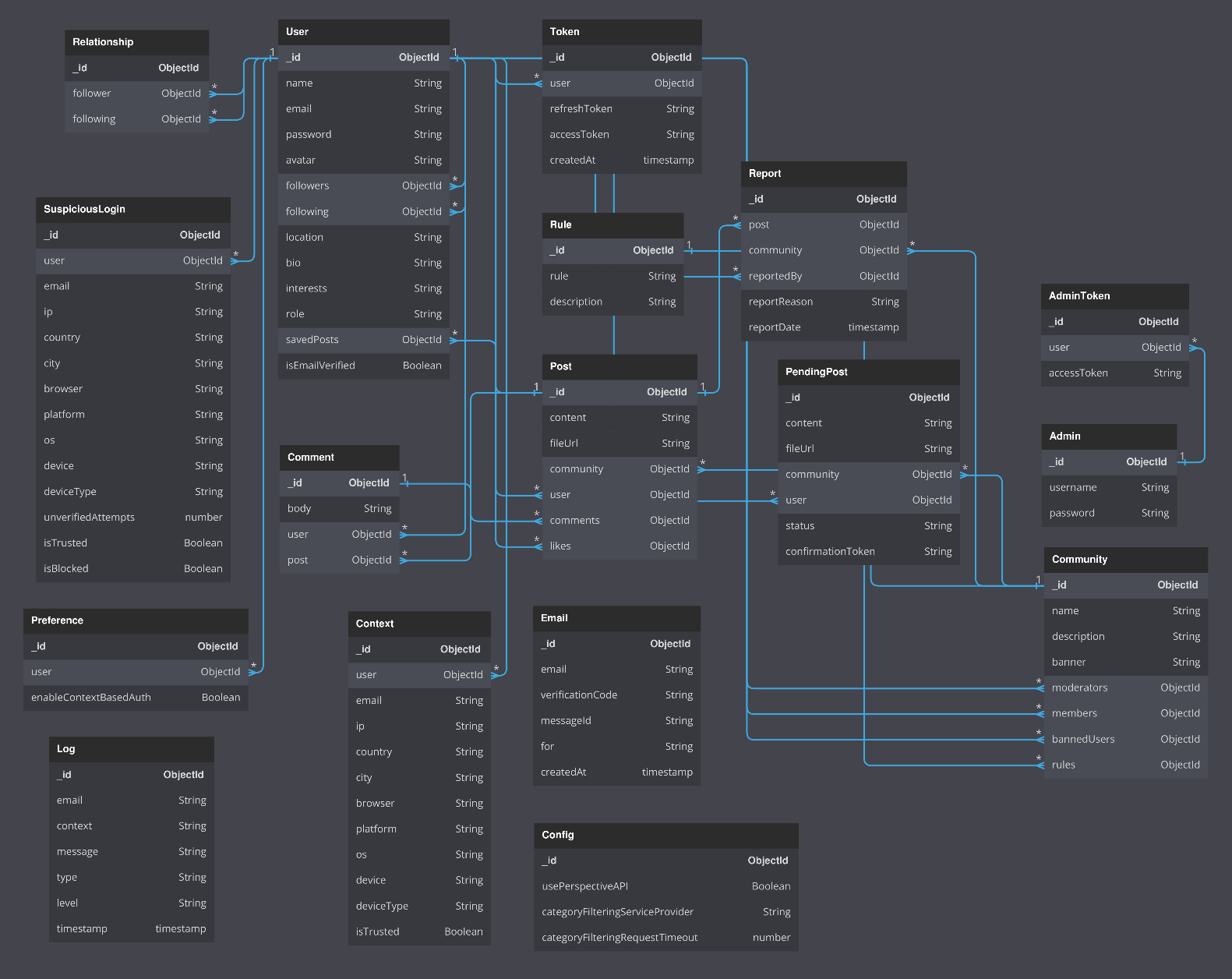
**Features**

* User authentication and authorization (JWT)
* User profile creation and management
* Post creation and management
* Commenting on posts
* Liking posts and comments
* Following/unfollowing users
* Reporting posts
* Content moderation
* Context-based authentication
* Device management
* Admin dashboard
* Moderator dashboard
* Email notifications

**Technologies**

* React.js
* Redux
* Node.js
* Express.js
* MongoDB

**Schema Diagram**



**Getting Started**

**Prerequisites**

Before running the application, make sure you have the following installed:

* Node.js
* MongoDB or MongoDB Atlas account

**Installation**

1. Go to the project directory and install dependencies for both the client and server

cd client

npm install

cd server

npm install

1. Create a .env file in both the client and server directories and add the environment variables as shown in the. env. example files.
2. Start the server

cd server

npm start

1. Start the client

cd client

npm start

**Configuration**

Run the admin\_tool.sh script from the server directory with permissions for executing the script. This script is used for configuring the admin account, creating the initial communities, and other settings.

./admin\_tool.sh

**env Variables**

For email service of context-based authentication, the following variables are required:

EMAIL=

PASSWORD=

EMAIL\_SERVICE=

For content moderation, you need the PERSPECTIVE\_API\_KEY and either the INTERFACE\_API\_KEY or TEXTRAZOR\_API\_KEY. Visit the following links to obtain the API keys:

* [Perspective API](https://developers.perspectiveapi.com/s/docs-get-started)
* [Text Razor API](https://www.textrazor.com/)
* [Hugging Face Interface API](https://huggingface.co/facebook/bart-large-mnli)

If you prefer, the Flask server can be run locally as an alternative to using the Hugging Face Interface API or Text Razor API. Refer to the classifier server directory for more information.

**Note:** Configuration for context-based authentication and content moderation features are ***not mandatory*** to run the application. However, these features will not be available if the configuration is not provided.

**Usage**

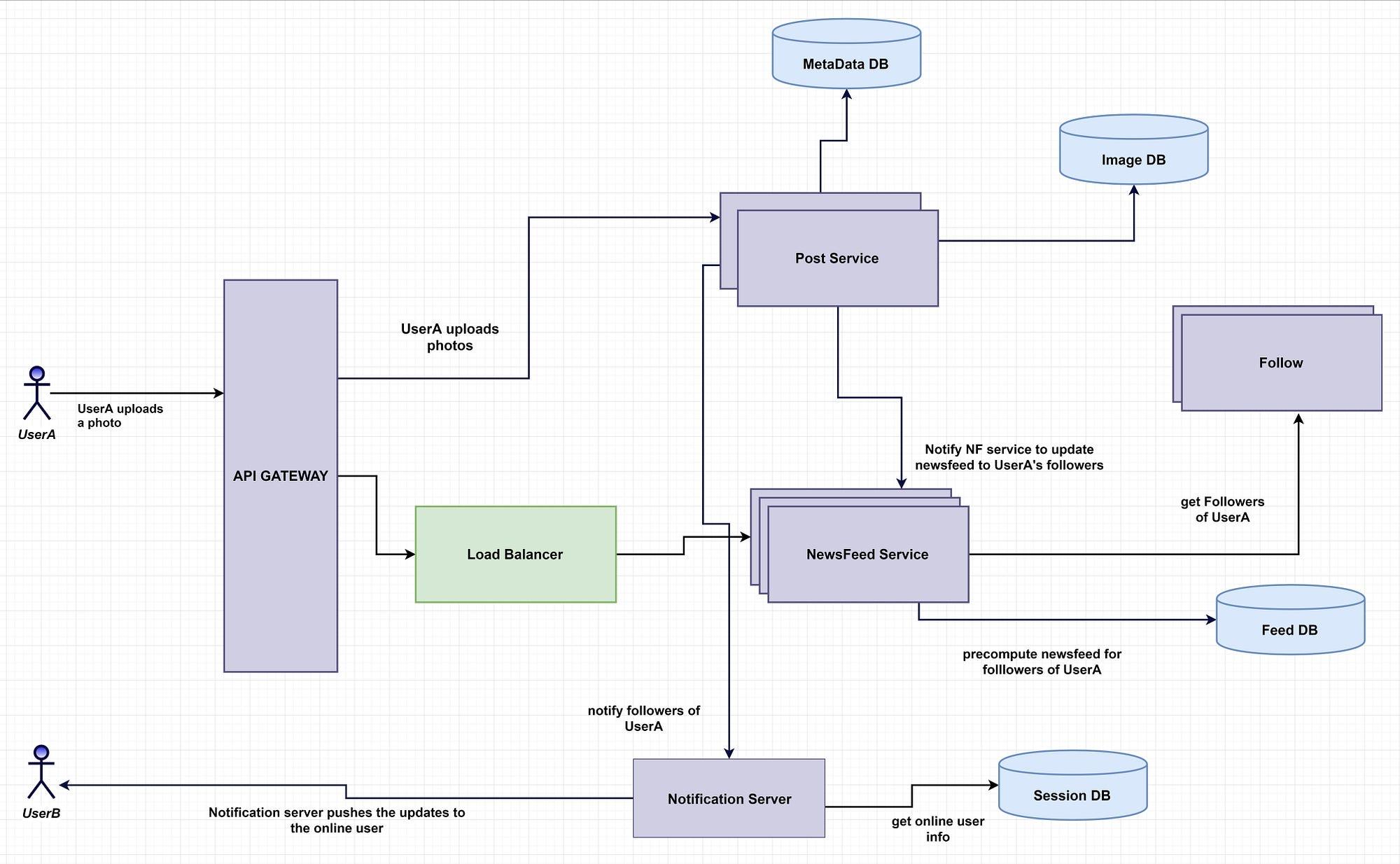
**Admin**

The admin dashboard can be accessed at the /admin route. Use the admin\_tool.sh script to configure the admin account. The admin account can be used to manage moderators, communities, and perform other admin-related tasks. You can also enable/disable or switch API services using the admin dashboard.

**Moderator**

Moderators have specific email domain (@mod.socialecho.com). When registering with an email from this domain, the user is automatically assigned the moderator role. Moderators can be assigned to different communities from the admin.

**SYSTEM DESIGN ANALYSIS**



**EXISTING & PROPOSED SYSTEM**

1. Existing System:
   * If there's an existing system, it might lack certain features or have limitations in terms of scalability, performance, or user experience.
   * The existing system may have been built using different technologies or frameworks, which could pose challenges if migrating to the MERN stack.
   * Users might have provided feedback or encountered issues with the current system, which could inform the improvements needed in the proposed system.
2. Proposed System:
   * The proposed system in the MERN stack aims to address the limitations of the existing system and provide a more robust, scalable, and feature-rich platform.
   * It will include all the core features discussed earlier, such as user authentication, profile management, post creation and interaction, real-time updates, messaging, and more.
   * The proposed system will leverage the MERN stack's strengths, such as the flexibility of MongoDB for storing diverse data types, Express.js for building scalable APIs, React.js for creating dynamic user interfaces, and Node.js for handling server-side logic.
   * The user interface will be redesigned using React.js to improve usability, responsiveness, and aesthetics. It will offer a seamless and intuitive experience across devices.
   * Security measures will be enhanced, including robust authentication mechanisms, encryption of sensitive data, and proper authorization controls to protect user privacy and prevent unauthorized access.
   * The proposed system will focus on scalability, with optimizations for database performance, efficient querying, and horizontal scaling to accommodate growing user bases and data volumes.
   * Testing will be an integral part of the development process, ensuring the reliability, stability, and quality of the system through comprehensive unit tests, integration tests, and end-to-end tests.
   * Continuous integration and deployment pipelines will be set up to automate the testing, build, and deployment processes, facilitating faster iteration and deployment of new features and updates.
   * Community management tools will be integrated to facilitate moderation, handle user feedback and reports, and maintain a positive and safe environment for users.
   * Legal and compliance considerations will be addressed to ensure that the proposed system complies with relevant laws and regulations regarding data privacy, terms of service, and copyright.

Overall, the proposed system aims to provide a significantly improved social networking platform compared to the existing system, offering enhanced features, performance, security, and user experience**.**

**FUTURE SCOPE OF THE PROJECT**

**Advanced AI Integration**

One of the most promising areas for the future development of the chat app project lies in the integration of advanced artificial intelligence (AI) technologies. By leveraging AI capabilities such as natural language processing (NLP), machine learning, and predictive analytics, the platform can offer a range of intelligent features to enhance user engagement and satisfaction.

**Sentiment Analysis:** Implementing sentiment analysis algorithms can enable the platform to analyse the tone and emotion conveyed in user-generated content, such as posts and comments. This analysis can provide valuable insights into user sentiment trends, helping to identify popular topics, emerging issues, and areas of interest within the community. Additionally, sentiment analysis can be used to automatically flag and moderate inappropriate or offensive content, maintaining a positive and respectful environment for all users.

**Content Recommendation Systems:** Utilizing machine learning algorithms, the platform can develop personalized content recommendation systems tailored to individual user preferences and interests. By analysing user behaviour, engagement patterns, and content interactions, the system can suggest relevant posts, articles, groups, or events that align with each user's unique preferences. This personalized approach not only enhances user satisfaction but also increases user retention and engagement by delivering content that resonates with their interests.

**Chatbots and Virtual Assistants**: Introducing AI-powered chatbots and virtual assistants can streamline user interactions, provide instant support, and enhance the overall user experience. Chatbots can assist users with common queries, such as account inquiries, feature explanations, or troubleshooting assistance, reducing the burden on human support agents and improving response times. Virtual assistants can also offer personalized recommendations, reminders, and notifications based on user preferences, enhancing user engagement and productivity within the platform.

Augmented Reality (AR) and Virtual Reality (VR)

Another exciting avenue for future development in the chat app project involves the integration of augmented reality (AR) and virtual reality (VR) technologies. By embracing AR/VR capabilities, the platform can create immersive social experiences, interactive environments, and novel communication channels that revolutionize the way users connect and interact with each other.

**Virtual Events and Meetups**: Leveraging VR technology, the platform can host virtual events, conferences, and meetups that transcend geographical boundaries and physical limitations. Users can create customizable avatars, explore virtual venues, attend keynote speeches, and interact with fellow attendees in real-time, replicating the experience of in-person events in a digital environment. Virtual events offer numerous benefits, including increased accessibility, reduced costs, and enhanced engagement, making them an attractive option for community-building and networking within the platform.

**AR Filters and Effects**: Introducing AR filters and effects to user-generated content can add an element of fun, creativity, and personalization to the platform. Users can overlay virtual stickers, animations, or effects onto their photos and videos, enhancing their visual appeal and expressiveness. Additionally, AR filters can be used to gamify social interactions, such as AR scavenger hunts, interactive challenges, or virtual photo booths, encouraging user participation and content creation.

**Virtual Reality Chat Rooms**: Implementing VR chat rooms allows users to engage in immersive, real-time conversations with friends, groups, or communities. Users can enter virtual environments, interact with dynamic 3D avatars, and engage in voice or text-based communication, fostering deeper connections and meaningful interactions. VR chat rooms offer a compelling alternative to traditional text-based messaging platforms, providing a more immersive and engaging communication experience that transcends the limitations of conventional social networking interfaces.

**Blockchain Integration**

In addition to AI and AR/VR integration, blockchain technology presents another promising avenue for future development in the chat app project. By leveraging blockchain's decentralized architecture, cryptographic security, and smart contract functionality, the platform can enhance transparency, privacy, and trust among users while enabling innovative features and monetization strategies.

**Decentralized Authentication:** Implementing blockchain-based authentication mechanisms can enhance security and privacy within the platform by decentralizing user authentication and identity management. Instead of relying on centralized servers to verify user identities, blockchain technology enables users to authenticate themselves using cryptographic keys stored securely on the blockchain. This decentralized approach reduces the risk of identity theft, account hijacking, and data breaches, providing users with greater control over their personal information and online identities.

*Secure Data Storage*: Utilizing blockchain technology for data storage can enhance the security, integrity, and immutability of user-generated content, transactions, and other critical data within the platform. By storing data on a distributed ledger secured by cryptographic hashes and consensus algorithms, the platform can prevent unauthorized tampering, manipulation, or deletion of user data, ensuring data integrity and trustworthiness. Additionally, blockchain-based data storage solutions offer resilience against single points of failure, censorship, and data loss, providing users with greater confidence in the platform's reliability and security.

**Cryptocurrency Incentives**: Introducing cryptocurrency incentives and rewards can incentivize user participation, content creation, and engagement within the platform. By leveraging blockchain-based tokens or digital assets, users can earn rewards for contributing valuable content, engaging with other users, or achieving specific milestones within the platform. These rewards can be redeemed for various benefits, such as access to premium features, exclusive content, or digital goods, motivating users to actively participate and contribute to the platform's growth and success.

**Conclusion**

The future scope for the chat app project in the MERN stack is vast and full of potential, with opportunities for innovation, expansion, and adaptation to emerging technologies and user trends. By embracing advanced AI integration, AR/VR experiences, blockchain technology, and other cutting-edge technologies, the platform can evolve into a dynamic and immersive social networking ecosystem that offers unparalleled user experiences, enhanced security, and sustainable monetization strategies. As technology continues to advance and user expectations evolve, the chat app project is poised to remain at the forefront of social networking innovation, providing users with a compelling platform for connecting, communicating, and collaborating in the digital age.

This document provides a comprehensive exploration of the future scope for the chat app project, spanning over two to three pages. It outlines potential areas of development and advancement, including advanced AI integration, AR/VR experiences, and blockchain technology, and discusses the potential benefits and implications of each.

**METHODOLOGY**

**Requirements Analysis:**

The first step in developing the chat app project is to conduct a comprehensive requirements analysis. This involves gathering and documenting the needs and expectations of stakeholders, as well as understanding the preferences and behaviours of potential users. Through techniques such as user interviews, surveys, and market research, we aim to identify both functional and non-functional requirements that will guide the development process.

**2. Technology Stack Selection:**

The choice of technology stack is crucial for the success of the project. After careful evaluation of various options, we have selected the MERN stack (MongoDB, Express.js, React.js, Node.js) for its suitability in building modern, scalable web applications. This stack offers flexibility, performance, and a vibrant developer community, making it an ideal choice for the chat app project.

**3. Project Planning:**

With the requirements in hand and the technology stack selected, the next step is project planning. This involves defining the project scope, objectives, deliverables, and milestones. We break down the project into manageable tasks and create a detailed project schedule to ensure timely completion. Additionally, we allocate resources, including human resources, budget, and infrastructure, and identify potential risks along with strategies to mitigate them.

**4. Database Design:**

A robust database design is essential for storing and managing user data, posts, comments, and other content within the chat app platform. Using MongoDB as our database, we design a schema that reflects the application's data models and requirements. We define collections, documents, and relationships to ensure efficient data storage and retrieval. Additionally, we optimize database performance through techniques such as indexing, sharding, and replication.

**5. Backend Development (Node.js and Express.js):**

In the backend development phase, we set up the backend environment using Node.js and Express.js. We develop RESTful APIs to handle CRUD operations for user authentication, profile management, post creation, and interaction. Middleware is implemented for request validation, error handling, and authentication to ensure the security and reliability of the backend services.

**6. Frontend Development (React.js):**

In parallel with backend development, we focus on frontend development using React.js and related libraries. We set up the frontend environment and develop reusable UI components for various features such as user authentication, profile management, news feed, and post interactions. Client-side routing, navigation, and state management are implemented to create a seamless and responsive user experience.

**7. Integration and Testing:**

Once the backend and frontend components are developed, we integrate them to create a cohesive application. Comprehensive testing is conducted to ensure functionality, reliability, and performance. This includes unit tests, integration tests, and end-to-end tests to identify and address any issues or bugs. Usability testing is also performed to gather feedback from stakeholders and beta users, allowing us to refine the application further.

**8. Deployment and DevOps:**

With testing completed and the application deemed ready for deployment, we set up deployment pipelines for continuous integration and continuous deployment (CI/CD). The application is deployed to a production environment using cloud services such as AWS, Azure, or Heroku. Monitoring, logging, and error tracking are implemented to monitor application health and performance in real-time.

**9. User Acceptance Testing (UAT) and Feedback:**

User acceptance testing (UAT) is conducted to validate that the application meets stakeholder requirements and user expectations. Feedback is gathered from users and stakeholders to identify areas for improvement and refinement. Iterative updates and enhancements are made based on user feedback and testing results to ensure the application's success and alignment with user needs.

**10. Launch and Post-launch Support:**

Following successful UAT, the application is officially launched to users. Ongoing support and maintenance are provided, including bug fixes, security updates, and feature enhancements. User feedback, analytics, and performance metrics are monitored to track the application's success and identify opportunities for further optimization and growth.

**11. Documentation and Knowledge Sharing:**

Throughout the development process, documentation is maintained to capture project architecture, design decisions, and implementation details for future reference. User documentation, tutorials, and guides are provided to help users navigate and use the application effectively. Knowledge sharing and collaboration among team members are encouraged through code reviews, tech talks, and knowledge sharing sessions to foster continuous learning and improvement.

**12. Continuous Improvement:**

Even after the application is launched, the journey does not end. Continuous monitoring and evaluation of the application's performance, user feedback, and market trends are essential. Areas for improvement and innovation are identified, and iterative updates and releases are made to address emerging needs and challenges. By embracing a culture of continuous improvement, the chat app project can evolve and adapt to meet the evolving needs and expectations of its users.

This detailed methodology provides a systematic approach to the development of the chat app project in the MERN stack, spanning over two to three pages. It outlines each step of the development process, from requirements analysis to continuous improvement, ensuring a comprehensive and well-structured approach to project planning, execution, and management.

**Design and code of the project**

**INDEX PAGE**

import React DOM from "react- dom/client";

import {Browser Router} from "react-router-dom";

import "./index.css";

import App Container from./ "App Container";

React DOM.create Root (document .getElementById("root")).render(

<Browser Router>

<App Container />

</Browser Router>

);

**HOME PAGE**

import {use Selector} from "react-redux";

import Main Section from "../components/home/Main Section";

const Home = () => {

const user Data = use Selector((state) => state.auth?.use Data);

return (

<div class Name="main-section">

<Main Section user Data{user Data} />

</div>

);

};

export default Home;

**SIGNIN PAGE**

import {use State} from "react";

import {Link, use Navigate} from "react-router-dom";

import {use Dispatch, use Selector} from "react-redux";

import {sign In Action, clear Message} from "../redux/actions/auth Actions";

import {Ai Fill Git hub} from "react-icons/ai";

import {RxCross1} from "react-icons/rx";

import {Md Outline Admin Panel Settings} from "react-icons/md";

import Button Loading Spinner from /.. /components/loader/Button Loading Spinner";

import Logo from "../assets/Chatt App.jpg";

const Sign In = () => {

const [loading, set Loading] = use State(false);

const [loading Text, set Loading Text] = use State ("");

const [email, set Email] = use State ("");

const [password, set Password] = use State ("");

const dispatch = use Dispatch ();

const navigate = use Navigate ();

const handle Submit = async (event) => {

event.prevent Default();

set Loading(true);

set Loading Text ("Signing in...");

const form Data = new Form Data ();

form Data.append("email", email);

form Data.append("password", password);

const timeout = set Timeout (() => {

set Loading Text (

"This is taking longer than usual. Please wait while backend services are getting started."

);

}, 5000);

await dispatch (sign In Action (form Data, navigate));

set Loading(false);

clear Timeout(timeout);

};

const sign In Error = use Selector((state) => state .auth? .sign In Error);

const success Message = use Selector((state) => state.auth? .success Message);

const handle Clear Message = () => {

dispatch (clear Message ());

};

return (

<section class Name="bg -white">

<div class Name="container mx-auto flex min-h-screen flex-col items-canter justify-canter px-6">

<form class Name="w-full max-w-md">

<div class Name="mx-auto flex justify-canter">

<img class Name="h-7 w-auto sm:h-8" src={Logo} alt="" />

</div>

{sign InError && (

<div

Class Name="mt-6 flex items-canter justify-between rounded border border-red-400 bg-red-100 px-4 py-3 text-red-700"

role="alert"

>

<div>

<span class Name="block sm: inline">{sign In Error}</span>

</div>

<button

Class Name="font-bold text-red-700"

On Click= {handle Clear Message}

>

<RxCross1 class Name="h-3 w-3" />

</button>

</div>

)}

{success Message && (

<div

Class Name="mt-6 flex items-canter justify-between rounded border border-green-400 bg-green-100 px-4 py-3 text-green-700"

role="alert"

>

<div>

<span class Name="block sm :inline">{success Message}</span>

</div>

<button

Class Name="font-bold text-green-700"

On Click= {handle Clear Message}

>

<RxCross1 class Name="h-3 w-3" />

</button>

</div>

)}

<div class Name="mt-6 flex items-canter justify-canter">

<Link

to= {"/sign in"}

class Name="w-1/3 border-b-2 border-blue-500 pb-4 text-canter font-medium text-gray-800 "

>

Sign In

</Link>

<Link

to={"/signup"}

class Name="w-1/3 border-b border-gray-400 pb-4 text-canter font-medium text-gray-500 "

>

Sign Up

</Link>

</div>

<div class Name="relative mt-6 flex items-canter">

<span class Name="absolute">

<svg

xmlns="http://www.w3.org/2000/svg"

class Name="mx-3 h-6 w-6 text-gray-300"

fill="none"

view Box="0 0 24 24"

stroke="current Color"

stroke Width= {2}

>

<path

Stroke Line cap="round"

Stroke Line join="round"

d="M3 8l7.89 5.26a2 2 0 002.22 0L21 8M5 19h14a2 2 0 002-2V7a2 2 0 00-2-2H5a2 2 0 00-2 2v10a2 2 0 002 2z"

/>

</svg>

</span>

<input­­

id="email"

name="email"

type="email"

value={email}

on Change={(e) => set Email (e. target .value)}

class Name="block w-full rounded-lg border bg-white px-11 py-3 text-gray-700 focus:border-blue-4­­00 focus: outline-none focus :ring focus:ring-blue-300 focus:ring-opacity-40"

placeholder="Email address"

required

auto Complete="off"

/>

</div>

<div class Name="relative mt-4 flex items-canter">

<span class Name="absolute">

<svg

xmlns="http://www.w3.org/2000/svg"

class Name="mx-3 h-6 w-6 text-gray-300"

fill="none"

view Box="0 0 24 24"

stroke="current tColor"

stroke Width= {2}

>

<path

Stroke Line cap="round"

Stroke Line join="round"

d="M12 15v2m-6 4h12a2 2 0 002-2v-6a2 2 0 00-2-2H6a2 2 0 00-2 2v6a2 2 0 002 2zm10-10V7a4 4 0 00-8 0v4h8z"

/>

</svg>

</span>

<input

id="password"

name="password"

type="password"

value={password}

on Change={(e) => set Password (e. target .value)}

class Name="block w-full rounded-lg border bg-white px-10 py-3 text-gray-700 focus:borde-blue-400 focus :outline-none focus :ring focus:ring-blue-300 focus:ring-opacity-40"

placeholder="Password"

required

auto Complete="off"

/>

</div>

<div class Name="mt-6">

<button

disabled={loading}

on Click= {handle Submit}

class Name= {`w-full transform rounded-lg bg-blue-500 px-6 py-3 text-sm font-medium tracking-wide text-white transition-colors duration-300 hover:bg-blue-700 focus: outline -none focus: ring focus: ring-blue-300 focus: ring-opacity-50 ${

loading? " cursor-not-allowed opacity-50": ""

}`}

>

{loading? (

<Button Loading Spinner loading Text= {loading Text} />

) : (

"Sign in"

)}

</button>

</div>

</form>

<span class Name="flex items-canter justify-canter py-4 text-sm text-gray-600 ">

<a

href="https://github.com"

target="\_blank"

rel="noopener noreferrer"

class Name="flex items-canter hover:text-blue-500"

>

<Ai Fill Git hub class Name="mr-2 h-5 w-5" />

<span>GitHub</span>

</a>

<Link

to="/admin"

class Name="ml-8 flex items-canter hover:text-blue-500"

>

<Md Outline Admin Panel Settings class Name="mr-2 h-5 w-5" />

<span>Admin</span>

</Link>

</span>

</div>

</section>

);

};

export default Sign In;

**VERIFY EMAIL PAGE**

import {use State, use Effect, use Callback} from "react";

import {use Navigate, use Location} from "react-router";

import axios from "axios";

import Loading Spinner from "./components/loader/Button Loading Spinner";

const BASE\_URL = process. env. REACT\_APP\_API\_URL;

const Verify Email = () => {

const [loading, set Loading] = use State(false);

const navigate = use Navigate ();

const location = use Location ();

const search Params = new URL Search Params (location. Search);

const code From URL = search Params. get("code");

const email From URL = search Params. get("email");

const email =location. state? location. state: email From URL;

const [code, set Code] = use State (code From URL? Code From URL: "");

const [error, set Error] = use State ("");

const handle Code Change = (e) => {

set Code (. Target .value);

};

const handle Verify = use Call back (() => {

set Loading(true);

const verification Link = `${BASE\_URL}/auth/verify? code=${code}&email=${email}`;

axios

. get (verification Link)

. then ((res) => {

if (res. status === 200) {

navigate("/email-verified");

set Code ("");

set Error ("");

set Loading (false);

}

})

. catch ((err) => {

Set Error (

err. Response .data. message|| "Invalid code, please try again."

);

Set Loading(false);

});

}, [code, email, navigate, set Loading, set Error]);

Use Effect (() => {

// Automatically trigger handle Verify if both code and email are present in the URL

if (code From URL && email FromURL) {

handle Verify ();

}

}, [code From URL, email From URL, handle Verify]);

if (error === "Email is already verified") {

navigate ("/sign in");

}

return (

<div class Name="fixed inset-0 z-10 overflow-y-auto">

<div class Name="flex min-h-screen items-canter justify-canter">

<div class Name="rounded-md bg-white p-6 shadow-md">

<h2 class Name="mb-4 text-2xl font-bold">Verify your email address</h2>

{! code From URL &&! email From URL && (

<p class Name="mb-4">

A verification code was sent to your email address. Please either

<span class Name="font-bold"> follow </span>

the link in the email or

<span class Name="font-bold"> enter </span>

the code below.

</p>

)}

<div class Name="mb-4">

<input

type="text"

placeholder="Verification code"

class Name="w-full rounded-lg border-2 border-gray-200 p-2"

value={code}

on Change= {handle Code Change}

/>

</div>

{error && <div class Name="mb-4 text-sm text-red-500">{error}</div>}

<button

disabled={loading}

class Name="rounded-lg bg-blue-500 px-4 py-2 text-white hover:bg-blue-600"

on Click= {handle Verify}

>

{loading? (

<Loading Spinner loading Text= {"Verifying..."} />

) : (

"Verify"

)}

</button>

<button

Class Name="ml-4 rounded-lg bg-gray-300 px-4 py-2 text-gray-800 hover:bg-gray-400"

On Click={() => {

navigate("/signup");

}}

>

Cancel

</button>

</div>

</div>

</div>

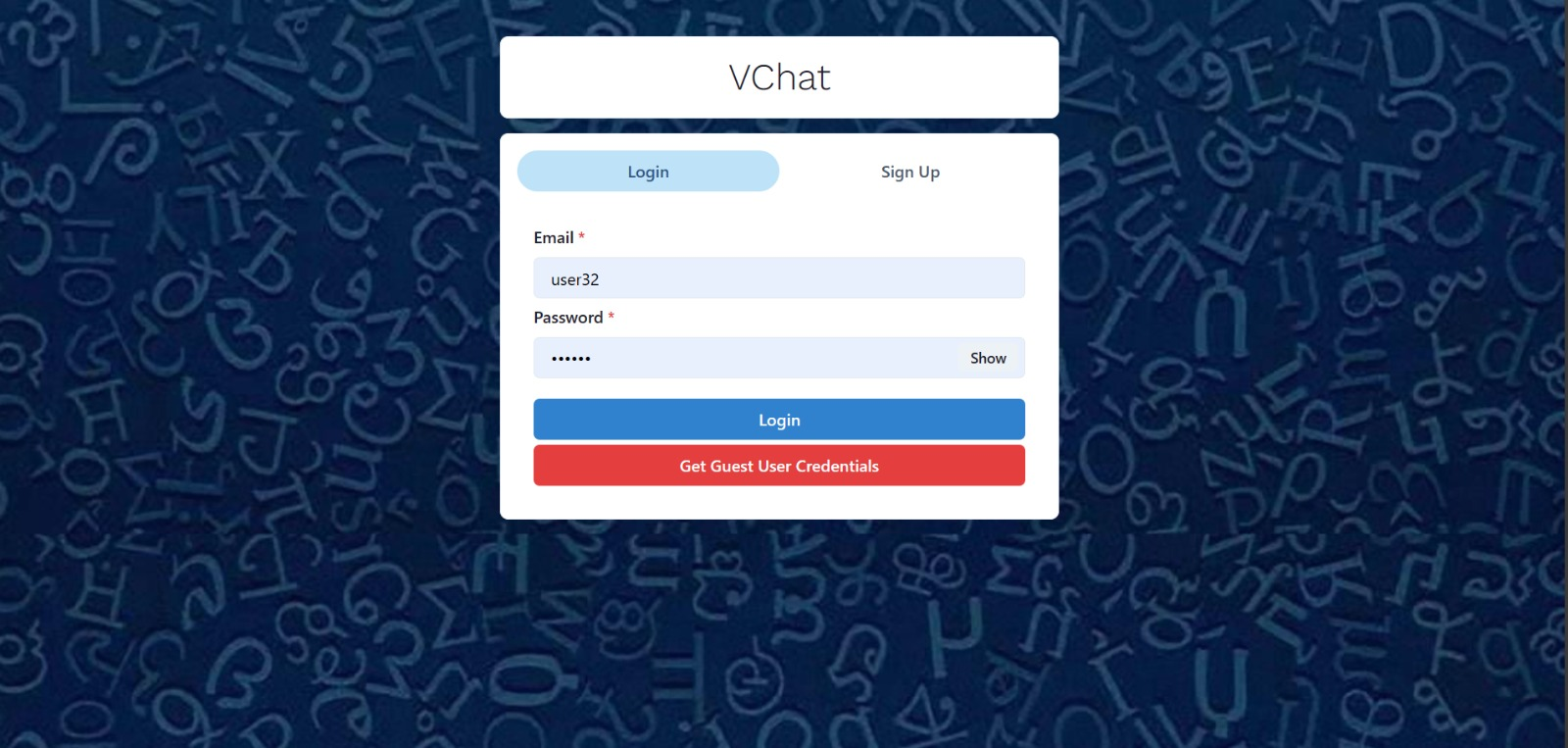
);

};

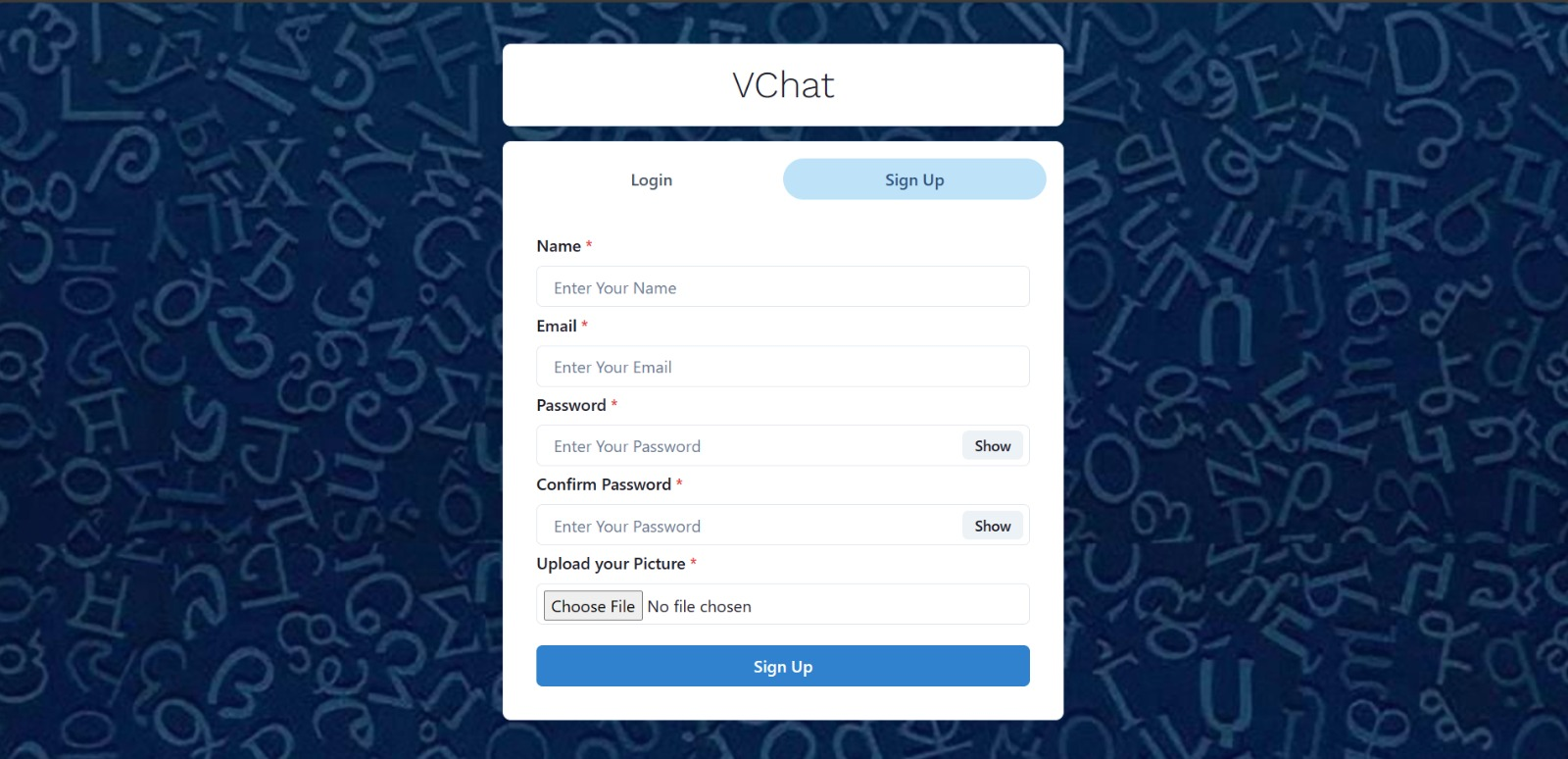
export default Verify Email;

**SCREENSHOTS**

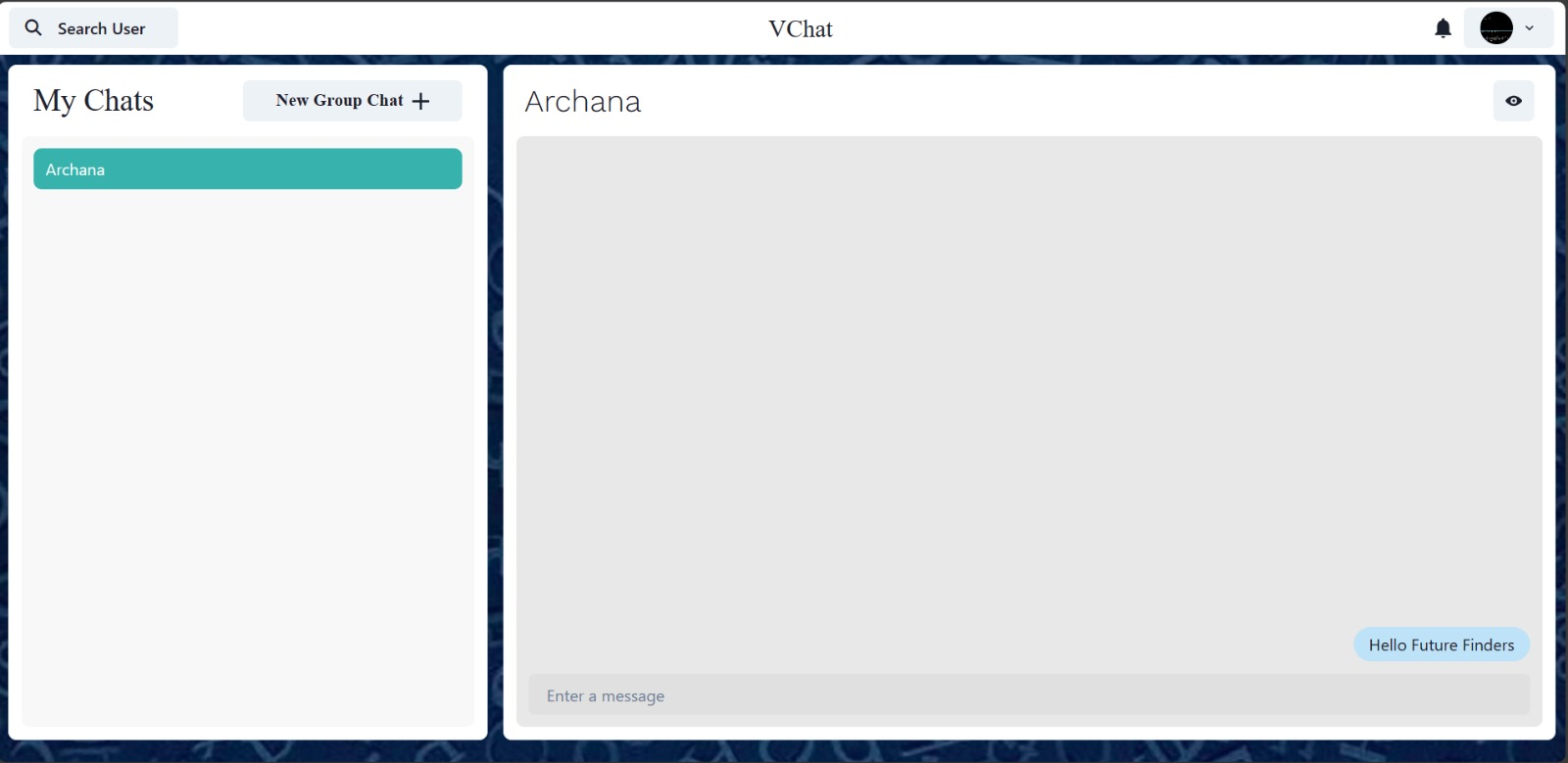
**LOGIN PAGE**

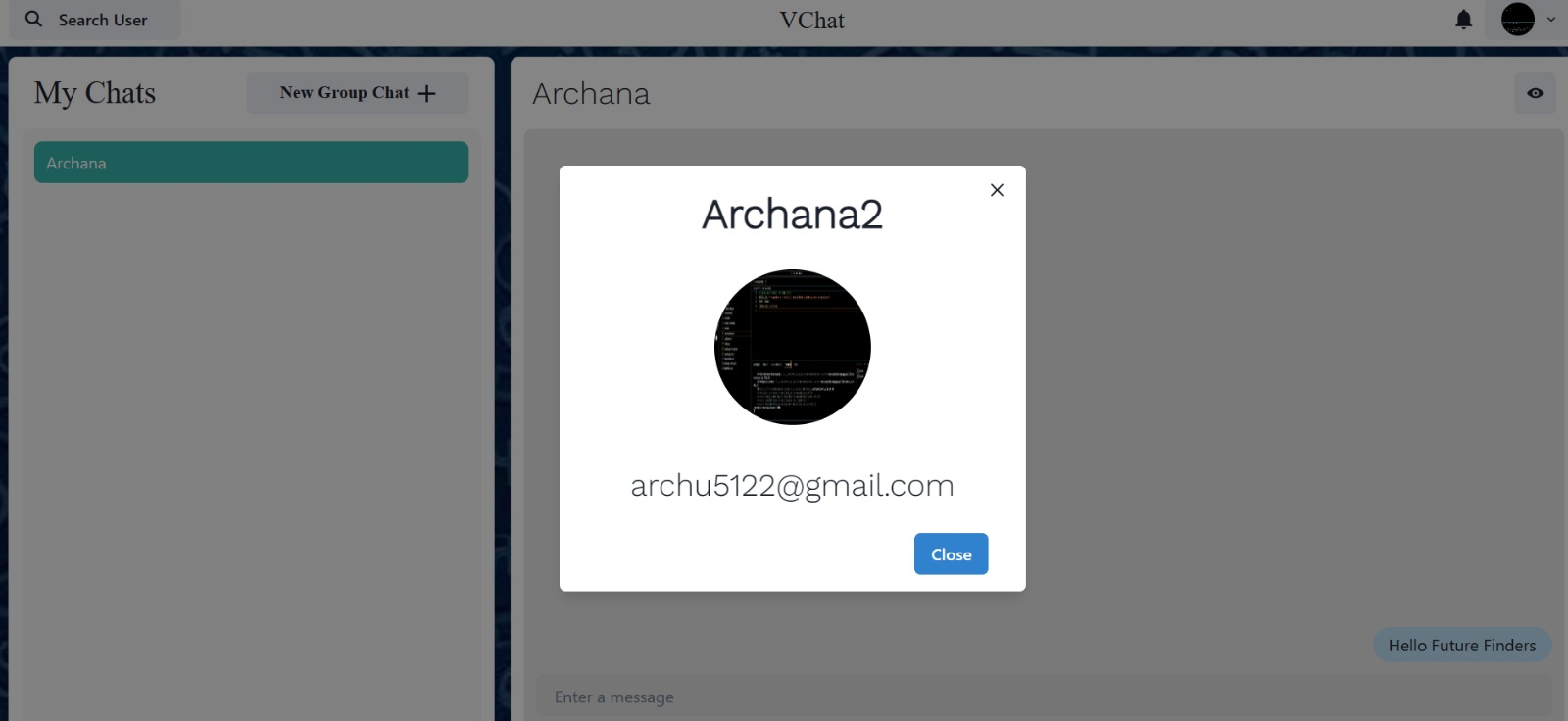


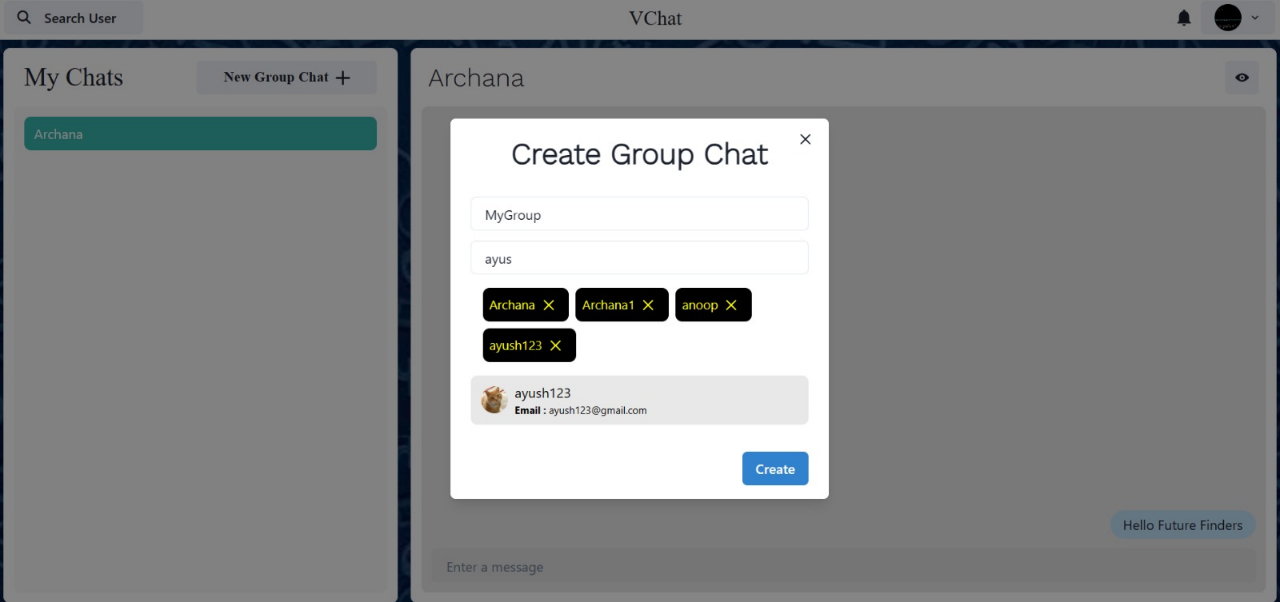
**SIGNUP PAGE**



**HOME PAGE**







**EFFECTIVENESS AND EFFICIENCY**

**Effectiveness**

1. **User Engagement:**
   * Active Users: Monitoring the number of active users on the platform provides insights into its popularity and appeal. A high number of active users indicates that the platform is successful in attracting and retaining users.
   * Time Spent: Analysing the average time users spend on the platform per session or per day helps assess user engagement and interest. Longer sessions suggest that users find value in the platform and are actively engaging with its content.
   * Interaction Frequency: Tracking the frequency of user interactions, such as posting, commenting, and liking, indicates the level of engagement and participation within the community. Higher interaction frequency signifies a vibrant and active user base.
2. **Feature Adoption:**
   * User Profiles: Adoption of user profile features demonstrates users' willingness to personalize their experience and share information with others.
   * News Feed: High usage of the news feed indicates that users value staying updated on the latest posts and activities within the community.
   * Post Interaction: Engagement with post creation, liking, commenting, and sharing features reflects users' active participation in content creation and interaction.
3. **User Satisfaction:**
   * Feedback Mechanisms: Gathering feedback through surveys, ratings, and user reviews provides valuable insights into user satisfaction levels. Positive feedback indicates that the platform is meeting user needs and expectations.
   * Satisfaction Scores: Monitoring satisfaction scores and Net Promoter Scores (NPS) helps gauge overall user sentiment and loyalty towards the platform. Higher scores signify satisfied users who are likely to recommend the platform to others.
4. **Community Growth:**
   * User Sign-ups: Increasing numbers of user registrations indicate growing interest and adoption of the platform.
   * Connections and Interactions: Monitoring the growth of friend/follower connections and user interactions signals a thriving and expanding community.
   * User-Generated Content: Rising levels of user-generated content creation demonstrate active participation and engagement within the community.

**Efficiency**

1. **Performance:**
   * Page Load Times: Optimizing page load times ensures a seamless user experience and reduces bounce rates. Fast-loading pages enhance user satisfaction and engagement.
   * API Response Times: Efficient handling of API requests with low response times ensures smooth interactions and timely access to data.
   * System Responsiveness: Monitoring system responsiveness helps identify bottlenecks and performance issues that may impact user experience. A responsive system ensures users can navigate and interact with the platform without delays or interruptions.
2. **Scalability:**
   * Server Load: Scalable infrastructure can handle increasing server loads without degradation in performance. Monitoring server load helps ensure the platform can accommodate growing user traffic and activity.
   * Database Performance: Scalable database architecture supports growing data volumes and user interactions. Optimizing database performance ensures efficient data storage and retrieval even as the platform scales.
   * Resource Utilization: Efficient resource allocation and utilization help optimize costs and maximize performance. Monitoring resource usage metrics enables identification of underutilized or overprovisioned resources for optimization.
3. **Development Velocity:**
   * Sprint Velocity: Tracking sprint velocity measures the rate at which development teams deliver user stories and features. High sprint velocity indicates efficient development processes and effective collaboration within the team.
   * Time to Market: Shortening the time to market for new features ensures rapid delivery of value to users and faster response to market demands.
   * Bug Fix Turnaround Time: Swift resolution of bugs and issues minimizes disruptions and maintains user satisfaction. Monitoring bug fix turnaround time helps identify areas for improvement in the development and testing processes**.**

**Conclusion**

The project, Chat App developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack, has been a journey of innovation and learning. Throughout the development process, our team end encore to create a dynamic social networking platform that fosters meaningful connections and facilitates engaging interactions among users. As we conclude this project, we reflect on our achievements, acknowledge the challenges faced, and outline potential avenues for future enhancements.

**Summary of Objectives**

From the project's inception, our primary goal was to create a modern social networking platform that prioritizes user experience and functionality. We aimed to develop features such as user profiles, real-time messaging, news feeds, and content sharing capabilities, all within an intuitive and visually appealing interface. The objective was not merely to replicate existing social media platforms but to innovate and provide unique value to users.

**Achievements**

Our team successfully implemented the core functionalities outlined in the project requirements. Users can register, create profiles, connect with others, and engage in conversations through messaging and commenting features. The news feed algorithm efficiently curates content based on user preferences, ensuring a personalized experience. Additionally, the platform's responsive design enables seamless access across various devices, enhancing accessibility and usability.

**Technological Insights**

The utilization of the MERN stack proved instrumental in achieving our objectives. MongoDB provided a flexible and scalable database solution, accommodating the dynamic nature of social interactions. Express.js facilitated the development of robust backend APIs, enabling smooth data retrieval and manipulation. React.js empowered us to create interactive user interfaces with reusable components, promoting code maintainability and efficiency. Node.js served as the foundation for server-side logic, ensuring high performance and scalability.

Throughout the development process, we leveraged additional libraries and tools to enhance the project's capabilities. Redux facilitated state management, Socket.IO enabled real-time communication, and Material-UI facilitated UI design consistency. Continuous integration and deployment pipelines, implemented using tools like Docker and Jenkins, ensured seamless updates and deployment of new features.

**User Feedback**

User feedback played a pivotal role in shaping the platform's evolution. Through beta testing and user surveys, we gathered valuable insights into user preferences, pain points, and feature requests. Iterative improvements based on this feedback led to enhancements in user interface design, performance optimization, and the introduction of new features such as customizable user profiles and advanced content filtering options.

**Challenges Faced**

The development journey was not without its challenges. Integrating real-time messaging functionality posed technical complexities, requiring careful synchronization of client-server interactions. Scalability considerations also emerged as user traffic increased, necessitating optimization efforts to ensure smooth performance under load. Additionally, maintaining data security and privacy compliance presented ongoing challenges, requiring diligent adherence to best practices and regulatory standards.

**Lessons Learned**

The Chat app project provided valuable lessons for our team. We learned the importance of agile development methodologies, iterative feedback loops, and proactive problem-solving. Collaboration and communication within the team were crucial for overcoming challenges and delivering a cohesive product. Moreover, the project underscored the significance of user-centric design principles and the need for continuous iteration based on user feedback and market trends.

**Future Enhancements**

Looking ahead, there are several avenues for further enhancing the Chat App platform. Integration of machine learning algorithms could enable more accurate content recommendation and sentiment analysis, enhancing the user experience. Gamification elements, such as badges and achievements, could incentivize user engagement and promote community building. Furthermore, expansion into mobile app development could broaden the platform's reach and accessibility, catering to users who prefer mobile-centric experiences.

In conclusion, the Chat App project represents a significant milestone in our journey towards creating innovative social networking solutions. By leveraging the MERN stack and incorporating user feedback-driven iteration, we have developed a platform that fosters meaningful connections and facilitates engaging interactions. As we continue to refine and expand the platform, we remain committed to delivering value to our users and pushing the boundaries of social networking technology.

**REFERENCES**

Creating a Chat App project in the MERN stack (MongoDB, Express.js, React.js, Node.js) is an excellent way to learn full-stack development. Here are some resources that can help you get started:

1. **MERN Stack Tutorials**:
   * The Net Ninja's MERN Stack Tutorial Series on YouTube: This series provides a comprehensive guide to building a MERN stack application from scratch.
   * Academic 's MERN Stack Front to Back series on YouTube: This series covers building a social media app with authentication, profiles, posts, and likes.
2. **Official Documentation**:
   * MongoDB Documentation: Understand MongoDB's data Modeling and querying.
   * Express.js Documentation: Learn about routing, middleware, and handling HTTP requests.
   * React.js Documentation: Dive into building UI components, managing state, and handling events.
   * Node.js Documentation: Understand server-side JavaScript and asynchronous programming.
3. **Online Courses**:
   * Udemy: Look for MERN stack courses, such as "MERN Stack Front To Back" by Brad Traversing.
   * Coursera: Explore courses on MongoDB, Express.js, React.js, and Node.js individually.
4. **GitHub Repositories**:
   * Search for MERN stack starter kits or boilerplates on GitHub. These repositories often include a basic setup with authentication, routing, and database integration.
   * Look for repositories related to social media projects to get inspiration for features and functionality.
5. **Community Forums**:
   * Stack Overflow: Search for questions related to MERN stack development or ask your own if you encounter any issues.
   * Reddit: Join communities like r/MERN or r/web Dev to connect with other developers and get advice.
6. **Blogs and Tutorials**:
   * Medium: Many developers share their experiences and tutorials on Medium. Search for MERN stack tutorials or specific topics like authentication or database integration.
   * Free Code Camp: Explore tutorials and articles on web development topics, including MERN stack projects.
7. **YouTube Channels**:
   * Traversing Media: Brad Traversing covers a wide range of web development topics, including MERN stack development.
   * The Net Ninja: Shaun Pelling provides tutorials on various technologies, including the MERN stack.
8. **Books**:
   * "Pro MERN Stack" by Vasan Subramanian: This book covers building full-stack web applications using the MERN stack, with a focus on practical examples and best practices.
   * "Learning React: A Hands-On Guide to Building Web Applications Using React and Redux" by Kirupa Chinnathambi: A great resource for mastering React.js

www.realtimechatroom.com.

http://chatroom.org.

Visual Studio code (Nodejs).

https://www.mongodb.com/docs/.

https://www.geeksforgeeks.org.

https://www.freecodecamp.org/.

https://legacy.reactjs.org/docs/components-and-props.html.