CHAT SPHERE

A PROJECT REPORT

Submitted by

JANVI (20BCS4537)
NIMISH SURI (20BCS4550)
GAURAV MALIK (20BCS4573)
SACHIN CHAUDHARY (20BCS4517)

in partial fulfillment for the award of the degree of

Bachelors of Computer Science Engineering

IN

INTERNET OF THINGS

Chandigarh University

NOVEMBER & 2023

BONAFIDE CERTIFICATE

Certified that this project report **"CHAT SPHERE"** is the bonafide work of "JANVI, NIMISH SURI, GAURAV MALIK, SACHIN CHAUDHARY" who carried out the project work under my/our supervision.

SIGNATURE
Dr. Aman Kaushik
Head of the Department

HEAD OF THE DEPARTMENT

AIT-CSE

SIGNATURE
Dr. Dhawan Singh

Professor

AIT-CSE

Submitted for the project viva-voce examination held on 30th November,2023.

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

List of Figures

ABSTRACT

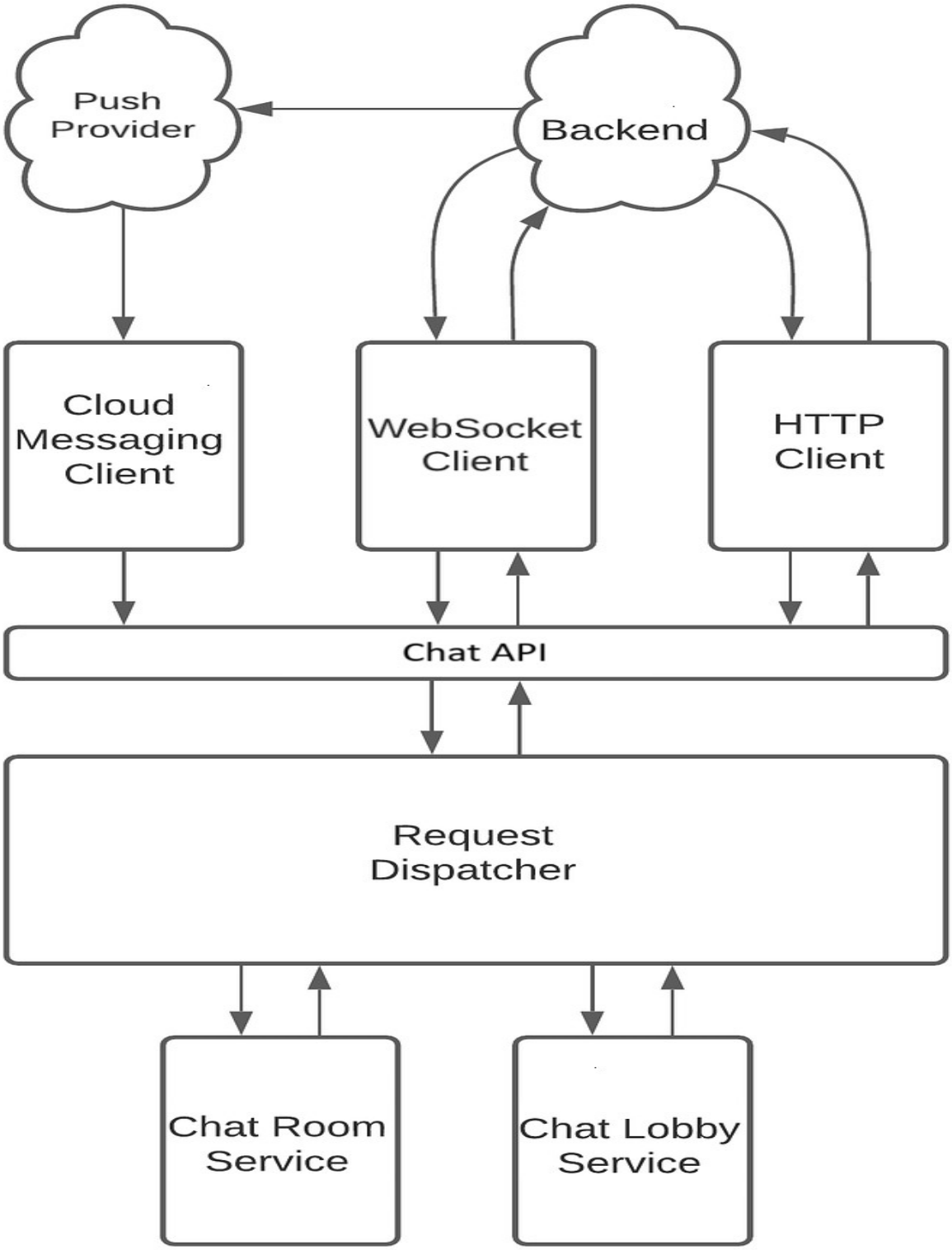
The emergence of new technologies has significantly reshaped the landscape of interpersonal communication. In today's digital age, messaging applications stand out as one of the most prevalent means of interaction. To meet this demand, a myriad of chat applications has been developed. This thesis introduces a feature-rich chat application crafted using the MERN (MongoDB, Express.js, React, Node.js) stack, augmented by the powerful WebRTC framework. Notably, the application incorporates the essential feature of two-way authentication, adding an extra layer of security to user accounts.

Within this chat application, users can seamlessly create accounts, engage in chat rooms, and exchange messages in real-time. Leveraging the capabilities of web sockets and Angular's two-way binding, the application ensures that messages are instantly visible as soon as they are sent. The inclusion of WebRTC further enhances the real-time communication aspect, providing robust support for audio, video, and text-based interactions.

A noteworthy addition to the application is the implementation of two-way authentication. This security feature requires users to authenticate their identity through two distinct methods, typically something they know and something they have . This dual-layered authentication mechanism adds an extra level of protection, ensuring secure access to chat rooms and safeguarding user accounts from unauthorized access.

Through this project, the aim is twofold: to showcase the feasibility and effectiveness of building chat spheres using the MERN stack and WebRTC, and to highlight the importance of incorporating advanced security measures such as two-way authentication. This amalgamation not only capitalizes on the strengths of the MERN stack and WebRTC for robust web application development but also prioritizes user security in an era where data protection is paramount.

GRAPHICAL ABSTRACT



ABBREVIATIONS

Abbrevations	Definitions
AES	Advanced Encryption Standard
AML	America Online instant messeger
API	Application programming interface
CSS	Cascading Style Sheet
E2EE	End to end encryption
HTML	Hyper text markup language
ICE	Interactive Connectivity Establishment
ICQ	I seek you
IRC	Internet Relay Chat
JSON/BSON	JavaScript Object Notation/ Binary JSON
MSN	Microsoft network
NAT	Network Address Translation
NAT	Network address translator
P2P	Peer-to-peer
SQL	Simple query language
STUN	Session Traversal Utilities for NAT
TURN	Traversal Using Relays around NAT
Web-RTC	Web Real-Time Communication
XSS	Cross site scripting

Chapter 1: Introduction

With the passage of time, the rapid advancement of information and communication technology is assuming an increasingly pivotal role in society. These technological innovations have granted people the ability to effortlessly retrieve information and engage in communication from virtually any location, at speeds hitherto inconceivable. In the contemporary milieu, a considerable number of individuals have shifted their reliance on conventional outlets such as newspapers and television news channels. Instead, the majority now turn to intelligent technology, encompassing smartphones and other devices, to access a diverse array of information.

Chat applications have assumed a central role in this transformation. Individuals are progressively employing these platforms to exchange information and engage in discussions about current events, both locally and on a global scale. Nevertheless, these chat applications come accompanied by their unique set of pros and cons, mirroring the nature of technology in general. User feedback and their firsthand encounters with these applications play a pivotal role in aiding individuals in evaluating the essential features and areas that warrant enhancement.

It's worth emphasizing that chat applications are meticulously crafted to offer complete responsiveness, ensuring a user-friendly and accessible experience across diverse devices, including web-based and mobile platforms. The technological framework supporting these applications encompasses MongoDB, along with HTML, CSS, JavaScript libraries, and WEB-rtc, all meticulously integrated to guarantee seamless real-time performance of the application.

Chapter 2: Literature survey:

A chat sphere enables instantaneous communication between users, fostering dynamic and interactive conversations. Unlike traditional messaging, real-time chat apps provide immediate message delivery, creating a seamless and responsive communication experience. Whether for personal or business use, these apps leverage cutting-edge technologies to ensure quick, efficient, and engaging conversations in real-time.

2.1 Historical evolution:

The historical backdrop of chat spheres reveals an intriguing journey that has fundamentally altered the way individuals communicate and exchange information in the digital age. This journey was initiated with the inception of Internet Relay Chat (IRC) in 1988, marking one of the earliest milestones in real-time chat technology. IRC provided users the means to connect to servers and partake in live text-based dialogues within chat rooms, thereby serving as the cornerstone for subsequent developments in real-time communication.

The late 1990s and early 2000s witnessed the flourishing of instant messaging with the introduction of pioneering platforms like AOL Instant Messenger (AIM), ICQ (I Seek You), MSN Messenger, and Yahoo Messenger. AIM, for instance, was launched by America Online (AOL) in 1997 and swiftly gained widespread popularity. It introduced the concept of a "buddy list," enabling users to monitor the online status of their contacts and engage in instant messaging, thereby enhancing the accessibility and user-friendliness of real-time chat.

ICQ, introduced in 1996, was notable for pioneering the concept of unique user numbers, and streamlining user connections. Microsoft's MSN Messenger and Yahoo's Messenger further enriched the real-time communication landscape by integrating chat features with additional capabilities such as file sharing.

The evolution of real-time chat persisted with the advent of smartphones and the surge of mobile messaging applications. Platforms like WhatsApp (2009), Facebook Messenger (2011), and WeChat (2011) extended real-time chat to mobile devices, allowing not only text-based messaging but also introducing multimedia features, such as photo and video sharing, rendering communication more dynamic and engaging.[3]

As chat spheres progressed, they integrated advanced features, including voice and video calls, group chats, and task-automating bots. Addressing concerns about security and privacy, end-to-end encryption was developed to safeguard user data. chat spheres have evolved into versatile tools, now integrating with an array of services, from payment systems to e-commerce and gaming platforms, making them comprehensive communication and interaction platforms. This historical journey demonstrates a remarkable transformation from basic text-based chat rooms to sophisticated multimedia communication platforms, ensuring their continuing significance in the dynamic landscape of digital communication.

2.2 Review of existing technologies:

2.2.1 Whatsapp:

Fig.2 (Whatsapp)

WhatsApp stands out for its advanced security features, including end-to-end encryption, ensuring that only the intended recipients can access messages. Its support for various message types, including text, voice, and video, makes it a comprehensive real-time communication platform. The user experience is further enhanced by features like multimedia sharing and the recently introduced multi-device support, providing users with a seamless and secure communication environment.[14]

2.2.2 Slack:

Fig.3 (Slack)

Slack is a prominent platform designed for team collaboration, offering channels, direct messaging, and integrations with third-party tools. Its real-time notifications keep teams informed, while the ability to share files and integrate productivity tools enhances collaboration.

Slack's user experience is defined by its organized workspace and customization options, catering to diverse team communication needs.[14]

2.2.3 Telegram:

Fig.4 (Telegram)

Telegram prioritizes speed and security, providing secret chats with end-to-end encryption. Its versatility is evident through features like channels, groups, and customizable bots, contributing to a dynamic communication ecosystem. With a clean and user-friendly interface, Telegram's commitment to privacy and continuous innovation positions it as a leading chat sphere.[14]

2.2.4 Facebook:

Fig.5 (Facebook)

Facebook Messenger integrates seamlessly with the broader Facebook ecosystem, offering a wide range of multimedia messaging features. Its user experience is characterized by a familiar interface and social media integration, providing users with an engaging environment. The platform continually introduces new features such as chat extensions and group video calls, contributing to its dynamic nature.[11]

2.2.5 Discord:

Fig.6 (Discord)

Initially catering to gamers, Discord has evolved into a versatile communication platform with features like servers, voice channels, and text chat. Its user experience is defined by community-building, offering a dynamic environment for diverse user interactions. Discord's innovative approach to combining text and voice communication has influenced how chat spheres can serve the needs of diverse user communities.[11]

2.2.6 Snapchat:

Fig.7 (Snapchat)

Snapchat revolutionized real-time communication with its emphasis on ephemeral content and multimedia sharing. The platform's unique features, including disappearing messages and augmented reality filters, have reshaped user expectations, particularly among younger demographics. Snapchat's innovative camera features and the introduction of Stories have redefined the possibilities of chat spheres.

2.3 Security and privacy consideration:

security of data and user data privacy should be on the highest priority as it can be dangerous if it gets in wrong hands. There are various malware and viruses that can harm user and data , also unauthorised or malicious user and hacker can also cause harm to data. there are various kind of security threats some of which are explained below

Fig.8 (Privacy & Security)

2.3.1 Phishing Attacks:

Phishing constitutes a malicious tactic wherein attackers leverage deceptive messages or fraudulent websites to coerce users into revealing sensitive information. This may encompass usernames, passwords, credit card details, or other confidential data. Phishing attacks often hinge on social engineering, creating a false sense of urgency or legitimacy to manipulate users into compromising their security.

2.3.2 Man-in-the-Middle Attacks:

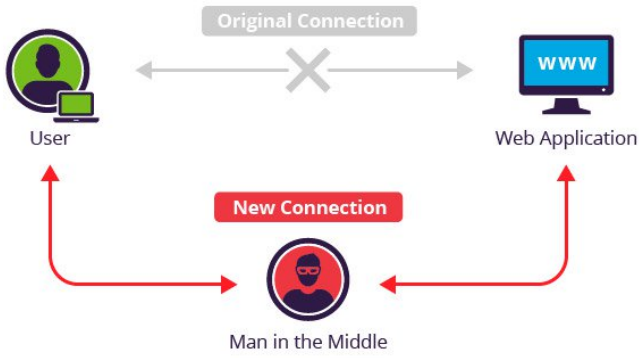


Fig.9 (MiM attack)

A Man-in-the-Middle attack involves an unauthorized entity intercepting and, in certain cases, altering communication between two parties without their knowledge. This can manifest in various forms, such as eavesdropping on public Wi-Fi networks, DNS spoofing, or session hijacking. Encryption, particularly end-to-end encryption, plays a pivotal role in mitigating the risks associated with MitM attacks by ensuring that even if communication is intercepted, the content remains confidential and secure.[20]

2.3.3 Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) Attacks:

Denial-of-Service attacks aim to overwhelm a system, network, or application, rendering it inaccessible to legitimate users.

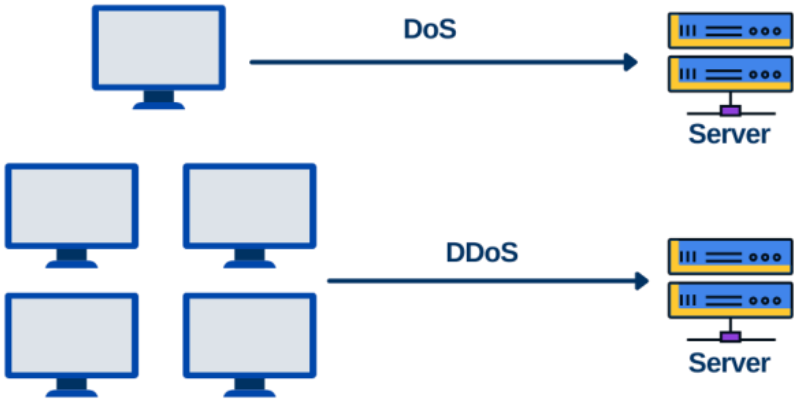


Fig.10 (DoS &DDoS)

Distributed Denial-of-Service attacks involve multiple sources coordinating the attack, amplifying the challenge of mitigation. These attacks flood the target with a high volume of traffic, causing a disruption in normal service. Mitigation strategies encompass implementing traffic filtering, rate limiting, and leveraging Content Delivery Networks (CDNs) to absorb and distribute traffic.

2.3.4 Brute Force Attacks:

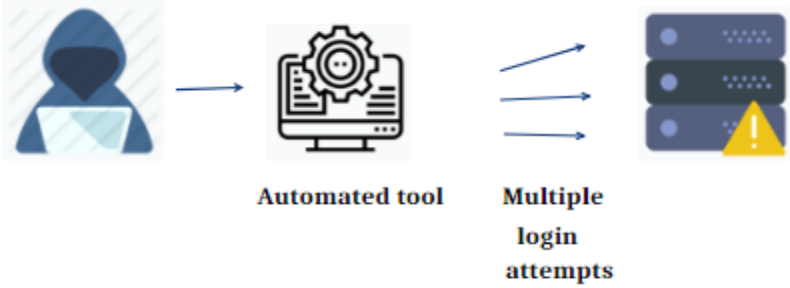


Fig.11 (Brute force)

Brute force attacks involve systematically attempting all possible combinations of passwords or encryption keys until the correct one is found. In the context of a chat application, this could entail repeatedly trying different password combinations until the attacker gains access. Implementing account lockout policies after a certain number of failed login attempts serves as a common defense mechanism against successful brute force attacks.

2.3.5 SQL Injection Attacks:

Fig.12 (SQL injection)

SQL injection is a type of attack where malicious SQL queries are injected into input fields of a web application. In instances where the application fails to validate or sanitize user input adequately, attackers can manipulate the SQL queries and potentially gain unauthorized access to the database. Implementing rigorous input validation and utilizing parameterized queries constitutes effective measures to prevent SQL injection attacks.

2.3.6 Cross-Site Scripting (XSS) Attacks:

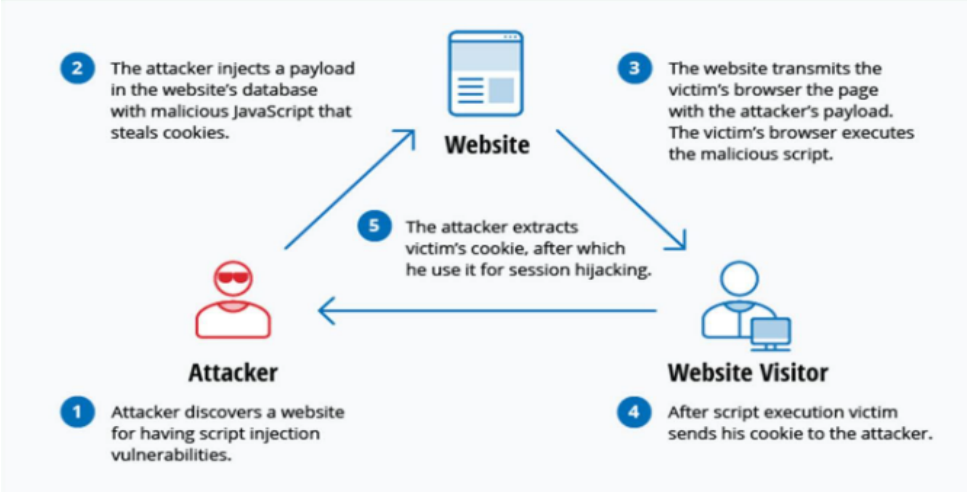


Fig.13 (XSS attack)

Cross-Site Scripting attacks involve injecting malicious scripts into web pages viewed by other users. Attackers exploit vulnerabilities in a web application to insert scripts that can be executed in the context of another user's browser. This could lead to the theft of sensitive information or unauthorized actions on behalf of the user. Employing thorough input validation and output encoding is essential to mitigate the risk of XSS attacks.

2.4 Objectives:

The main objective of our chat application is to improve the user experience and make the communication easier and effective. We have focused on the following topics-

2.4.1 Easy Login:

The application offers a hassle-free login experience, requiring users to input only their name. Upon registration, each user is assigned a unique IP address, streamlining the login process for swift and efficient access to the chat platform.

2.4.2 Global Connection:

The chat application leverages the power of the internet, fostering global connectivity. Users can seamlessly connect with individuals from around the world, transcending geographical boundaries and enabling diverse and widespread communication.

2.4.3 Notifications:

The application provides timely and relevant notifications to keep users informed about new messages, mentions, or other important activities. This feature enhances user engagement and ensures that participants stay connected and up-to-date with ongoing conversations.

2.4.4 Profile Customization:

Users have the flexibility to personalize their profiles according to their preferences. This feature allows individuals to add a unique touch to their online identity, enhancing the overall user experience and fostering a sense of individuality within the chat community.

2.4.5 High Security Measures:

The chat application prioritizes security at the highest level, implementing robust measures to safeguard user data and communications. Through advanced encryption protocols and stringent access controls, the platform ensures a secure environment, protecting users from potential threats and unauthorized access.

2.4.6 Privacy Assurance:

The chat application prioritizes user privacy by implementing stringent measures to protect personal information. Users can communicate with confidence, knowing that their conversations are kept private and secure. The platform adheres to privacy standards, offering a trustworthy environment for users to engage in discussions without concerns about unauthorized access or data breaches.

2.4.7 Text, Audio, and Video Chat:

The application provides a versatile communication experience by supporting text, audio, and video chat functionalities. Users can seamlessly switch between these modes, allowing for dynamic and engaging conversations tailored to their preferences. Whether exchanging messages, participating in voice calls, or having face-to-face video interactions, the platform offers a comprehensive suite of communication options.

2.4.8 End-to-End Encryption:

End-to-End Encryption (E2EE) is a critical and sophisticated security feature embedded in the chat application, playing a pivotal role in safeguarding the privacy and confidentiality of user communications.

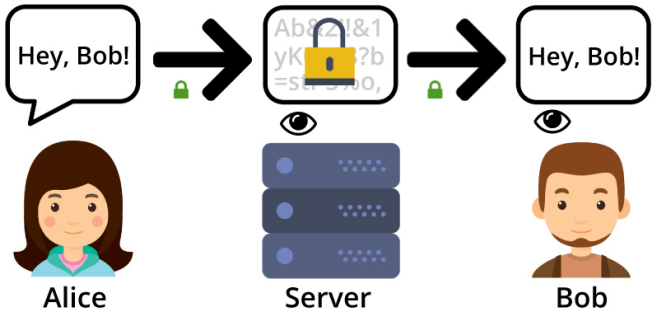


Fig.14 (E2EE)

This encryption mechanism operates on the fundamental principle of securing the transmission of messages by encrypting them on the sender's device and allowing decryption solely on the recipient's end. The implementation involves the generation of a unique pair of cryptographic keys for each user: a public key, shared openly, and a private key, exclusively stored on the user's

device. This dual-key system ensures that even if the communication channel is compromised, only the designated recipient possessing the private key can access and decipher the messages. E2EE stands as a robust shield against unauthorized access, eavesdropping, and interception, providing users with the assurance that their conversations, shared content, and data are kept private, secure, and inaccessible to any third-party entities.[19] This feature is instrumental in creating a trustworthy and secure environment, reinforcing the commitment to user privacy within the chat application.

Chapter 3: Design flow/Process:

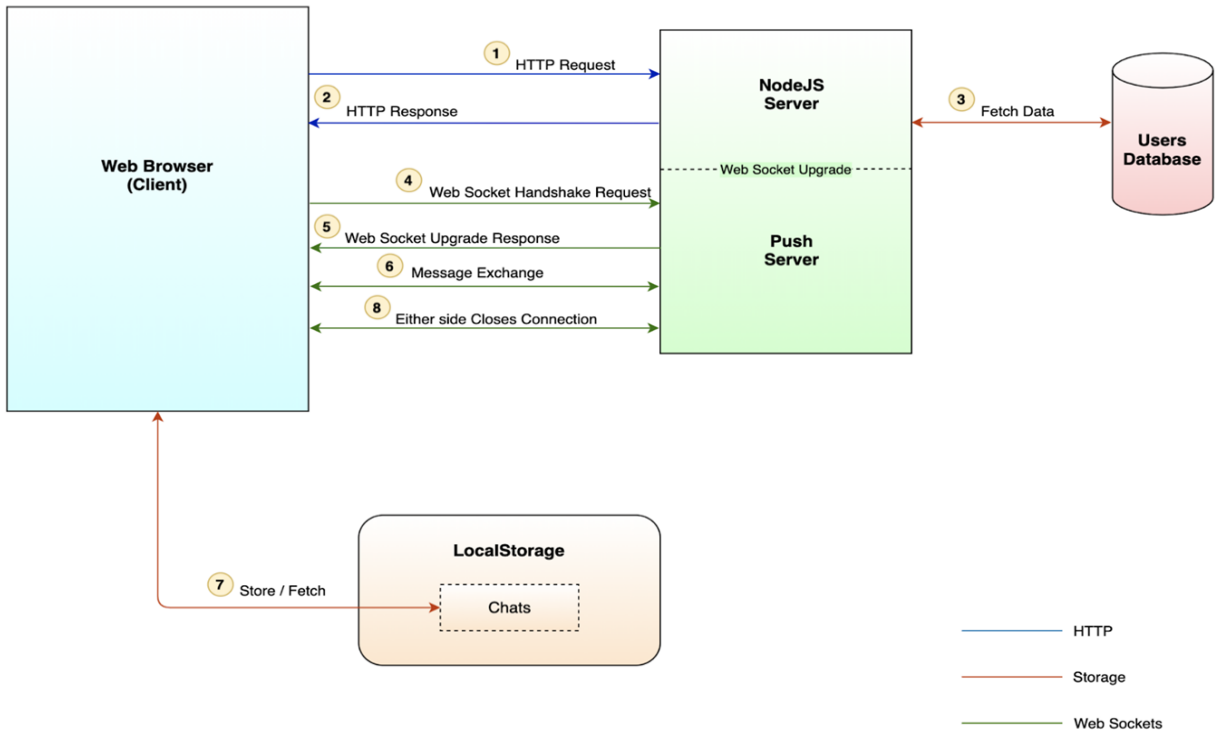


Fig.15 (Design of the project)

Real-time chat platform is a versatile tool that provides a diverse range of essential features and functions aimed at enriching communication and collaborative experiences. The core of these applications is instant messaging, allowing users to participate in live, text-based conversations. Users can extend their interactions beyond text by seamlessly sharing a variety of multimedia

content, including photos, videos, and documents. Group chat capabilities are fundamental, enabling multiple users to engage in a collective conversation. The inclusion of appealing stickers enhances the expression of emotions, while users can monitor message delivery and acknowledgments. Robust security measures, such as high-level message encryption, guarantee the confidentiality of conversations. Timely push notifications keep users informed about incoming messages, and contact lists simplify the process of locating and communicating with friends and colleagues. For added convenience, users have the option to create backups of their chat history. The user interface is intentionally crafted to be user-friendly and intuitive, ensuring effortless navigation. Users can also engage in slight text customization to personalize their conversations. Cosmetic choices and customizable themes allow users to adjust the app's appearance according to their preferences. Effective search functions facilitate the rapid retrieval

of specific information from chat histories. Integration with additional applications and services, coupled with the presence of automated bots, elevates the overall utility of these platforms. Regardless of whether they are used for personal or professional purposes, the versatility and cross-platform compatibility of these chat applications establish them as invaluable tools for modern communication and collaboration.[8]

3.1 Design flow:

Designing the flow for a chat sphere involves creating a user experience that is intuitive, responsive, and efficient.

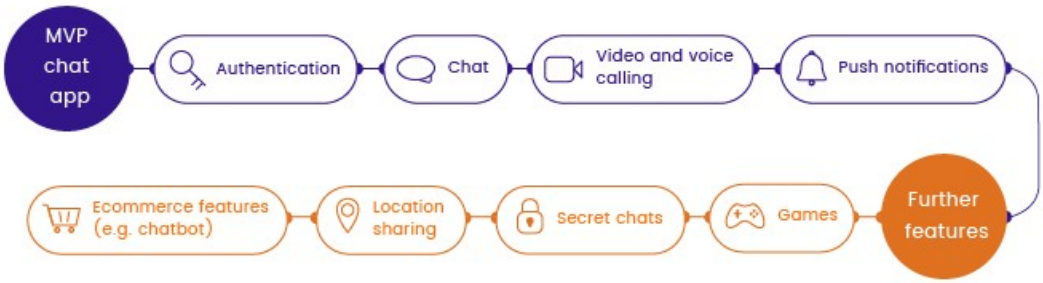


Fig.16 (Model flow)

3.1.1 User Registration and Authentication:

Users initiate their journey by either registering a new account or logging in with existing credentials. The system rigorously verifies user identity through multifaceted authentication mechanisms, ensuring a secure and trustworthy onboarding process.

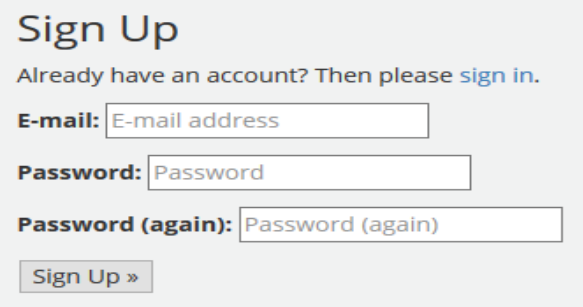


Fig.17 (Registration)

3.1.2 User Profile Setup:

Upon the inaugural login, users are prompted to craft their profiles, adding a personal touch to their digital presence. Beyond the basics like profile pictures and display names, users can

furnish additional details, fostering a sense of individuality and enhancing the overall personalization of their experience.

3.1.3 Contact Management:

The application simplifies networking by enabling users to effortlessly search for and add contacts to their chat list. Presence indicators, a subtle yet crucial feature, keep users informed about the online/offline status of their contacts, facilitating timely and context-aware communication.

3.1.4 Initiating Chat:

Users seamlessly transition into conversations by either selecting a contact for one-on-one interaction or creating dynamic group chats. The process involves not just naming the group but also actively curating the list of participants, ensuring a cohesive and inclusive chat environment.

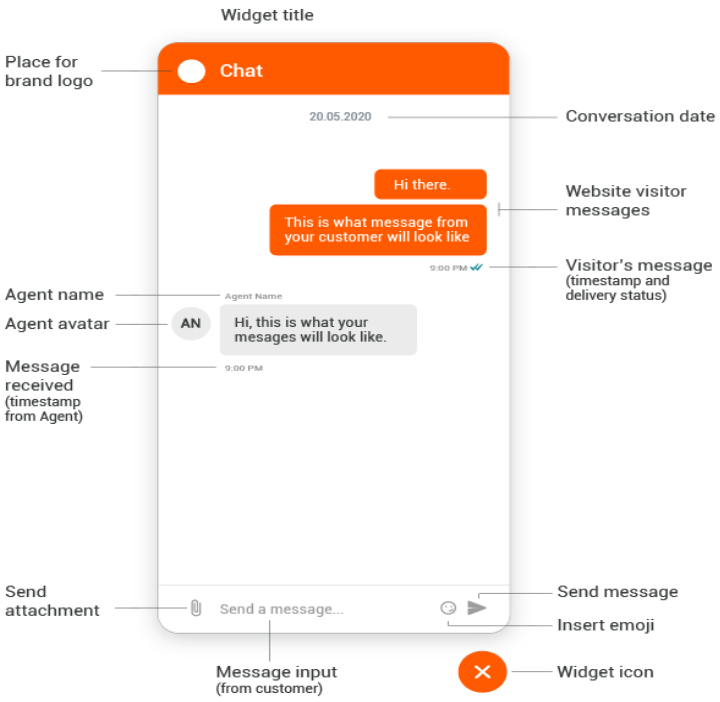


Fig.18 (Basic chat interface)

3.1.5 Real-Time Messaging:

The heart of the application lies in real-time messaging, where users engage in instantaneous communication. Messages flow seamlessly between users, creating an environment where conversations unfold organically. Push notifications serve as a conduit, promptly notifying users of incoming messages and ensuring a responsive interaction cycle.

3.1.6 Multimedia Sharing:

Beyond text, users have the expressive freedom to share multimedia content, enriching conversations with images, videos, documents, and voice messages. This feature transcends conventional messaging, fostering a more dynamic and engaging form of communication.

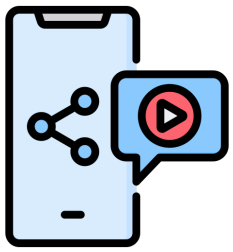


Fig.19 (Multimedia)

3.1.7 Group Chat Features:

Group chat functionalities empower users with administrative control, facilitating smooth management of participants. Admins enjoy additional privileges, providing a structured and organized group communication experience.

Fig.20 (Group chat)

3.1.8 Notifications and Alerts:

Push notifications serve as a constant conduit, keeping users abreast of new messages, even when the application is running in the background. Additionally, alerts promptly notify users of significant events, such as incoming contact requests or important announcements within group chats.

Fig.21 (Notifications)

3.1.9 Settings and Customization:

Acknowledging user preferences, the application grants access to a spectrum of settings for customization. These options extend beyond mere aesthetics, encompassing notification preferences, theme selection, and privacy settings, allowing users to tailor the application to suit their individual preferences.

3.1.10 Security Features:

Fig.22 (Security)

The bedrock of user trust lies in robust security measures. The application employs end-to-end encryption, safeguarding user data and privacy. These security features create a secure environment, fostering open and confidential communication.

3.1.11 Logout and Account Management:

Recognizing the need for user control, the application allows users to gracefully exit by logging out or manage their account settings. This extends to actions such as changing passwords or even deleting accounts, ensuring users have agency over their digital presence.

3.1.12 Cross-Platform Accessibility:

Fig.23 (Cross platform accesibilty)

The application transcends device boundaries, offering a consistent and seamless experience across multiple platforms. Whether accessed via web browsers, desktop applications, or mobile devices, users can engage in uninterrupted conversations, contributing to the application's ubiquity.

3.2 Technologies Used:

In developing our chat sphere, a strategic combination of technologies and algorithms was crucial to achieving optimal performance. The backend relies on the WebSocket protocol, chosen for its bidirectional communication capabilities that facilitate real-time data exchange with minimal latency. Node.js, known for its lightweight and event-driven nature, was selected to handle asynchronous communication efficiently. Express.js was employed as a robust framework to streamline server-side operations.

For data management, MongoDB, a NoSQL database, was chosen for its flexibility and scalability, making it well-suited for the storage and retrieval of chat messages. Additionally, Redis was integrated for caching and pub/sub capabilities, enhancing real-time message distribution while alleviating the load on the primary database.

On the frontend, React.js was employed for its component-based architecture, simplifying UI management and updates. The use of Socket.io on the frontend abstracts WebSocket implementation, ensuring seamless real-time communication. This combination of technologies

and algorithms is aimed at delivering a responsive and efficient real-time chat experience for users, striking a balance between performance and scalability.

The technological framework supporting these applications encompasses MongoDB, along with HTML, CSS, JavaScript libraries, and WEB-rtc, all meticulously integrated to guarantee seamless real-time performance of the application.

3.2.1 MongoDB:

MongoDB stands out as a source-available, cross-platform document-oriented database program, earning its classification as a NoSQL database. Developed by MongoDB Inc., it leverages JSON-like documents, offering the flexibility of optional schemas. This architecture is particularly advantageous in handling diverse and evolving data structures. As a member of the MACH Alliance, MongoDB aligns with the principles of Microservices, API-first, Cloud-native,

and Headless technologies. Its design focuses on scalability and efficiency, making it suitable for a wide range of applications, from small-scale projects to large enterprises.[13]

MongoDB's approach to data management is distinctive, as it does away with the traditional table-based relational database structure. Instead, it organizes data in flexible, JSON-like BSON (Binary JSON) documents, which can be nested and hierarchically structured. This schema-less model allows for dynamic adaptation to changing data requirements, a feature especially beneficial in dynamic and fast-paced development environments.

Fig.24 (MongoDB)

However, it's essential to note that MongoDB's licensing model has been a point of discussion. Current versions are licensed under the Server Side Public License (SSPL), a license that some organizations and distributions consider non-free. This aspect prompts considerations about compatibility and licensing requirements when integrating MongoDB into specific projects or environments. Despite the licensing discussions, MongoDB continues to be a popular choice due to its ability to handle large volumes of unstructured data, its horizontal scaling capabilities, and its compatibility with cloud-based deployments. Its integration with various programming languages and frameworks further enhances its appeal, making MongoDB a versatile and powerful option for developers and businesses seeking scalable and flexible database solutions.

Why use mongoDB?

The database serves as the foundational infrastructure for a chat application, playing a pivotal role in ensuring the seamless functioning of various critical components. It acts as the repository for user-related information, storing details like usernames, hashed passwords, and profile information, which is essential for user authentication during the login process. Furthermore, the database is instrumental in preserving the ongoing conversations within the application, encompassing a diverse range of messages, including text, multimedia content, and files. This message storage not only facilitates real-time communication but also enables users to retrieve their message history, contributing to a comprehensive and continuous user experience. Through efficient indexing, the database ensures quick and accurate retrieval of messages based on

parameters such as sender, receiver, timestamps, and chat room. Additionally, the database manages user presence, status, and relationships, tracking whether a user is online or offline and maintaining contact lists. This functionality is integral to the application's capability to inform users about the availability of their contacts and update statuses in real-time. Lastly, the database stores information related to notifications, managing aspects like unread messages and delivery status, contributing to an enriched and dynamic user notification experience. In essence, the database is the linchpin of the chat application, providing the necessary structure to store, retrieve, and manage user data and interactions.

3.2.2 WEB-RTC:

Fig.25 (WEB-RTC)

WebRTC, or Web Real-Time Communications, stands as an open-source initiative that facilitates real-time voice, text, and video communication capabilities among web browsers and various devices. It equips software developers with JavaScript-based Application Programming Interfaces (APIs) to establish peer-to-peer (P2P) communications seamlessly between web browsers and mobile applications, eliminating concerns about compatibility and support for audio, video, or text content.

The beauty of WebRTC lies in its ability to enable real-time data transfer without the need for custom interfaces, additional plugins, or specialized software for browser integration. With WebRTC, real-time audio and video communication become as simple as opening a webpage. The functioning of WebRTC involves the use of JavaScript, APIs, and Hypertext Markup Language to embed communication technologies directly into web browsers. Its design prioritizes user-friendliness and ease of implementation for audio, video, and data communication across browsers, compatible with most major web browsers. Key functionalities of WebRTC APIs encompass accessing and recording video, audio, and text data from devices, initiating, monitoring, and terminating P2P connections between devices via browsers, and

facilitating bidirectional data transfer through multiple channels.[20]

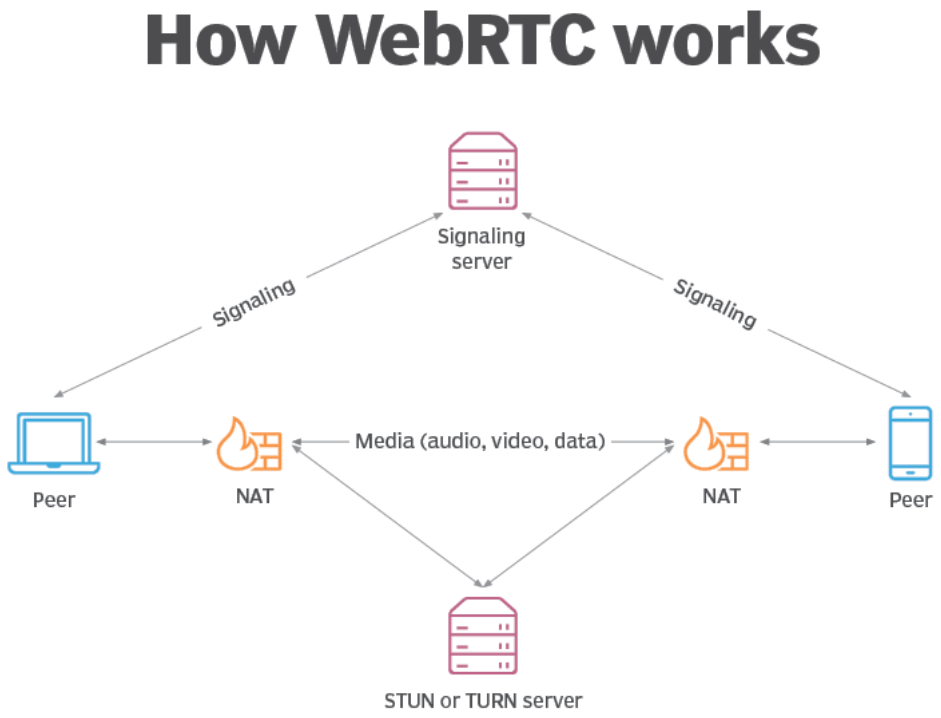


Fig.26 (WebRTC working)

A STUN (Session Traversal Utilities for NAT) server is a type of server used in VoIP (Voice over Internet Protocol) and other real-time communication systems to help clients behind firewalls or NAT (Network Address Translation) devices to connect with other clients.

Typically, WebRTC achieves real-time communication by directly transferring audio, video, and data between devices via P2P connections. However, in scenarios where users are on different Internet Protocol (IP) networks with Network Address Translation (NAT) firewalls that hinder Real-Time Communication (RTC), WebRTC collaborates with Session Traversal Utilities for NAT (STUN) servers. STUN servers translate IP addresses into public internet addresses, enabling the establishment of peer connections.

In cases where networks are highly restrictive, even a STUN server may not suffice. WebRTC, in such instances, pairs with a Traversal Using Relays around NAT (TURN) server. The TURN server acts as a relay, facilitating traffic between users and ensuring connection establishment. The Interactive Connectivity Establishment (ICE) protocol is employed to identify the optimal connection under these circumstances.

3.2.3 Two way authentication (2FA):

Two-factor authentication (2FA) serves as an additional layer that effectively verifies a user's identity, making unauthorized access considerably more challenging. In this method, users first enter their primary authentication credentials, such as a username and password. Subsequently, they are required to provide a secondary piece of identifying information.

Fig.27 (2FA)

The secondary factor is typically more intricate, involving something the legitimate user would have access to but unrelated to the specific system. Potential secondary factors include a one-time password generated by an authenticator app, a linked phone number or device capable of receiving push notifications or SMS codes, or a biometric element like fingerprint (Touch ID), facial recognition (Face ID), or voice recognition.

The implementation of 2FA significantly diminishes the risk of system or resource compromise, as it becomes highly improbable that an unauthorized user possesses both authentication factors. While the widespread adoption of two-factor authentication attests to its effectiveness in enhancing security, it's crucial to acknowledge that it may introduce some inconvenience for users, a factor that should be carefully considered during implementation.

3.2.4 Algorithms and tools:

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely

used today as it is a much stronger than DES and triple DES despite being harder to implement.

Fig.28 (AES-Advanced Encryption Standard)

AES is a block cipher, The key size can be 128/192/256 bits, Encrypts data in blocks of 128 bits each, That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

WebSocket is a computer communications protocol, providing simultaneous two-way communication channels over a single Transmission Control Protocol (TCP) connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011.

The current specification allowing web applications to use this protocol is known as WebSockets.[1] It is a living standard maintained by the WHATWG and a successor to The WebSocket API from the W3C.[2] WebSocket is distinct from the Hypertext Transfer Protocol (HTTP) used to serve most webpages. Both protocols are located at layer 7 in the OSI model and depend on TCP at layer 4.

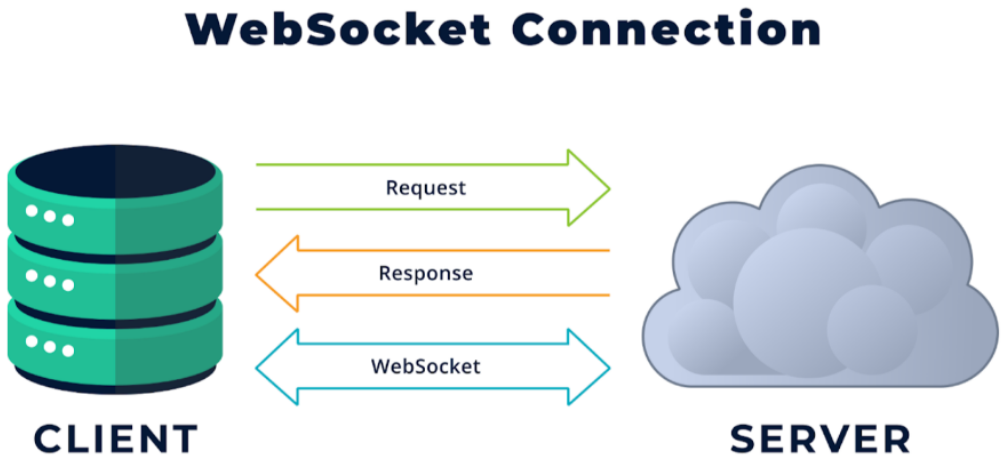


Fig.29 (Web socket connection)

WebSocket "is designed to work over HTTP ports 443 and 80 as well as to support HTTP proxies and intermediaries", thus making it compatible with HTTP. To achieve compatibility, the WebSocket handshake uses the HTTP Upgrade header[3] to change from the HTTP protocol to the WebSocket protocol.

The WebSocket protocol enables full-duplex interaction between a web browser (or other client application) and a web server with lower overhead than half-duplex alternatives such as HTTP polling, facilitating real-time data transfer from and to the server. This is made possible by providing a standardized way for the server to send content to the client without being first requested by the client, and allowing messages to be passed back and forth while keeping the connection open. In this way, a two-way ongoing conversation can take place between the client and the server.

The communications are usually done over TCP port number 443 (or 80 in the case of unsecured connections), which is beneficial for environments that block non-web Internet connections using a firewall. Additionally, WebSocket enables streams of messages on top of TCP. TCP alone deals with streams of bytes with no inherent concept of a message. Similar two-way browser–server communications have been achieved in non-standardized ways using stopgap technologies such as Comet or Adobe Flash Player.

3.3 Other important considerations:

3.5.1 Economic Issues:

From an economic perspective, the development team must weigh the costs associated with creating and maintaining the chat application. This includes considerations of software development, server infrastructure, and ongoing maintenance expenses. Exploring diverse monetization strategies, such as subscription plans or advertisements, is essential to ensure the application's financial viability.

3.5.2 Environmental Issues:

In addressing environmental concerns, the design should prioritize energy efficiency. This involves implementing features that minimize the application's energy consumption, particularly for users accessing the platform on battery-powered devices. Sustainable practices in server management, data storage, and overall infrastructure contribute to a reduced environmental footprint.

3.5.3 Health Issues:

The application's design should actively consider the well-being of its users. Features that promote healthy usage patterns, such as customizable notification settings, reminders for breaks, and options to limit screen time, play a crucial role. Additionally, addressing cybersecurity concerns becomes paramount to ensure the protection of user data and privacy, contributing to the overall health and safety of the user experience.

3.5.4 Manufacturability:

While not traditionally associated with software design, the concept of manufacturability in this context extends to the ease and efficiency of deploying and updating the application. Streamlining the deployment process and ensuring smooth updates contribute to the manufacturability of the software, allowing for efficient and scalable distribution.

3.5.5 Safety:

Safety considerations encompass both digital and physical aspects. Ensuring the security of user data through robust encryption measures is fundamental. Moreover, the application should promote safe usage practices, such as discouraging distracted driving and providing features that minimize potential risks associated with real-time communication.

3.5.6 Professional Issues:

Professional considerations involve adherence to industry standards and best practices. The design process should align with established norms, ensuring the application meets professional expectations for reliability, performance, and user experience. Professionalism extends to maintaining transparent communication with users and addressing issues promptly.

3.5.7 Ethical Issues:

Ethical considerations in the design process involve decisions regarding user privacy, data handling, and the responsible use of technology. Implementing clear and transparent privacy policies, obtaining user consent for data collection, and mitigating the risk of unethical use of the application are critical ethical considerations.

3.5.8 Social Issues:

The social impact of the application encompasses its role in fostering positive user interactions, combating issues like cyberbullying, and promoting inclusivity. Design choices should align with principles that contribute to a positive and supportive online community, fostering healthy social interactions.

3.5.9 Political Issues:

Political considerations involve navigating legal and regulatory landscapes. Compliance with data protection laws, ensuring user rights are respected, and addressing geopolitical concerns that may impact data storage and transmission are essential. Additionally, being mindful of the broader political context can contribute to the application's resilience in a dynamic regulatory environment.

Chapter 4: Results analysis and validation:

chat spheres have become indispensable tools in numerous contexts and industries, providing a wide array of advantages. In professional settings, they streamline business communication, both internally among team members and externally with customers and partners. These applications facilitate swift issue resolution in customer support, leading to higher customer satisfaction and often reducing support costs. Moreover, real-time chat apps are instrumental in fostering collaboration and teamwork among colleagues, allowing for real-time discussions, project collaboration, and efficient file sharing, making remote work increasingly feasible.

In the realm of education, real-time chat plays a pivotal role, especially in the era of e-learning. Educational institutions have harnessed this technology to facilitate online classes, student-teacher interactions, and virtual classroom experiences, bridging geographical gaps and making education more accessible. Furthermore, the healthcare sector benefits from real-time chat in the domains of telemedicine and remote patient monitoring. These applications enable instant communication between healthcare professionals and patients, offering valuable consultation and monitoring services.

The influence of real-time chat is also palpable in the realm of social networking. Popular social media platforms leverage real-time chat features, allowing users to engage in instant messaging with friends and contacts, enhancing the quality of interactions. In the e-commerce industry, these apps prove invaluable for addressing customer inquiries, order tracking, and providing support. Live chat support can significantly boost sales by offering real-time assistance during the shopping process, improving the overall shopping experience.

Entertainment and gaming sectors have witnessed a transformation due to chat spheres. [5] Gamers can now communicate with their fellow players, strategize in real time, and share their gaming experiences. This interactivity has fostered a vibrant gaming community, connecting players globally. Streaming services also make use of real-time chat to engage with their audiences, offering features for instant interactions, questions, and discussions during live broadcasts and online events.

Event management and coordination have become more efficient thanks to chat spheres. Event organizers and participants alike utilize these apps for real-time updates, discussions, and seamless coordination during events, conferences, and gatherings. Last but not least, in emergency response situations and disaster management, real-time chat plays a critical role. It facilitates the coordination of efforts, the sharing of vital information, and the provision of assistance during critical situations, saving

lives and ensuring effective response.

In sum, chat spheres have emerged as versatile and transformative tools, enhancing communication and collaboration across various domains. They contribute to improved efficiency, accessibility, and user experiences, ultimately reshaping the way people interact and connect in an increasingly interconnected world.

Fig.30 (RTCA Pic_1)

Fig.31 (RTCA Pic_2)

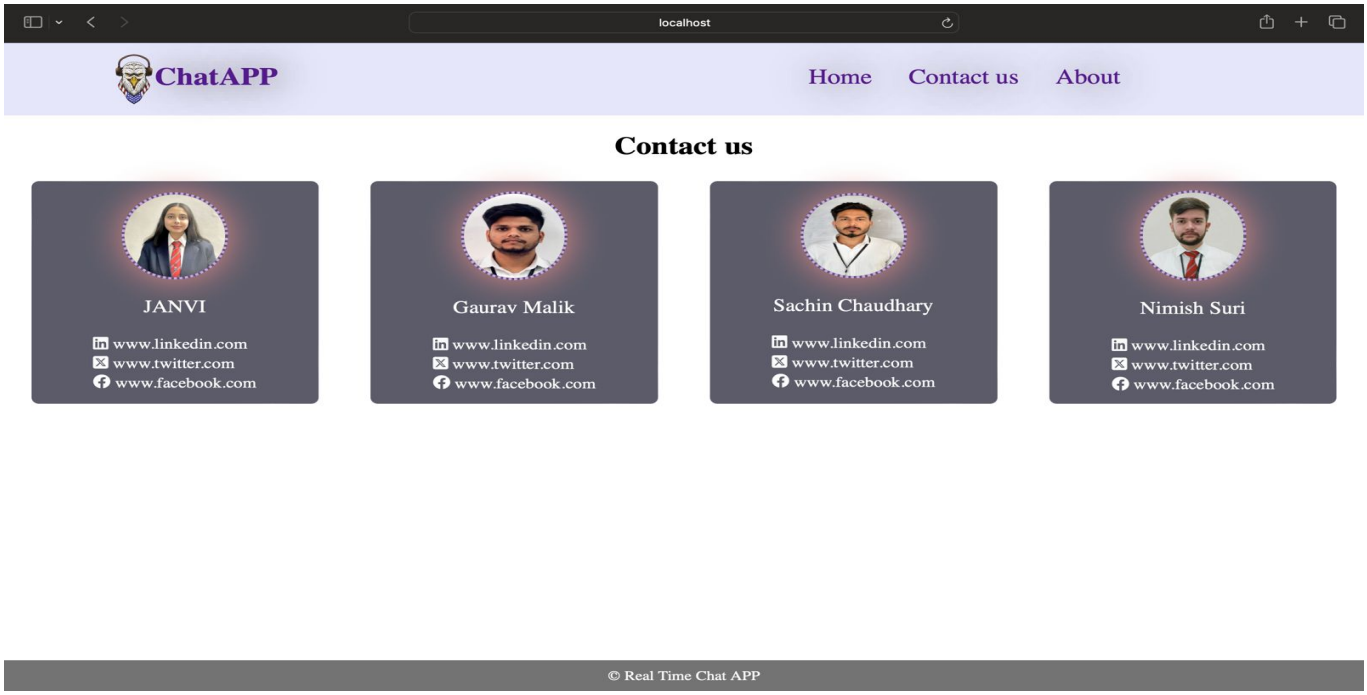


Fig.32 (RTCA Pic_3)

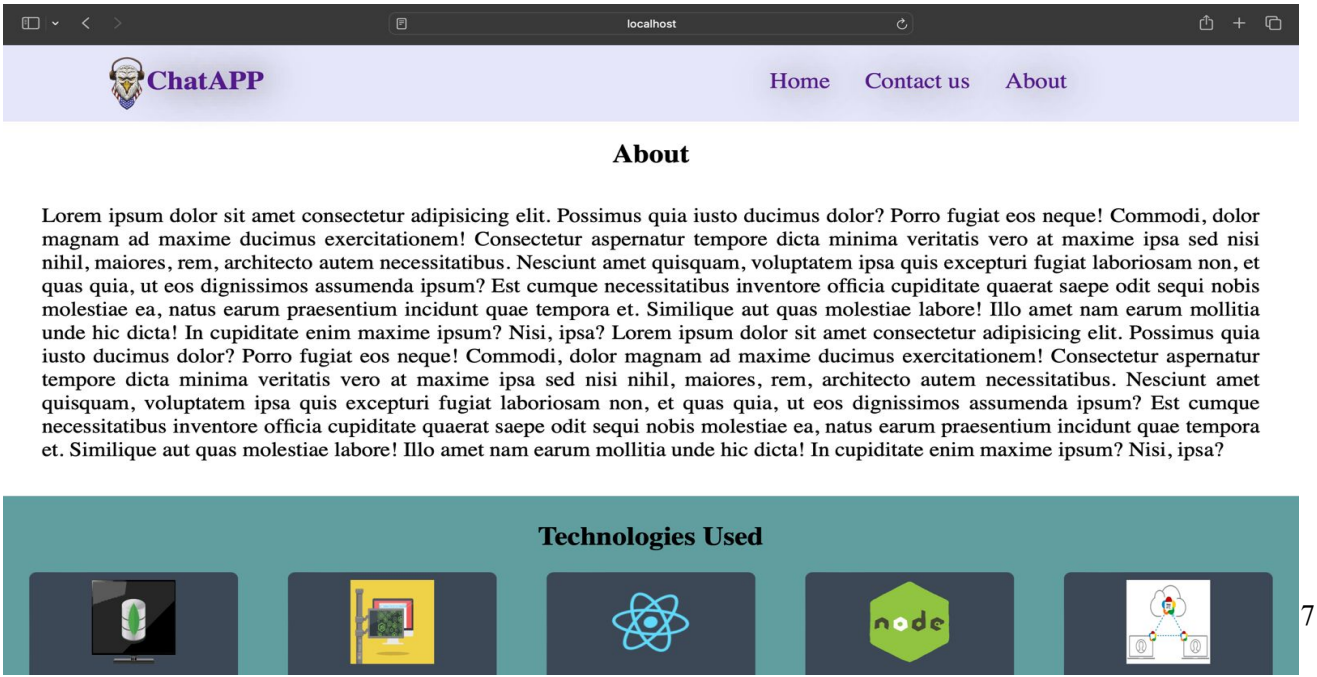


Fig.33 (RTCA Pic_4)

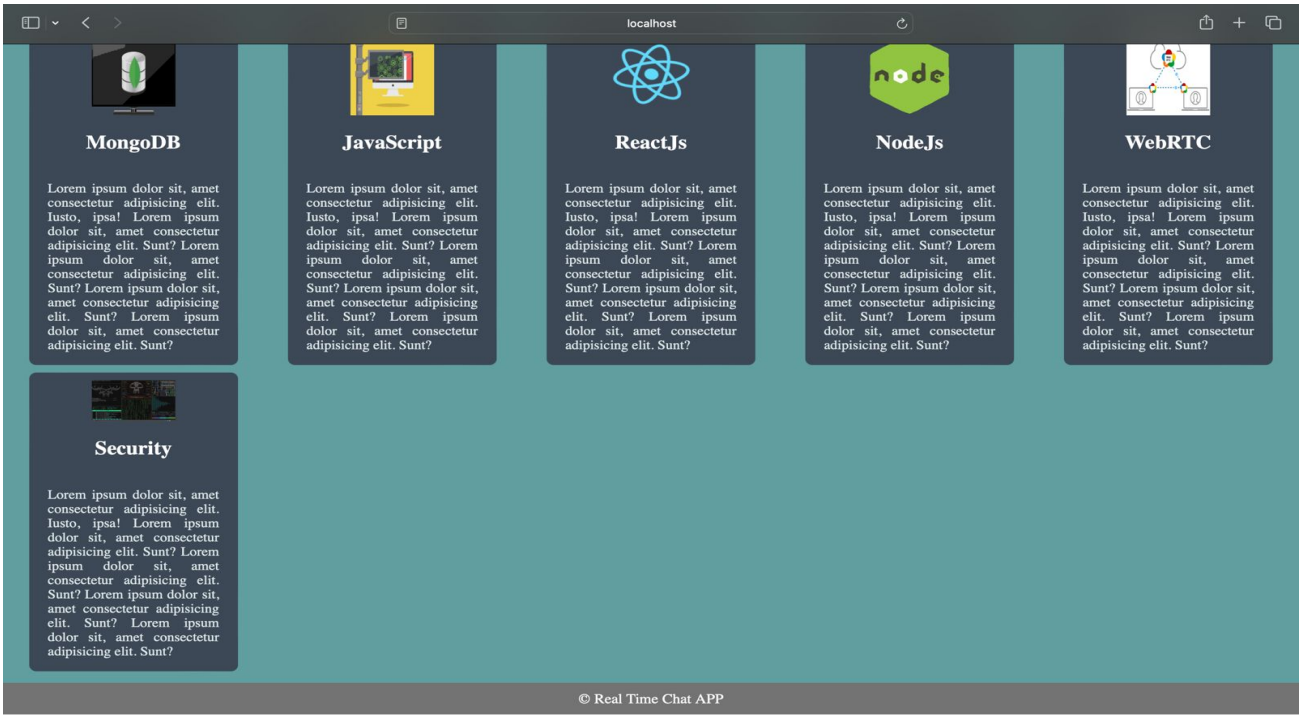


Fig.34 (RTCA Pic_5)

Chapter 5: Conclusion and future work:

In conclusion, developing a chat application using the MERN stack has been a challenging but rewarding experience. The use of MongoDB, Express, React, and Node.js provides a powerful and flexible framework for creating real-time communication and collaboration solutions that can be tailored to meet a wide range of use cases and industries. The app has been designed with user experience in mind, and features such as real-time message updates and user authentication have been implemented to provide a seamless and secure communication experience. The scalability and robustness of the MERN stack ensure that the app can handle a high volume of users and messages without compromising on performance. Going forward, there are many opportunities for further development and improvement of the app. This includes adding new features such as video and voice chat, integrating with other applications and platforms, and enhancing the user interface to make it more intuitive and user-friendly. Overall, the chat app developed using the MERN stack represents a significant achievement in the field of real-time communication and collaboration, and has the potential to revolutionize the way people connect and communicate online.

In conclusion, the development and implementation of the chat sphere have proven to be a significant milestone in enhancing communication and collaboration. This report has provided a comprehensive overview of the key components, features, challenges, and solutions associated with the creation of a chat sphere.

The real-time nature of the application has addressed the growing need for instantaneous communication in various sectors, including business, education, and social interactions. Users can now exchange messages, share multimedia content, and collaborate seamlessly, fostering a sense of connectivity in an increasingly digital world.

One of the major strengths of the application lies in its user-friendly interface and intuitive design. The incorporation of multimedia features, such as image and file sharing, has enriched the user experience, making the application versatile and adaptable to a wide range of communication needs. The responsive design ensures a consistent and enjoyable experience across different devices, catering to the diverse preferences of users.

Security and privacy concerns have been paramount in the development process. The implementation of end-to-end encryption ensures that user data and conversations remain confidential and secure. Additionally, robust authentication mechanisms have been integrated to safeguard against unauthorized access and protect user identities.

However, the journey was not without its challenges. Technical issues, such as scalability and server load, posed significant hurdles during the development phase. The team had to employ innovative solutions and optimize the backend infrastructure to ensure a smooth and reliable user experience, even during peak usage periods.

Moreover, the constant evolution of technology and user expectations necessitates ongoing updates and improvements to the application. Regular maintenance, bug fixes, and feature enhancements will be crucial to staying competitive in the dynamic landscape of real-time communication applications.

Looking ahead, the chat sphere is poised to play a pivotal role in shaping the future of communication. As technology continues to advance, there is immense potential for further integration with emerging technologies such as artificial intelligence and virtual reality, opening new avenues for enriched user experiences.

The future scope of chat spheres is poised for a transformative evolution, marked by a nuanced integration of cutting-edge technologies. Artificial Intelligence (AI) will play a central role in the advancement of these applications, with the emergence of highly sophisticated chatbots. These intelligent entities will transcend mere automation, offering personalized interactions based on intricate analyses of user behavior and preferences. Natural Language Processing (NLP) enhancements will contribute to a more contextually aware communication environment, allowing for nuanced and meaningful conversations.

Furthermore, the integration of Augmented Reality (AR) and Virtual Reality (VR) technologies will redefine the user experience. chat spheres will move beyond traditional messaging interfaces to offer immersive communication spaces. Users can anticipate virtual meeting rooms and collaborative environments that simulate physical interactions, ushering in a new era of engagement and connectivity. Additionally, AR elements within chat interfaces will enable interactive and creative multimedia sharing, introducing augmented reality filters and effects for a dynamic and personalized communication experience.[18]

Multimedia features within these applications will undergo continuous refinement, with an emphasis on delivering high-quality and dynamic content. Video streaming capabilities will reach new heights, incorporating 3D image rendering for a more visually engaging experience. Augmented reality filters will not only serve as a means of creative expression but also as a tool for enhancing interactive multimedia messaging.

Cross-platform compatibility will be a cornerstone of future developments. chat spheres will strive for seamless integration across a diverse range of operating systems and devices, ensuring a consistent and user-friendly experience for individuals across different platforms. As these applications continue to evolve, their future lies in the dynamic fusion of innovation, adaptability, and a commitment to providing users with increasingly sophisticated and immersive communication experiences.

REFERENCES:

- **Masiello Eric. (2017). "Mastering React Native. January 11, 2017. This book is a comprehensive guide to building mobile applications using React Native.**

- **Naimul Islam Naim. "ReactJS: An Open-Source JavaScript library for front-end development. Metropolia University of Applied Sciences. This article provides an overview of ReactJS and its key features for front-end web development.**

- **Stefanov Stoyan, editor. (2016). "React: Up and Running: Building web Applications. First Edition; 2016. This book is a beginner-friendly introduction to React, covering its core concepts and providing practical examples for building web applications.**

- **Horton Adam, Vice Ryan. (2016). "Mastering React; February 23; 2016. This book provides a comprehensive guide to React, covering its core concepts, practical examples, and advanced techniques for building complex applications.**

- **Alex Kondov. "Express Architecture Review. This article provides a review of the architecture of Express.js, a popular web framework for building Node.js applications.**

- **Express.js documentation. This documentation provides a comprehensive guide to building web applications using Express.js.**

- **Adam Horton. "Node.js vs Python: What to Choose. This article provides a comparison of Node.js and Python for web development, highlighting their strengths and weaknesses.**

- **Node.js documentation. This documentation provides a comprehensive guide to building server-side applications using Node.js. International Journal of Scientific Research in Science, Engineering and Technology | www.ijsrset.com | Vol 10 | Issue 2 Ms. Archana Nikose et al Int J Sci Res Sci Eng Technol, March-April-2023, 10 (2): 496-501 501**

- **VSChart. This website provides a comparison of various programming languages and frameworks based on popularity, community support, and other factors.**

- **MongoDB documentation. This documentation provides a comprehensive guide to using MongoDB, a popular NoSQL database for building web applications.**

- **The paper by Lakshmi Prasanna Chitra and Ravikanth Satapathy aims to compare the performance of Node.js and traditional web servers, specifically Internet Information Services (IIS), in optimizing web application development. The authors conducted various tests to evaluate the performance of both platforms and determine which one is better for developing high-performance web applications.**

- **React vs Angular: Key Differences. This article provides a comparison of React and Angular, two popular front-end frameworks for building web applications.**

- **Al-Riyami, SS and K.G. Paterson, 2003. Uncertified public key cryptography. Methods for the Ninth World Theoretical Conference furthermore, Use of Cryptology and Information Security, November 30-Dec. 4, Springer Berlin Heidelberg, Taiwan, pages: 452-473. DOI: 10.1007/978-3-540-40061-5_29 Azab, A., P. Watters and R. Layton, 2012.**

- **Matches the Skype Network Traffic Forensics. 3 Internet Criminal Procedures and Trusted Computer Workshop, October 29-30, IEEE Xplore Press, Ballarat, pages: 19-27. Segment: 10.1109/CTC.2012.14 Bardis, N.G. furthermore, K. Ntaikos, 2008.**

- **Building security AES cryptographic-based visit application calculation and key administration. Methodology for tenth World WSEAS Conference on Mathematics Methods, Numeracy Methods and Intelligent Systems, (TIS '08), ACM, USA, pages: 486-49**

- **Horton Adam. Vice Ryan, author. Mastering React; February 23; 2016. Accessed 1 Jan 2022**

- **International Journal of Engineering Research & Technology Published by: ijert.org vol. 10 Issue 06, June-2021. Performance Optimization using MERN stack on Web Application.**

- **Lakshmi Prasanna Chitra, Ravikanth Satapathy Department Of Computer Science Department Of Computer Science GITAM University Visakhapatnam, India Performance Comparison and Evaluation Of Node.Js And Traditional Web Server (IIS)**

- **2020 3rd International Conference on Computer and Informatics Engineering (IC2IE) 978-1-7281-8247-6/20/$31.00 ©2020 IEEE 261 Online Integration of SQL and No-SQL Databases using RestAPIs: A Case on 2 furniture e-Commerce Sites**

- **Building a Real-Time Chat Application with React and Firebase" by Leigh Halliday**

- **React Native Chat: Build a Chat App and Learn React Native and Firebase"**

by Konstantin Shkut.

- **PL/SQLBy Ivan Bayross.**

APPENDICES:

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import Root from "./components/root";
import Contactus from "./components/contactus";
import About from "./components/about";
import Home from "./components/home";
import './components/App.css'
import ErrorPage from "./components/error-page";

import { library } from '@fortawesome/fontawesome-svg-core'
import { fas } from '@fortawesome/free-solid-svg-icons'
import { faTwitter, faLinkedin, faFontAwesome, faSquareXTwitter, faFacebook } from '@fortawesome/free-brands-svg-icons'

import { createBrowserRouter, RouterProvider } from "react-router-dom";

library.add(fas, faTwitter,faLinkedin, faFontAwesome, faSquareXTwitter, faFacebook)
const router = createBrowserRouter([
```

```
  {
    path: "/",
    element: <Root />,
    errorElement: <ErrorPage/>,
    children: [
     {
       path: "/",
       element: <Home />,
     },{
       path: "about",
       element: <About />,
     },{
       path: "contactus",
       element: <Contactus />,
     },
    ],
  }

]);


const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <RouterProvider router={router}/>
```

);

Contactus.js

```
import React from 'react'
import { FontAwesomeIcon } from '@fortawesome/react-fontawesome'
import { Link } from 'react-router-dom'

const Contactus = () => {
   return (
      <div style={{ textAlign: 'center' }}>
         <h1>Contact us
         </h1>
         <div className="container">

            <div className="contact-d">
               <img src="../images/janvi.jpeg" alt="" />
               <p>JANVI</p>
               <div className='contactLinks'>
                  <Link className=' linkedin'>
                              <FontAwesomeIcon icon="fa-brands fa-linkedin"
style={{ color: "#fafafa", }} />
                     {'  '}www.linkedin.com
                  </Link>
```

```
                <Link className=' twitter'>
                                        <FontAwesomeIcon  icon="fa-brands
fa-square-x-twitter" style={{ color: "#ffffff", }} />
                  {'  '}www.twitter.com
                </Link>
                <Link className=' facebook'>
                        <FontAwesomeIcon icon="fa-brands fa-facebook"
style={{ color: "#ffffff", }} />
                  {'  '}www.facebook.com
                </Link>
              </div>
            </div>
            <div className="contact-d">
              <img src="../images/gaurav.jpeg" alt="" />
              <p>Gaurav Malik</p>
              <div className='contactLinks'>
                <Link className=' linkedin'>
                        <FontAwesomeIcon icon="fa-brands fa-linkedin"
style={{ color: "#fafafa", }} />
                  {'  '}www.linkedin.com
                </Link>
                <Link className=' twitter'>
                                        <FontAwesomeIcon  icon="fa-brands
fa-square-x-twitter" style={{ color: "#ffffff", }} />
```

```
                {'  '}www.twitter.com
              </Link>
              <Link className=' facebook'>
                    <FontAwesomeIcon icon="fa-brands fa-facebook"
style={{ color: "#ffffff", }} />
                {'  '}www.facebook.com
              </Link>
            </div>
          </div>
          <div className="contact-d">
            <img src="../images/sachin.jpeg" alt="" />
            <p>Sachin Chaudhary</p>
            <div className='contactLinks'>
              <Link className=' linkedin'>
                    <FontAwesomeIcon icon="fa-brands fa-linkedin"
style={{ color: "#fafafa", }} />
                {'  '}www.linkedin.com
              </Link>
              <Link className=' twitter'>
                                  <FontAwesomeIcon  icon="fa-brands
fa-square-x-twitter" style={{ color: "#ffffff", }} />
                {'  '}www.twitter.com
              </Link>
              <Link className=' facebook'>
```

```
                        <FontAwesomeIcon icon="fa-brands fa-facebook"
style={{ color: "#ffffff", }} />
                    {'  '}www.facebook.com
                </Link>
            </div>
        </div>
        <div className="contact-d">
            <img src="../images/img2.jpeg" alt="" />
            <p>Nimish Suri</p>
            <div className='contactLinks'>
                                        <Link  className='  linkedin'
to={'https://www.linkedin.com/in/nimish-suri-837348200/'}>
                        <FontAwesomeIcon icon="fa-brands fa-linkedin"
style={{ color: "#fafafa", }} />
                    {'  '}www.linkedin.com
                </Link>
                <Link className=' twitter'>
                                    <FontAwesomeIcon  icon="fa-brands
fa-square-x-twitter" style={{ color: "#ffffff", }} />
                    {'  '}www.twitter.com
                </Link>
                <Link className=' facebook'>
                        <FontAwesomeIcon icon="fa-brands fa-facebook"
style={{ color: "#ffffff", }} />
```

```
                    {'  '}www.facebook.com
                  </Link>
                </div>
              </div>
            </div>
          </div>
    )
}

export default Contactus
```

Root.js

```
import React from 'react'
import { Link,Outlet } from 'react-router-dom'
import Footer from './footer'


const Root = () => {

  return (
     <>
   <div style={{ display: 'flex', backgroundColor: 'lavender' }}>
    <div style={{     marginRight: '26%', padding: '1% 0 0% 8%' }}>
```

```
      <Link className='navlink' style={{display:'flex'}} to='/'>
        <img src="../images/pic.png" alt="logo.png" />
        <h3>ChatAPP</h3>
      </Link>
    </div>
    <nav className='n'>
      <Link className='l navlink' to='/'>Home</Link>
      <Link className='l navlink' to='/contactus'>Contact us</Link>
      <Link className='l navlink' to='/about'>About</Link>
    </nav>
  </div>
  <Outlet/>
  <Footer/>
  </>

 )
}

export default Root
```

App.css

```
body{
 margin: 0;
```

```
  user-select: none;
}
#root{
  margin: 0;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
}
.n {
  color: black;
  padding: 2%;
  display: flex;
  justify-content: center;
  width: 45%;
}

.l {
  padding: 0% 3%;
  font-size: calc(0.75*(1.5vh + 1.5vw));
  text-decoration: none;
}

.navlink{
  font-size: calc(1.2*(1vh + 1vw));
```

```
  text-shadow:2px 1px 42px #000000c7;
  text-decoration: none;
}
a h3{
  margin-top: 6.9%;
}
a img{
  width: calc(1.95*(1vh + 1vw));
  height:calc(2.95*(1vh + 1vw));
  float:left;
}
.navlink:hover{
  color: #c75c6a;
  text-shadow: 1px 1px 30px #000;
  transition: 900ms;
}
.container{
  padding: 0 2%;
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(15%, 2fr));
  column-gap: 4%;
}
.about-d{
  display: flex;
```

```
  flex-direction: column;
  align-items: center;
  justify-content: center;
  background-color: rgba(51, 51, 69, 0.804);
  color: #fff;
  border-radius: .5em;
  text-align: justify;
  margin: 2% 0;
  padding-top: 4%;
}
.about-d p{
  padding: 0% 9%;
  color: rgb(223, 233, 233);
}
.about-img{
  width: 40%;
  height: 40%;
  object-fit:fill;

}
#technologyUsed{
  background-color: cadetblue;
  padding : 0.7% 0;
```

```
}
@keyframes rotate{
  to{ transform: rotate(360deg); }
}
.contact-d{
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  background-color: rgba(51, 51, 69, 0.804);
  color: #fff;
  border-radius: .5em;
  text-align: justify;
  margin: 2% 0;
  padding: 4% 0;
}
.blank{
  background-color: #fff;
}
.contact-d img{
  width: 35%;
  object-fit: cover;
  border-radius: 5rem;
  border: 3px dotted rgb(208, 163, 234);
```

```
  box-shadow: 1px 1px 40px rgb(224, 138, 138);
}
.contact-d p{
  font-size: 1.6vw;
}
.contactLinks{
  display: flex;
  flex-direction: column;
}
.contactLinks a{
  text-decoration: none;
  font-size: 1.3vw;
  color:white;
  margin: 1% 0%;
}
```

ErrorPage.js

```
import { useRouteError } from "react-router-dom";
import Navbar from "./root";

export default function ErrorPage() {
  const error = useRouteError();
  console.error("nimish",error);
```

```
  console.log('hello')

  return (

    <div style={{textAlign:'center'}}>
      <Navbar/>
     <h1>Oops!</h1>
     <p>Sorry, an unexpected error has occurred.</p>
     <p>
      <i>{error.statusText || error.message}</i>
     </p>
    </div>
  );
}
```

Package.json

```
{
  "name": "chatapp",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@fortawesome/fontawesome-svg-core": "^6.4.2",
    "@fortawesome/free-brands-svg-icons": "^6.4.2",
```

```
  "@fortawesome/free-regular-svg-icons": "^6.4.2",
  "@fortawesome/free-solid-svg-icons": "^6.4.2",
  "@fortawesome/react-fontawesome": "^0.2.0",
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-loader-spinner": "^5.4.5",
  "react-router-dom": "^6.19.0",
  "react-scripts": "5.0.1",
  "web-vitals": "^2.1.4",
  "@babel/plugin-proposal-private-property-in-object":""
},
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
"eslintConfig": {
  "extends": [
    "react-app",
    "react-app/jest"
```

```
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  }
}
```

Home.jsx

```
import React from 'react'
import { useState } from "react";
import Loader from "./loader";



const Home = () => {
```

```
const [isLoading, setIsLoading] = useState(true);


setTimeout(() => {
   setIsLoading(false);
}, 10);
return (
   <>
      {isLoading ? (
         <div
            style={{
               display:'flex',
               justifyContent:'center',
               alignItems:'center',
               marginTop:'30vh'
            }}
         >
            <Loader />
         </div>
      ) : (
         <center>
            <h1>
               Welcome to the Home Page
            </h1>
         </center>
```

```
            )}

        </>
    )
}

export default Home
```