A PROJECT REPORT

on

“CHAT ENGINE”

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR’S DEGREE IN

COMPUTER SCIENCE AND ENGINEERING

BY

SUMIT KUMAR	**20051586**
AYUSHMAAN RATHOD	**20051693**
SAURABH ANIKET	**20051702**
ADESH PRATAP SINGH	**20051707**

UNDER THE GUIDANCE OF

DR. SOUMYA RANJAN MISHRA

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA - 751024

May 2023

A PROJECT REPORT

on

"CHAT ENGINE"

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award of

BACHELOR'S DEGREE IN
COMPUTER SCIENCE AND ENGINEERING

BY

SUMIT KUMAR	20051586
AYUSHMAAN RATHOD	20051693
SAURABH ANIKET	20051702
ADESH PRATAP SINGH	20051707

UNDER THE GUIDANCE OF

DR. SOUMYA RANJAN MISHRA

SCHOOL OF COMPUTER ENGINEERING
KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA -751024

May 2023

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is certify that the project entitled

“CHAT ENGINE“

submitted by

SUMIT KUMAR	20051586
AYUSHMAAN RATHOD	20051693
SAURABH ANIKET	20051702
ADESH PRATAP SINGH	20051707

is a record of bonafide work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering) at KIIT Deemed to be university, Bhubaneswar. This work is done during year 2022-2023, under our guidance.

Date: 4/5 /2023

Dr. Soumya Ranjan Mishra

Project Guide

Acknowledgements

I would like to extend my deepest appreciation and heartfelt thanks to **Dr. Soumya Ranjan Mishra** for his exceptional guidance and constant encouragement during the entirety of my project, CHAT ENGINE. From its conception to its completion, Dr. Soumya Ranjan Mishra expert advice and unfailing support were instrumental in helping me achieve my goals and exceed my own expectations.

Throughout the project, Dr. Soumya Ranjan Mishra served as an invaluable mentor, providing me with invaluable insights, strategies, and resources to help me navigate the challenges and complexities of my work. His extensive knowledge and expertise in the field were truly invaluable, and I could not have asked for a more dedicated or knowledgeable teacher to guide me through this process.

Dr.Soumya Ranjan Mishra's passion and enthusiasm for the subject matter were infectious and inspiring, and his unwavering support and encouragement kept me motivated and focused even in the face of setbacks or difficulties. I am truly grateful to have had the opportunity to work with such an outstanding teacher, and I know that the skills and knowledge I have gained under his guidance will serve me well in all of my future endeavors.

In conclusion, I would like to express my deepest thanks and gratitude to Dr. Soumya Ranjan Mishra for his expert guidance, patience, and support throughout the CHAT ENGINE project. I consider myself fortunate to have had the opportunity to work with such an exceptional mentor, and I am confident that his contributions have been pivotal to the success of my work.

SUMIT KUMAR

AYUSHMAAN RATHOD

SAURABH ANIKET

ADESH PRATAP SINGH

ABSTRACT

A real-time chatting app is a software application that enables instant messaging and communication between two or more individuals through the internet. The app allows users to send and receive messages in real-time, creating a virtual conversation experience that simulates face-to-face interaction. The app typically includes features such as typing indicators, read receipts, and multimedia support, allowing users to send images, videos, and audio messages. These apps often incorporate end-to-end encryption to ensure secure communication, and many also offer group chat functionality, allowing users to engage in group conversations. Real-time chatting apps have become increasingly popular in recent years due to the rise of remote work, social distancing measures, and the need for instant communication in our daily lives. These apps have revolutionized the way we communicate, making it easier than ever to stay in touch with friends, family, and colleagues around the world.

One of the primary benefits of real-time chatting apps is their convenience. Users can send and receive messages from anywhere, at any time, without having to wait for a response. This has made it easier for people to coordinate with one another, especially in situations where time is of the essence, such as emergencies or urgent work situations.

Another key feature of real-time chatting apps is their multimedia support. In addition to text messages, users can send and receive images, videos, and audio messages. This has opened up new avenues for communication, enabling users to share their experiences and emotions in a more expressive way.

Overall, real-time chatting apps have transformed the way we communicate, making it easier than ever to stay connected with those who matter most. As technology continues to evolve, we can expect these apps to become even more sophisticated, offering new features and capabilities that will further enhance the communication experience.

Contents

List of Figures

Chapter 1

Introduction

A chat engine project is a software application that utilizes natural language processing (NLP) and machine learning algorithms to enable users to interact with a computer system via text or voice-based messaging.

The main objective of a chat engine project is to provide users with an intuitive and conversational interface to accomplish specific tasks , such as answering customer inquiries , providing personalized recommendations , or automating customer support.

Current Need of the Project:-

Improved Communication: A chat engine can improve communication between team members, customers, and stakeholders by providing a centralized platform for messaging and collaboration.

Efficiency: A project chat engine can improve the efficiency of team communication by allowing team members to quickly and easily share updates, ideas, and feedback.

Real-Time Collaboration: A chat engine can facilitate real-time collaboration between team members, allowing them to work together on projects in real-time, regardless of their physical location.

Centralized Information: A chat engine can provide a centralized repository for project information, including project timelines, tasks, and deadlines.

Task Management: A chat engine can be used to manage tasks and track progress, making it easier for teams to stay on track and meet project deadlines.

Security: A project chat engine needs to be secure and protect sensitive project information, such as client data and financial information, from unauthorized access.

Chapter 2

Basic Concepts/ Literature Review

2.1) Socket.io

Socket.io is a JavaScript library that enables real-time, bidirectional and event-based communication between a client and a server. It works on the web and on mobile devices and can be used with any back-end programming language.

Here are some key features of Socket.io:

Real-time communication: Socket.io enables real-time, bidirectional communication between a client and a server, which means that data can be sent and received instantly.
Event-based communication: Socket.io uses an event-based model to facilitate communication between the client and server. Clients can send events to the server and vice versa, and events can have data payloads attached to them.
Room-based communication: Socket.io allows clients to join and leave "rooms" on the server, which enables broadcast-style communication between multiple clients.
Automatic re connection: Socket.io automatically attempts to reconnect to the server if the connection is lost, ensuring that real-time communication is maintained.
Cross-browser and cross-platform compatibility: Socket.io works on all major web browsers and mobile devices, and can be used with any backend programming language.

2.2) MongoDB

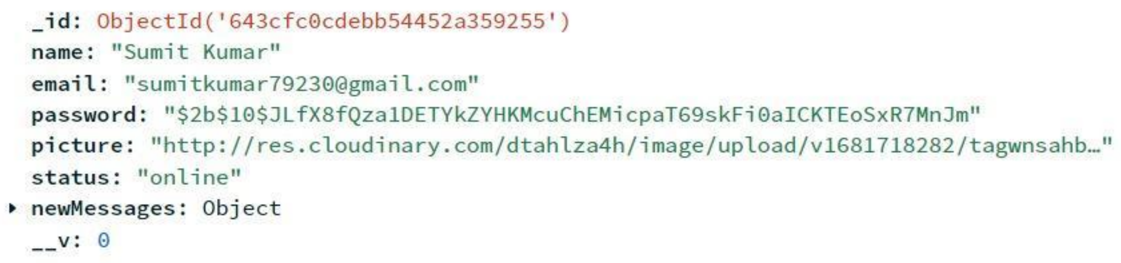
MongoDB can be used as a backend database for a chat engine. MongoDB is a popular NoSQL document database that can store data in flexible, JSON-like documents. This makes it well-suited for storing chat messages, user data, and other information related to a chat engine.

We can use MongoDB in many ways:

Storing chat messages: MongoDB can be used to store chat messages in a collection, with each document representing a single message. The document can include fields such as the message content, sender ID, recipient ID, timestamp, and other metadata.
Storing user data: MongoDB can also be used to store user data, such as usernames, passwords, and user preferences. This can be useful for authenticating users and customizing the chat experience for each user.

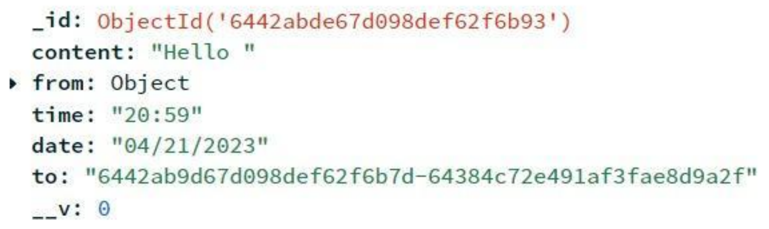
Indexing and querying: MongoDB supports indexing and querying of data, which can make it faster to retrieve chat messages and user data.

Replication and sharding: MongoDB also supports replication and sharding, which can improve the scalability and availability of a chat engine. Replication allows multiple instances of MongoDB to store the same data, while sharding allows the data to be partitioned across multiple MongoDB instances.



```
  _id: ObjectId('643cfc0cdebb54452a359255')
  name: "Sumit Kumar"
  email: "sumitkumar79230@gmail.com"
  password: "$2b$10$JLfX8fQza1DETYkZYHKMcuChEMicpaT69skFi0aICKTEoSxR7MnJm"
  picture: "http://res.cloudinary.com/dtahlza4h/image/upload/v1681718282/tagwnsahb…"
  status: "online"
▸ newMessages: Object
  __v: 0
```

Fig. 1: User data being stored in MongoDB



```
  _id: ObjectId('6442abde67d098def62f6b93')
  content: "Hello "
▸ from: Object
  time: "20:59"
  date: "04/21/2023"
  to: "6442ab9d67d098def62f6b7d-64384c72e491af3fae8d9a2f"
  __v: 0
```

Fig. 2: Message content being stored in MongoDB in JSON format

2.3) Express and Node.js

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside the browser. The most important advantage of using Node is that we can use JavaScript as both a front-end and back-end language.

Express.js, or simply Express, is a web application framework for Node.js. Express provides a robust set of features for web and mobile applications. Express provides a thin layer of fundamental web application features, without obscuring Node.js features.

2.4) React.JS

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 one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding

Chapter 3

Problem Statement / Requirement Specifications

The problem statement for a chat engine can change based on the project's particular aims and goals. However, in general, the goal of a chat engine is to develop a real-time conversational platform that lets users interact with one another. The chat engine must to be built to support numerous users concurrently and deliver a smooth, user-friendly experience.

In order to improve the operation of the chat engine and make it more intelligent and sensitive to user requests, the project may also involve integrating artificial intelligence and natural language processing technology.

3.1 Project Planning

1) Define project scope and requirements: Define the scope of your chat engine project by identifying the features and functionalities you want to include. This step will help you understand the project's goals, budget, and timeline.

2) Choose a suitable technology stack: Choose a technology stack that fits your project requirements. Consider factors like scalability, security, and cost when choosing the right technology.

3) Design the chat engine architecture: Define the architecture of your chat engine, including how different components interact with each other. Consider using a microservices-based architecture for better scalability.

4) Develop the chat engine: Implement the features and functionalities you have identified, including text-based messaging, voice and video calls, file sharing, and group chats.

5) Integrate AI and NLP: Integrate artificial intelligence and natural language processing technologies to enhance the chat engine's functionality and make it more intelligent and responsive to user needs.

6) Perform testing and quality assurance: Test the chat engine to ensure that it meets the functional and non-functional requirements, and that it is scalable, secure, and reliable.

7) Deploy the chat engine: Deploy the chat engine on a production server and make it available to users.

8) Provide ongoing maintenance and support: Provide ongoing maintenance and support to ensure that the chat engine is running smoothly and any issues are resolved quickly.

9) Continuously update and improve: Continuously update and improve the chat engine by incorporating user feedback, adding new features, and optimizing its performance
.

3.2 Project Analysis

The project also needs to be analyzed for potential risks and challenges. This includes identifying any security vulnerabilities, scalability issues, or performance bottlenecks that may arise during the development process. Mitigation strategies need to be developed to address these risks and ensure the project is delivered on time and within budget.

Finally, the project needs to be analyzed from the user's perspective. This includes testing the usability and user experience of the chat engine, ensuring that it is intuitive and easy to use. Feedback from beta testers and user testing should be incorporated to refine the design and functionality of the chat engine.

3.3 System Design

3.3.1 Design Constraints

1) Platform compatibility refers to how well the chat engine functions on different hardware, including desktop, laptop, and mobile platforms. The chat engine's hardware and software must work with a variety of web browsers and operating systems.

2) Infrastructure for the network: The chat engine needs to be tuned to function well under a variety of network situations. The system needs to be built to handle situations with high latency and low bandwidth.

3) Security: It is important to keep security in mind when developing the chat engine. This includes encrypting data while it is in transit and putting authentication and access control measures in place to stop unauthorised access.

4) Scalability: The chat engine needs to be able to deal with a lot of users and messages. As demand changes, the system should be able to scale up and down without compromising performance.

5) Usability: The chat engine should be created to be simple to use and user-friendly. This entails offering a user interface that is simple to use as well as tools like message threading, search, and notifications.

3.3.2 System Architecture or Block Diagram

A chat engine typically involves several components that work together to enable real-time communication between multiple clients. Here's a brief overview of the block diagram for a typical chat engine project:

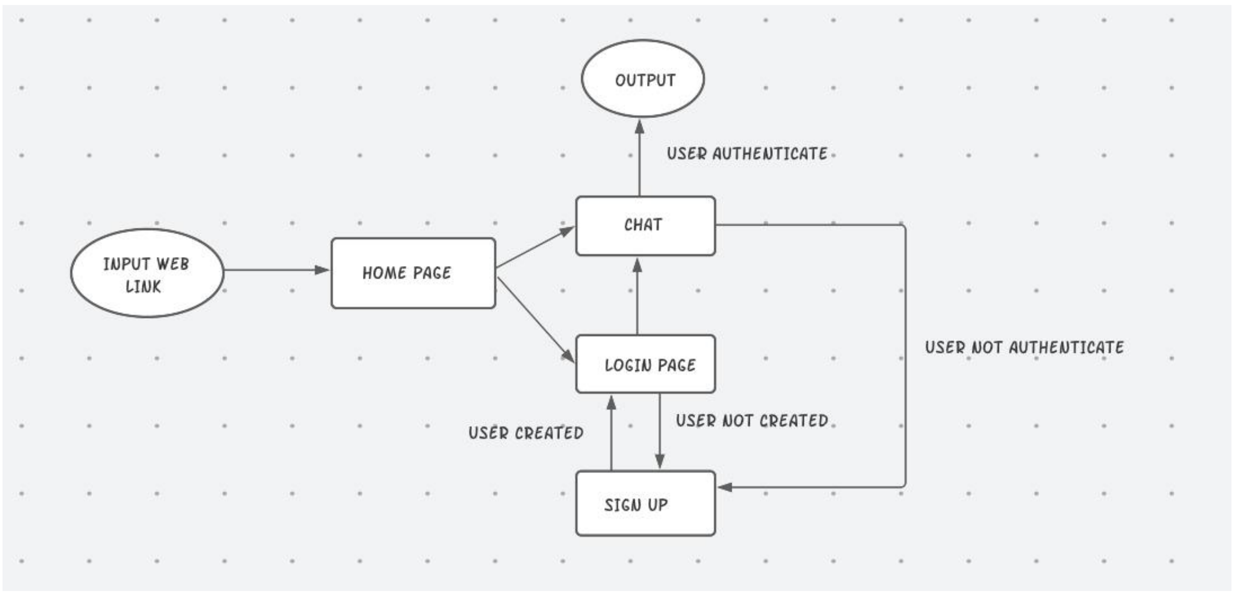


FIG 3.3.2.1 Block Diagram of Chat Engine

Chapter 4 Implementation

Okay, let's explore how you can use this app.

STEP 1:

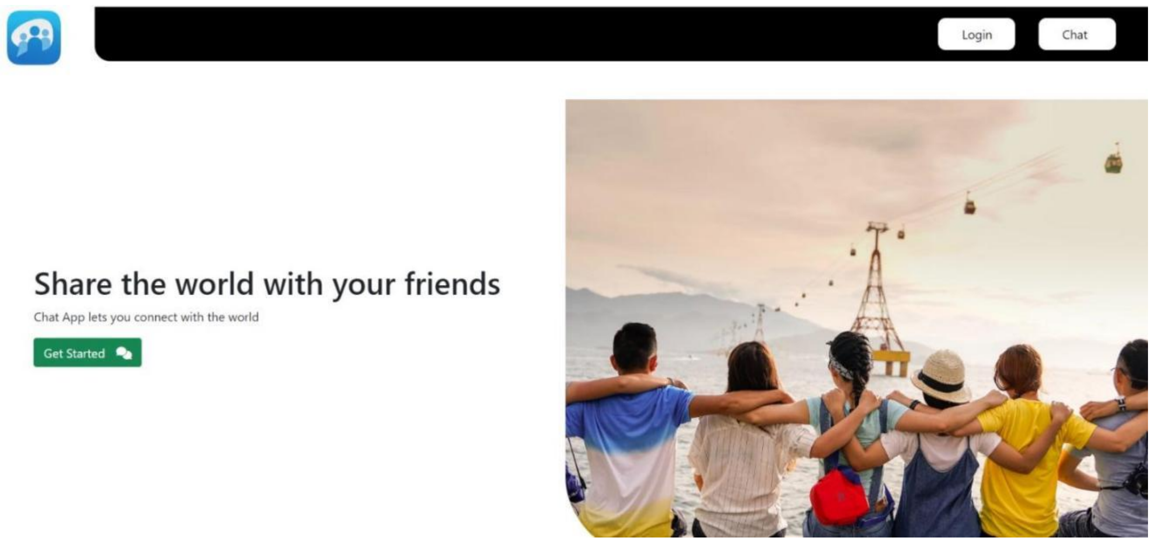

Login
Chat
Share the world with your friends
Chat App lets you connect with the world
Get Started

FIG 4.1 Homepage

This is our homepage where you can see two options in the top right corner. The first option is "Login" and the second option is "Chat". If you click on the "Chat" option without creating an account on the app it will redirect you to the Sign-up page.

STEP 2:

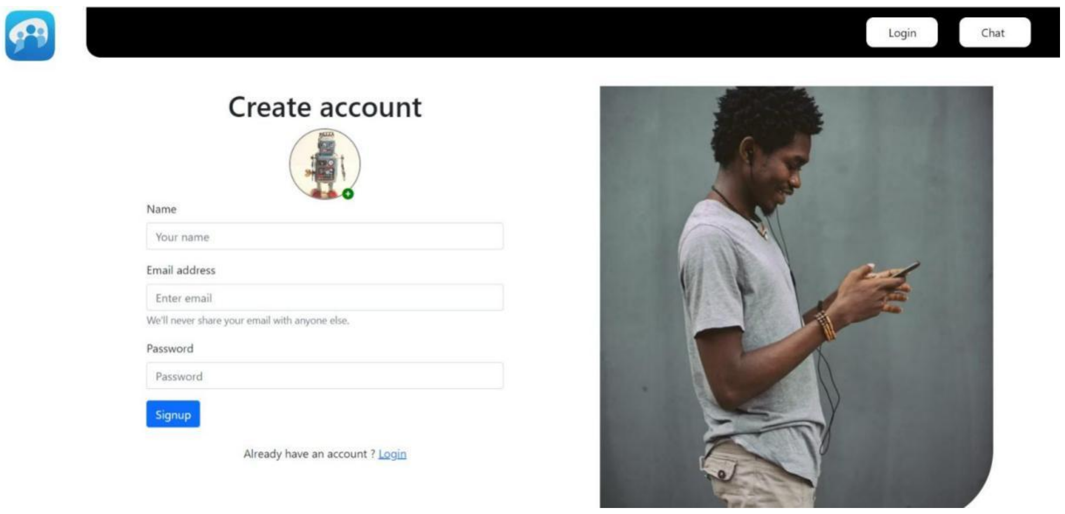
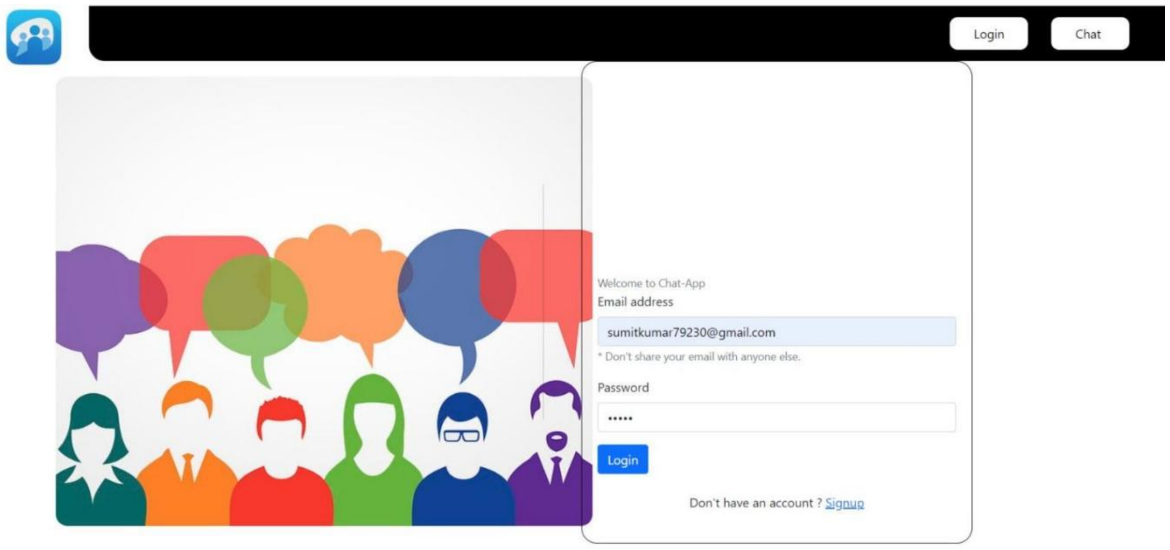


FIG 4.2 SignUp Page

Now, you are on the create account page where you have to provide us with your basic information like name, email, and password. This information will eventually be saved in our database. We also require you to upload a profile picture.

STEP 3:



Welcome to Chat-App

Email address

sumitkumar79230@gmail.com

* Don't share your email with anyone else.

Password

•••••

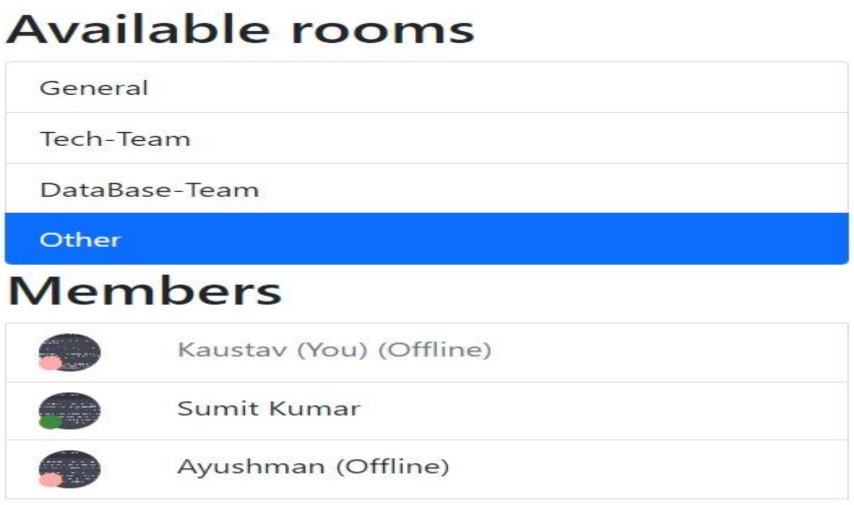
Login

Don't have an account ? Signup

FIG 4.3 Login Page

Once you have created your account, you can now login to our website using your email address and password.

STEP 4:



Available rooms

General

Tech-Team

DataBase-Team

Other

Members

Kaustav (You) (Offline)

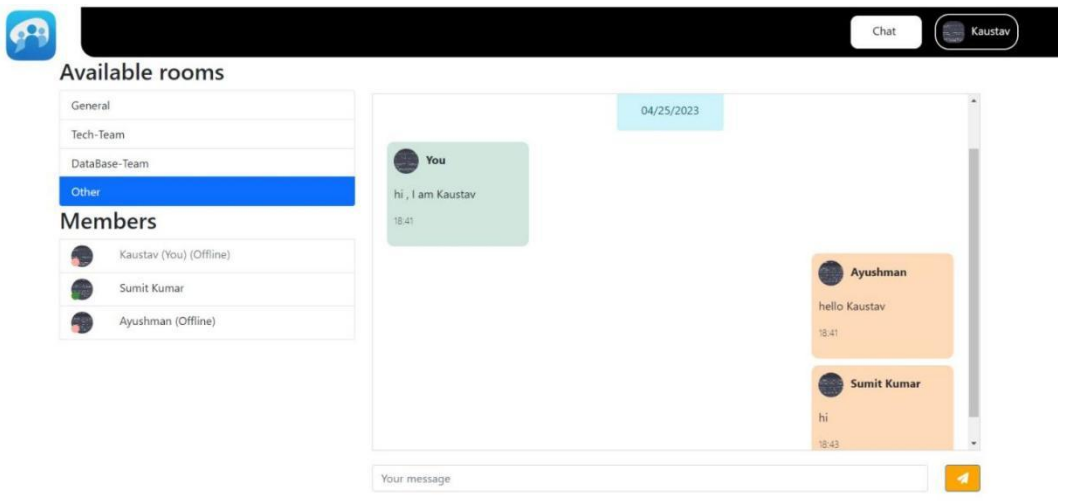
Sumit Kumar

Ayushman (Offline)

FIG 4.4 User Name & Available Rooms

On the left side, you can see the available chat rooms where you can join and start talking to other users. You can also see who is currently online by looking at the list of members at the bottom of the page. If you want to have a private conversation, you can click on the person's name and start chatting with them personally.

STEP 5:



Kaustav (You) (Offline)
Sumit Kumar
Ayushman (Offline)

04/25/2023

You
hi , I am Kaustav
18:41

Ayushman
hello Kaustav
18:41

Sumit Kumar
hi
18:43

FIG 4.5 Chatting Area

This is the main chat page where all public conversations take place. You can see that I sent the message "hi" at 18:41 and ayushmaan and Sumit replied back. One of the features we have added is similar to Telegram's chat feature where new users can see previous messages even if they join the chat late. We think this is a great idea because new users can catch up on important messages they may have missed when they were not present in the group chat.

4.1 Methodology OR Proposal

1) Front-end Development: The chat engine online application's user interface was created using HTML, CSS, and JavaScript. In order to make the web application responsive on various devices, we also used Bootstrap.

2) Back-end Development: The back-end of the chat engine was created using Node.js and Express.js. The back-end server controls client communication and maintains a database of chat data.

3) Database Design: The chat data was stored in MongoDB. User profiles and chat messages will be stored in the database schema that we created.

4) Real-time Communication: To enable client-to-client real-time communication, we used Socket.IO. This eliminated the need to refresh the website in order for users to send and receive messages immediately.

5)Message History: We made use of the message history function to make it possible for new users to view the past chat room messages.

6) Authentication and Authorization: For user authentication and authorization, we made use of JSON Web Tokens (JWT). This made sure that only authorised users could enter and send messages to the chat rooms.

4.2 Testing OR Verification Plan

Any project must go through testing or verification to make sure it adheres to the requirements and performs as expected. Testing is frequently used in software development to examine the software's usability, performance, security, and user interface.

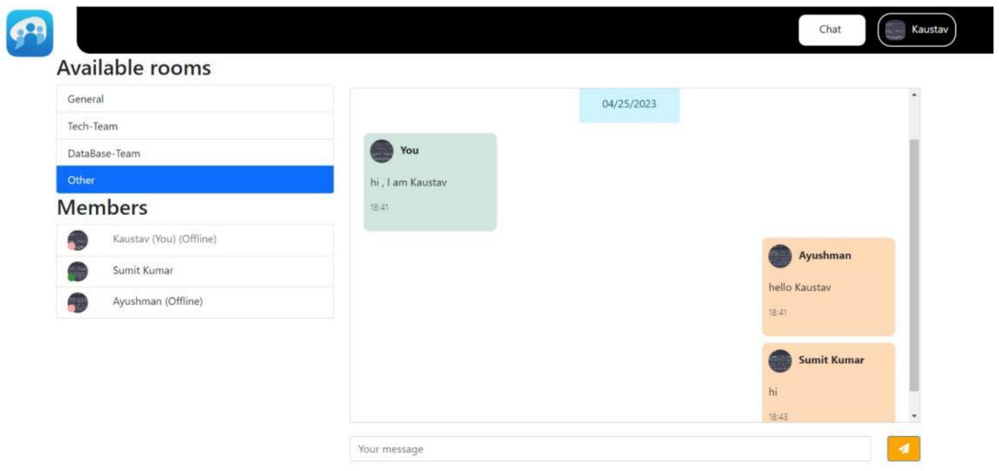
Some test cases that could be included for verification may include:

1) Functionality : Okay, the first thing we will do is check the functionality of your project, which includes sending and receiving messages, creating and deleting chat rooms, inviting users, and blocking users.

2) Performance: Performance of the chat engine under different conditions, such as the number of users, message traffic, and network bandwidth. This will help ensure that the chat engine can handle high volumes of users and messages without crashing.

3) User interface: Test the user interface of the chat engine, ensuring that it is user-friendly, intuitive, and accessible to users with disabilities.

4.3 Result Analysis OR Screenshots



04/25/2023

You
hi , I am Kaustav
18:41

Ayushman
hello Kaustav
18:41

Sumit Kumar
hi
18:43

FIG 4.6 Result Analysis

4.4 Quality Assurance

We recently conducted a Quality Assurance (QA) process as part of our study on developing a chat engine using natural language processing and machine learning models to improve conversational AI. Our designated professor at KIIT examined our project and provided feedback on our strategy, process, and outcomes.

During the QA process for our chat engine project, our professor at KIIT conducted a comprehensive review of our project documentation. This included our proposal, methodology, testing and verification strategy, and findings. In addition, our professor examined our code and tested the functionality of our project, including the bi-directional chatting

Overall, the QA procedure was a beneficial experience for our research because it gave us the chance to get criticism from an accomplished academic and raise the calibre of our work. We are appreciative of the help and direction our KIIT professor has given us.

Chapter 5

Standards Adopted

5.1 Design Standards

1) Define project requirements: Clearly define the functional and non-functional requirements of the chat engine project, such as its features, performance, scalability, and security.

2) Use a design methodology: Here we have use a design methodology that is appropriate for the chat engine project, such as Agile or Waterfall, and follow it consistently throughout the project.

3) Create a project plan: Plan that outlines the schedule, budget, and resources required for the chat engine project.

4) Document design decisions: Document all design decisions and assumptions, including trade-offs and rationale.

5) Use database design standards: Follow database design standards, such as normalization and data modeling, to ensure data integrity and efficiency.

6) Follow coding standards: Follow coding standards, such as naming conventions, formatting, and commenting, to improve readability and maintainability of code.

7) Use design patterns: Use design patterns to address recurring design problems and improve code reusability and maintainability.

8) Test and validate design: Test and validate the design through unit testing, integration testing, and user acceptance testing to ensure that it meets the specified requirements and quality standards.

5.2 Coding Standards

Coding standards are collections of coding rules, guidelines, and best practices. Few of the coding standards are:

```
function Signup() {

    const [email, setEmail] = useState("");

    const [password, setPassword] = useState("");

    const [name, setName] = useState("");

    const [signupUser, { isLoading, error }] = useSignupUserMutation();

    const navigate = useNavigate();

    //image upload states

    const [image, setImage] = useState(null);

    const [upladingImg, setUploadingImg] = useState(false);

    const [imagePreview, setImagePreview] = useState(null);
```

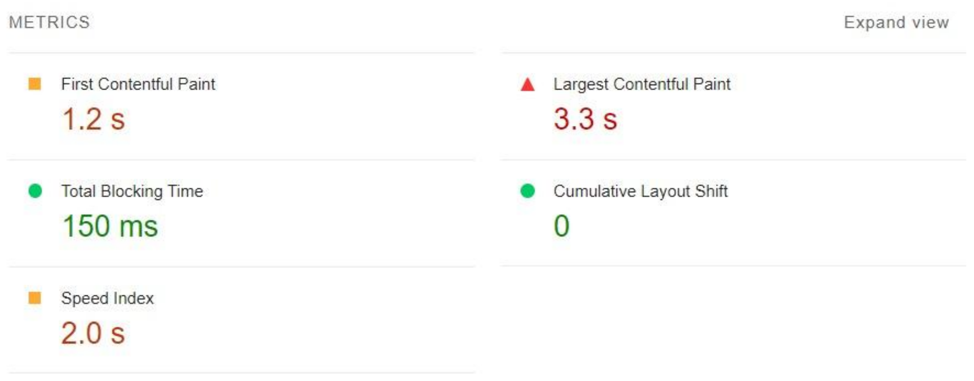
Okay so here we defines a Signup component. The component uses the useState hook to define state variables for email, password, and name, which are initialized to empty strings.The component also uses the useSignupUserMutation hook from some GraphQL client library to perform a sign-up mutation. The signupUser variable contains the function to call the mutation. The isLoading variable indicates whether the mutation is currently being processed. The error variable holds any errors that occur during the mutation.The component uses the useNavigate hook to get access to the navigate function from the React Router library. This function can be used to navigate to a different page in the application.Additionally, the component defines state variables for image, upladingImg, and imagePreview. These variables are used for uploading an image during the sign-up process.Overall, this component is used for signing up a new user in the application, with support for uploading an image as part of the sign-up process.

```
async function handleSignup(e) {

        e.preventDefault();

        if (!image) return alert("Please upload your profile picture");

        const url = await uploadImage(image);

        console.log(url);
        // signup the user
        signupUser({ name, email, password, picture: url }).then(({ data }) => {
          if (data) {
            console.log(data);
            navigate("/chat");
          }
        });
      }
```

Function called handleSignup that handles the sign-up process for a user. The function is likely called when a user submits a form to sign up for the application.The function first calls preventDefault() on the event object passed to the function to prevent the default form Submission behavior.Next, the function checks if an image has been uploaded by the user. If not, the function displays an alert message asking the user to upload their profile picture.If an image has been uploaded, the function calls the uploadImage function to upload the image to Cloudinary and retrieve the URL of the uploaded image. The URL is then logged to the console for debugging purposes.The function then calls the signupUser function, which was defined earlier and likely sends a GraphQL mutation to sign up the user with their name, email, password, and profile picture URL. The function uses the .then() method to handle the response data returned from the mutation. If the data contains any values, the function logs the data to the console and uses the navigate function from the React Router library to navigate the user to the chat page.

5.3 Testing Standards

There are some ISO and IEEE standards for quality assurance and testing of the product.



METRICS

Expand view

■ First Contentful Paint
1.2 s

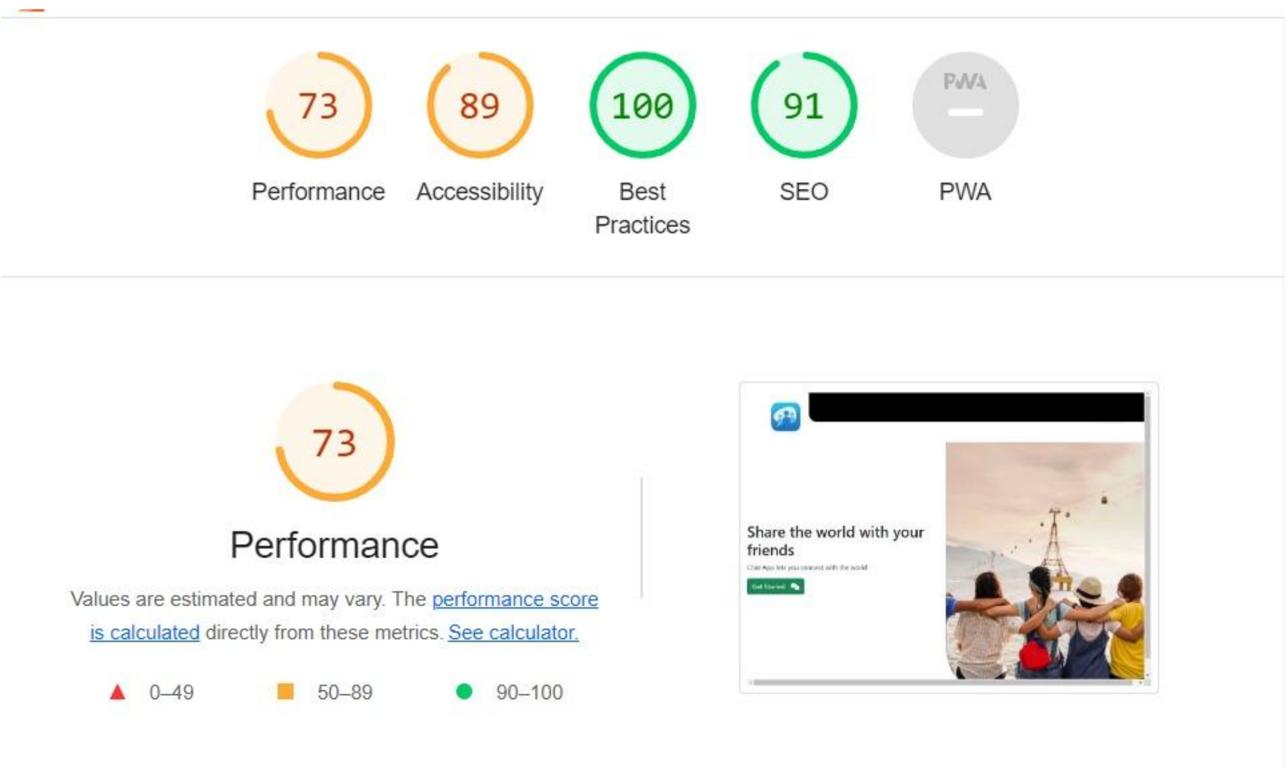
▲ Largest Contentful Paint
3.3 s

● Total Blocking Time
150 ms

● Cumulative Layout Shift
0

■ Speed Index
2.0 s

FIG 5.3.1 TESTING METRICS



73 Performance | 89 Accessibility | 100 Best Practices | 91 SEO | PWA PWA

73

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

[redacted]

FIG 5.3.2 PERFORMANCE TESTING

Conclusion and Future Scope

6.1 Conclusion

In conclusion, this chat engine project has successfully developed a real-time chat application that allows users to communicate with each other in real-time using text messages. The project has implemented various features such as user authentication, online status, notifications, and user profile management.

During the project, we faced several challenges such as scalability, user experience, and security, but we managed to overcome them by adopting appropriate technologies and design patterns.

The project has some limitations such as limited support for video and audio calls, lack of end-to-end encryption, and limited accessibility features for users with disabilities. These limitations can be addressed in future work by adopting appropriate technologies and design patterns.

Overall, this chat engine project has successfully achieved its goals and objectives by providing an efficient and user-friendly chat application. The project can be further improved by incorporating additional features and enhancements to meet the evolving needs of users.

6.2 Future Scope

The future scope of the project could include adding features such as video and audio calls, end-to-end encryption, multi-language support, artificial intelligence and chatbots, and other potential improvements to enhance the user experience and security.

References:-

[1] https://socket.io/docs/v4/server-api/
[2] https://socket.io/docs/v4/client-api/
[3] https://expressjs.com/en/4x/api.html
[4] https://developer.mozilla.org/en-US/docs/Glossary/CORS
[5] https://www.mongodb.com/docs/
[6] https://nodejs.org/en/docs/
[7] https://reactjs.org/docs/getting-started.html
[8] https://www.freecodecamp.org/news/building-a-realtime-chat-app-with-node-js-socket-io-and-react/
[9] https://pusher.com/tutorials/react-chat-app

INDIVIDUAL CONTRIBUTION REPORT:

CHAT-ENGINE

SAURABH ANIKET

20051702

Abstract: The aim of a Chat Engine project is to provide an effective and dependable chat programme that enables users to connect with one another in real time. The goal is to develop a platform that offers a seamless user experience with no latency and downtime, as well as being safe and easy to use.

Individual contribution and findings: My role in this project was as the group leader. I prepared the proposal PowerPoint presentation which was submitted to the project supervisor. I worked on the back-end side of the project, which included ensuring that the website's security features worked well. Identification of potential security threats, implementation of security features like encryption, firewalls, and access controls, testing the features to ensure their efficacy, and ongoing monitoring and updating of the security features were all part of the planning involved in my portion of the project.While working on this project, I gained a lot of new knowledge and skills, such as using databases to store data and implementing bi-directional chatting. I also learned about security measures such as encryption and authentication that are necessary to protect user privacy and prevent unauthorized access.

Individual contribution to project report preparation: My role in preparing the project report was to write the content for Chapter 2 (Basic Concepts/Literature Review) and Chapter 6 (Conclusion and Future Scope) parts.

Individual contribution for project presentation and demonstration: My role in the project included identifying potential security threats, implementing security features like encryption, firewalls, and access controls, testing the features to ensure their efficacy, and ongoing monitoring and updating of the security features. To make this work, I followed a few stepsI started by establishing the MongoDB database. I then developed the login and registration API endpoints. I created middleware to secure user authentication and authorisation by verifying the token's validity before granting access to particular API endpoints. This made sure that only users who had been authenticated could access certain resources or do certain actions. I encrypted the user credentials using a hashing algorithm to make sure they were saved safely.

Full Signature of Supervisor: Full signature of the student:

INDIVIDUAL CONTRIBUTION REPORT:

CHAT-ENGINE

SUMIT KUMAR

20051586

Abstract: The aim of a Chat Engine project is to provide an effective and dependable chat programme that enables users to connect with one another in real time. The goal is to develop a platform that offers a seamless user experience with no latency and downtime, as well as being safe and easy to use.

Individual contribution and findings: My role in the project was to work on the back-end part, which included ensuring that the database was connected and able to store the messages sent in the group. This would allow new users to see the messages sent before they joined the group. Additionally. I also worked on implementing bi-directional chatting on the website. While working on this project, I gained valuable experience in using Socket.IO, a popular JavaScript library for real-time communication between clients and servers. I also learned how to use the React framework, which was extremely helpful in designing the user interface. In addition, I gained knowledge about DynamoDB, a NoSQL database that is well-suited for scalable and high-

Individual contribution to project report preparation: My role in preparing the project report was to write the content for Chapter 3 (Problem Statement/Requirement Specifications)

Individual contribution for project presentation and demonstration: I will outline my role in a project where I worked on the back-end part. My primary responsibility was to ensure that the database was connected and able to store messages sent in a group. This would allow new users to see the messages sent before they joined the group.In conclusion, implementing bi-directional chatting on a website using Socket.IO required setting up Socket.IO, creating API endpoints for chatting, creating a front-end UI, and testing and debugging the feature. This process required careful planning and attention to detail to ensure that the feature was robust and secure. With these steps, we were able to create a chat feature that met our project's requirements and enabled users to communicate with one another in real-time. Through this project, I gained valuable experience in using Socket.IO and learned the importance of real-time communication in creating engaging web applications.

Full Signature of Supervisor: Full signature of the student:

INDIVIDUAL CONTRIBUTION REPORT:

CHAT-ENGINE

AYUSHMAAN RATHOD

20051693

Abstract: The aim of a Chat Engine project is to provide an effective and dependable chat programme that enables users to connect with one another in real time. The goal is to develop a platform that offers a seamless user experience with no latency and downtime, as well as being safe and easy to use.

Individual contribution and findings: My role in the project was to work on the front-end part. which included the design of the very user friendly interactive UI. I used React in this project to design a good front end for the project. Throughout the project, I was able to learn and apply various React concepts such as component composition, state management, and life cycle methods. These skills proved valuable not only in this project but also in future projects that I have worked on.

Individual contribution to project report preparation: My role in preparing the project report was to write the content for Chapter 5 (Standard Adopted)

Individual contribution for project presentation and demonstration: In this report, I will outline my role in a project where I worked on the front-end part. My primary responsibility was to design a user-friendly and interactive UI for the web application. I used React in this project to create a responsive and visually appealing front-end design.

Full Signature of Supervisor: Full signature of the student:

INDIVIDUAL CONTRIBUTION REPORT:

CHAT-ENGINE

ADESH PRATAP SINGH

20051707

Abstract: The aim of a Chat Engine project is to provide an effective and dependable chat programme that enables users to connect with one another in real time. The goal is to develop a platform that offers a seamless user experience with no latency and downtime, as well as being safe and easy to use.

Individual contribution and findings: My role in the project was to work on the front-end part. which included the design of the very user friendly interactive UI. I React in this project to design a very attractive user-friendly front end. I also worked on the documentation of the project. While working on this project, I gained valuable experience and in depth knowledge in HTML CSS, Java script and React. I learn how to coordinate and work as a team by maximizing the potential of each member of the group.
My responsibilities were to design the UI, create the essential components, and ensure that they worked properly. I also had to ensure that the front-end and back-end components of the project were properly connected, providing a seamless user experience.

Individual contribution to project report preparation: My role in preparing the project report was to write the content for Chapter 4 (Implementation)

Individual contribution for project presentation and demonstration: To make the project successful, I followed several steps in designing and developing the user interface (UI). The first step was to design the UI, where I created wireframes, mockups, and prototypes based on user personas and feedback from the project team. The second step was to develop the UI using React, utilizing concepts such as component composition, state management, and life cycle methods to create a responsive and visually appealing UI. The third step was to integrate the UI with the back-end, working closely with the back-end team to ensure seamless integration and utilization of APIs to fetch and display data on the UI. Finally, after completing the UI development and integration, I conducted extensive testing and cross-browser testing to ensure the UI was functioning correctly and meeting project requirements before deploying it to the production environment.

Full Signature of Supervisor: Full signature of the stude