Minor Project Synopsis

on

REAL TIME CHAT APPLICATION

In partial fulfillment of requirements for the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION CYBER SECURITY

Submitted by:

GOURANG ANIYA [21100BTCSICS09528]

PARIDHI AKOTIYA [21100BTCSICS09546]

VAISHNAVI MAURYA [21100BTCSICS09556]

Under the guidance of

PROF. RAHUL CHAUDARY

DEPARTMENT OF INFORMATION TECHNOLOGY

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY

SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA, INDORE

JAN - JUNE 2024

Contents

ABSTRACT

Real Time Chat Application is a comprehensive real-time chat application that is developed using the MERN (MongoDB, Express.js, React.js, Node.js) stack. This application provides users with a seamless and efficient platform for instant messaging and communication.
At the core of the Real Time Chat Application is MongoDB, a powerful Node Sql database that is used for storing user information and chat history. MongoDB's flexible document based structure allows for easy scalability and efficient data management.
The backend of the application is built using Express.js, a fast and minimalist web application framework for Node.js. Express.js handles HTTP requests and provides a robust API for communication between the frontend and backend. It enables efficient routing, middleware integration, and error handling, ensuring smooth data flow and reliable performance.
On the frontend, the Real Time Chat Application utilizes React.js, a popular JavaScript library for building user interfaces. React.js offers a component-based architecture that allows for reusable and modular UI elements. This results in a responsive and interactive user interface, providing a smooth and engaging chatting experience.
Node.js serves as the runtime environment for the Real Time Chat Application. It enables server-side JavaScript execution, facilitating efficient handling of real-time operations and ensuring optimal performance.
In summary, the Real Time Chat Application leverages the power of MongoDB, Express.js, React.js, and Node.js to create a comprehensive real-time chat application. It offers a seamless and efficient platform for instant messaging and communication, providing users with a responsive and interactive chatting experience
Chat app website we visit use HTTP to make API calls, which means the client sends a request to the server, and the server sends back a response. This kind of communication can only be initiated by the client. It suits most needs for a website, e.g. to get data from a server once.

INTRODUCTION

The old manual system was suffering from a series of drawbacks. Since the whole of the system was to be maintained with hands the process of keeping, maintaining and retrieving the information was very tedious and lengthy. The records were never used to be in a systematic order. there used to be lots of difficulties in associating any particular transaction with a particular context. If any information was to be found it was required to go through the different registers, documents there would never exist anything like report generation. There would always be unnecessary consumption of time while entering records and retrieving records. One more problem was that it was very difficult to find errors while entering the records. Once the records were entered it was very difficult to update these records. The reason behind it is that there is lot of information to be maintained and have to be kept in mind while running the business .For this reason we have provided features Present system is partially automated (computerized), actually existing system is quite laborious as one has to enter same information at three different places.

The web app will consist of 2 parts, the client and the server. The client will contain 2 main components: a login form, where the user can write a username to use for the chat and the option to upload an image to use as an avatar throughout the conversation. The second component will consist of the chat messages, where each message that is sent, is viewed from anyone that's connected to the chat room. The server will accept two type of connections: an HTTP request, toaccept an image file, upload to a Cloudinary media library through their API and return the image's link. By creating an account, you can get about 100MB of storage. The second connection the server manages is a WebSocket port to manage incoming connections and broadcast messages to all connected clients, in real time. As we expect a large volume of data to be written to the database in a short amount of time and the data that we save are unstructured (using an avatar is not mandatory), it makes sense to use a Nodesql type of database. For this application, we will be using MongoDB

LITERATURE REVIEW

1. Timeline of the reported problem

Email , chat applications provide one to one communication i.e only two people can talk to each other. But what if we want to send something to a group .So our application solves that only.
The second problem is authentication. Many people create fake ids and can spam or sell products to you. This is the second problem that we have solved.

2. Proposed solutions

In this application we will be able to make a group and send something globally. After that there is a proper login system that marks your identity and at what time you are sending message. Authentication enables organizations to keep their networks secure by permitting only authenticated users or processes to gain access to their protected resources.
This may include computer systems, networks, databases, websites and other network based applications or services.

3. Bibliometric analysis

Chat rooms have become very popular after the covid era. Whether it is team meetings or you want to share something everyone uses chat apps. Authentication is used by a server when the server needs to know exactly who is accessing their information or site. Authentication is used by a client when the client needs to know that the server is system it claims to be.

4. Review Summary

A chat application makes it easy to communicate with people anywhere in the world by sending and receiving messages in real time. With a real-time chat app, users are able to receive the same engaging and lively interactions through custom messaging features, just as they would in person.

5. Problem Definition

We will be able to make a group and send something to a group. After that there is a proper login system that marks your identity and at what time you are sending message. This is doneusing nodesql database mongodb.

Problem Statement

The "Online Chat Application" has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. Moreover this system is designed for the particular need of the company to carry out operations in a smooth and effective manner. The application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data .No formal knowledge is needed for the user to use this system. Thus by this all it proves it is user-friendly. Online Chat Application, as described above, can lead to error free, secure, reliable and fast management systems. It can assist the user to concentrate on their other activities rather than concentrating on the record keeping. Thus it will help organizations in better utilization of resources. Every organization, whether big or small, has challenges to overcome and manages the information of Chat User, Chat Profile, Group Chat, Multi Chat, Smileys Chat. Every Online Chat Application has different Chat Profile needs, therefore we design exclusive employee management systems that are adapted to your managerial requirements. This is designed to assist in strategic planning, and will help you ensure that your organization is equipped with the right level of information and details for your future goals. Also, for those busy executives who are always on the go, our systems come with remote access features, which will allow you to manage your workforce anytime, at all times. These systems will ultimately allow you to better manage resources.

Objectives

Statements setting the milestones during the course of project work.
Keeping in mind
1.To make a encrypted chat app using mern stack.
2.To learn react js because it is one of the most trending technologies.3.To learn socket.io.
4.To know the fundamentals of nodesql database.
5.To work together efficiently in groups.
6. To create a safe and secure environment for users .
7. Cross platform is efficient.
8.Multimedia sharing .
9.Can be customized and is scalable.

IMPLEMENTATION OF METHODOLOGY

1. Evaluation & Selection of Specifications/Features

The web app will consist of 2 parts, the client and the server. The client will contain 2 main components: a login form, where the user can write a use name to use for the chat and the option to upload an image to use as an avatar throughout the conversation. The second component will consist of the chat messages, where each message that is sent, is viewed from anyone that's connected to the chat room.

2. Design Constraints

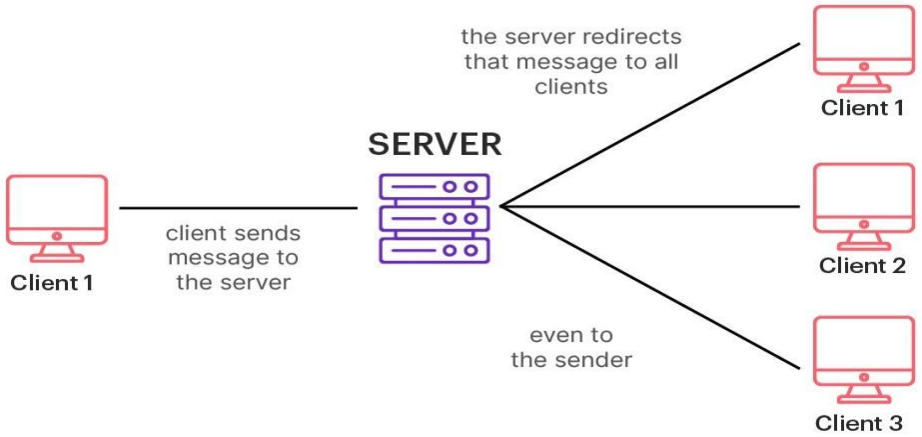
The server will accept two type of connections: an HTTP request, to accept an image file, upload to a Cloudinary media library through their API and return the image's link. By creating an account, you can get about 100MB of storage. The second connection the server manages is a WebSocket port to manage incoming connections and broadcast messages to all connected clients, in real time. As we expect a large volume of data to be written to the database in a short amount of time and the data that we save are unstructured (using an avatar is not mandatory), it makes sense to use a Nodesql type of database. For this application, we will be using MongoDB.

3. Analysis and Feature finalization subject to constraints

In this application we will be able to make a group and send something globally. After that there is a proper login system that marks your identity and at what time you are sending a message. Authentication enables organizations to keep their networks secure by permitting only authenticated users or processes to gain access to their protected resources. This may include computer systems, networks, databases, websites and other network-based applications or services. We will be able to make a group and send something to a group. After that there is a

proper login system that marks your identity and at what time you are sending a message. This isdone using nodesql database mongodb.

4. Design Flow And Implementation plan/methodology.



This is how the app works. One user can send messages to multiple users at a time. Everytime a user sends a message it gets registered in the database and then pops up on the screen of all the members of the group at once. There is also a notification tab which shows notification when a user's message comes.

CHAT CLIENT

The chat client is what the user experiences. A desktop, web or smartphone chat application, the chat client is responsible for interacting with the operating system. Interactions include sending push notifications, displaying data to the user and storing messages and files. When you type a message and hit send, the chat client transmits that message to the server.

CHAT SERVER

The chat server is just that, a that hosts all the software, frameworks and databases necessary for the chat app to operate. This server is responsible for securely receiving a message, identifying the receiver and then forwarding the message to the client.

CHAT rest API

A Chat REST API is used to facilitate the functionality of the chat app outside of messaging. For example, authentication, profile settings and notification settings can all be managed through a REST API.

TECHNOLOGY REQUIREMENTS TO BE USED .

HARDWARE PLATFORMS:

1. Processor (CPU):

- Dual-core processor or higher.

2. RAM:

- 4 GB or more.

3. Storage:

- SSD or HDD with sufficient space for the application and data storage**.**

4. Network Interface:

- Wired or wireless network interface for internet connectivity**.**

5. Graphics:

- Integrated graphics are generally sufficient for a chat application. Dedicated graphics are not typically necessary unless the application involves multimedia processing.

SOFTWARE PLATFORMS:

The minimum software requirements for a real-time chat application on a PC typically include:

1. Operating System:

- Windows, macOS, or Linux distributions are common choices. Ensure that the chat application is compatible with the targeted operating system**s.**

2. Web Browser:

- Support for popular web browsers such as Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge is essential if the chat application is web-based.

3. Internet Connection:

- A stable and reasonably fast internet connection is necessary for real-time communication and message delivery.

4. Messaging Application:

- If the chat application is a standalone desktop application, users may need to install the application specific to their operating system.

TOOLS.

1. *Text Editor or Integrated Development Environment (IDE):*

- Choose a text editor like Visual Studio Code, Sublime Text, or an IDE like Visual Studio, IntelliJ, or Eclipse for coding.

2. *Version Control System:*

- Use a version control system like Git to track changes in your codebase. Platforms like GitHub, GitLab, or Bitbucket can host your repository.

3. *Web Browser:*

- A modern web browser (e.g., Google Chrome, Mozilla Firefox) for testing and debugging your chat application.

4. *Node.js and npm:*

- Install Node.js to run JavaScript on the server side and npm (Node Package Manager) for managing packages. These are essential for building and running server-side applications.

5. *Backend Framework:*

- Choose a backend framework for your chat application. Popular choices include Express.js (Node.js), Flask (Python), Django (Python), or Spring Boot (Java).

6. *WebSocket Library:*

- If your chosen backend doesn't have built-in WebSocket support, you may need a WebSocket library like Socket.io (for Node.js) or Flask-SocketIO (for Flask)

Real-time chat applications offer several advantages:

1. *Instant Communication:*
Users can exchange messages in real time, facilitating quick and seamless communication.

2. *Enhanced Collaboration:*
Real-time chat fosters collaboration among users, whether for work projects, team discussions, or group activities.

3. *Quick Decision-Making:*
Enables swift decision-making through immediate communication and information exchange.

4. *Increased Productivity:*
Real-time communication reduces delays, leading to faster responses and increased productivity in both personal and professional settings.

5. *User Engagement:*
Provides an engaging and interactive platform, keeping users connected and involved in ongoing conversations.

6. *Flexibility:*
Users can access the chat application from various devices, allowing flexibility and ensuring communication continuity.

7. *Rich Media Sharing:*
Supports the sharing of multimedia content, including images, videos, and files, enhancing the depth of communication.

8. *User Presence Indicators:*
Displays indicators showing who is online, helping users know when others are available for chat.

The future scope for real-time chat applications is promising and may include several trends and advancements:

1. AI-powered Chatbots:

Integration of advanced artificial intelligence (AI) and natural language processing (NLP) to enhance chatbots, making them more intelligent and capable of handling complex interactions.

2. Augmented Reality (AR) and Virtual Reality (VR) Integration:

Incorporating AR and VR technologies to create immersive and interactive chat experiences, especially in areas like virtual meetings or collaborative workspaces.

3. Enhanced Security Measures:

Continued focus on improving security protocols, including advanced encryption techniques, to safeguard user data and privacy in real-time communications.

4. Voice and Video Integration:

Further development of voice and video communication features within chat applications, offering users a seamless transition between text, voice, and video interactions.

5. Decentralized and Blockchain-Based Messaging:

Exploration of decentralized and blockchain-based solutions for Messaging, aiming to enhance privacy, security, and eliminate the need for centralized servers.

6. Multilingual and Cross-Platform Capabilities:

Continued improvement in supporting multiple languages and ensuring seamless cross-platform compatibility to enhance accessibility for a diverse user base.

7. Advanced Multimedia Sharing:

Evolution of multimedia sharing features with better support for high-quality images, videos, 3D content, and interactive media to enrich communication.

8. Real-Time Translation:

Improved real-time language translation features to facilitate communication between users who speak different languages.

EXPECTED OUTCOME

In conclusion, the real-time chat application mini project successfully demonstrated the implementation of a dynamic and interactive communication platform. Through the use of WebSocket technology and a well-designed architecture, the application achieved instant messaging, user presence indicators, and real-time updates. The inclusion of multimedia sharing, group chat, and customization options enhanced the overall user experience.

The project also emphasized key principles such as security, scalability, and cross-platform compatibility. By incorporating end-to-end encryption and considering the scalability of the application, we addressed crucial aspects of user privacy and the ability to handle increased user activity.

Throughout the development process, attention was given to creating a user-friendly interface, optimizing the application for responsiveness, and implementing features like push notifications to keep users informed even when the application is not actively in use.

While this mini project provides a solid foundation for a real-time chat application, continuous improvements and refinements could further enhance its capabilities. This project serves as an educational endeavor, offering insights into the complexities and considerations involved in developing an effective real-time communication system. Future iterations could explore additional features, integrations with external services, and optimizations for even smoother performance. Overall, the real-time chat application mini project successfully demonstrated the practical application of key concepts in real-time communication technology.

REFERENCES

So, in order to complete this project, we have collected many things from the following onlineresources from development to deployment ,from learing to implementing:
https://www.heroku.com/
https://stackoverflow.com/www.youtube.com
https://nodejs.org/en/docs/
https://www.quackit.com/css/color/tools/css_color_picker.cfm
www.w3schools.com
https://reactjs.org/docs/getting-started.html https://www.mongodb.com/developer/products/atlas