REAL TIME CHATTING APPLICATION FOR NATIVE ORGANIZATION

Niranjan V[1], Nithiesh K V[2]

[1] *Student, Computer Technology, Bannari Amman Institute of Technology, Tamil Nadu, India*
[2] *Student, Computer Science and Engineering, Bannari Amman Institute of Technology, Tamil Nadu, India*

ABSTRACT

The chat application is an essential tool in today's society since it makes it possible for everyone to easily have real-time chats with one another from any location and from any device. Because you can now send a message to the other user and get the assistance you require without having to make a call, this technology also addresses the issue of having to call someone every time you have a question about your job. The chat program's biggest feature is its seamless ability to deliver the same message to numerous users at once. To achieve this, we can either construct a large group of users or broadcast the message to a selected set of users. The challenge is how to create and implement real-time responsive web-based, mobile, or other device chat applications, according to the problem statement we have specified. There are two sides to the application: a client side and a server side. We wish to confirm that the message sent by this user and the time it was sent are visible to everyone who has signed into the chat room. A message sent by the client gets to the server since the client and server are connected in a bidirectional fashion. But, the server can only print that message on the terminal or perform other actions, such as dumping it into a file, when it receives that message. There will be more clients added, and this time, the clients will get data from the server. The server will inform all other clients that client 1 has successfully delivered a message. Clients will get this signal, and they can then use a browser to render the message. In essence, the server receives every message a client provides and distributes it to each client currently present in the room.

Keyword: *Chat Application, Node.Js, Socket.Io, Multiple-Platform, Multiple-User, Real-Time Chat.*

1. INTRODUCTION

Our group chat software for real-time communication which was developed using cutting-edge tools like ReactJS, Firebase, SocketIO, and Material UI. Our application is intended to promote interpersonal interaction and enhance effective group communication. Our platform provides a fantastic way to stay in touch with your colleagues, friends, and family in light of the rising need for remote work and social isolation. Our group chat programme is a great option for anyone seeking a cutting-edge communication tool because it is quick, dependable, and simple to use. Real-time texting is one of our platform's main features. This entails that communications can be sent and received immediately, without any lag time. Also, you can see when other users are typing, which helps you gauge how the chat is progressing. SocketIO, a well-liked library for creating real-time applications, powers this feature. Our platform offers several chat rooms that you may build and join in addition to real-time messaging. This allows you to maintain orderly and manageable talks with several groups of individuals by having distinct conversations with each group. Each chat room's name and color can be changed, making it simple to tell which conversation you are in at a look. The use of Firebase is just another intriguing aspect of our platform. We can store and retrieve data in real-time using Firebase, a potent backend service. This means that regardless of Whether you're using a desktop computer or a mobile phone, you can view your chat history from any device. Your interactions are kept private and

secure thanks to Firebase robust security measures Also, our platform supports Material UI, a well-liked front-end library for creating contemporary user interfaces. This indicates that our application's user interface is clear and simple and that it is both useful and aesthetically pleasing. Our application looks fantastic on every device, regardless of screen size, thanks to Material UI's responsive design. As a result, our real-time group chat programme is an effective tool for maintaining communication among group members. Our platform offers a contemporary and user-friendly solution for all your communication needs with real-time messaging, several chat rooms, Firebase connectivity, and Material UI. We sincerely hope you enjoy using our site and eagerly await your comments. Each chat application must prioritize security, especially when it comes to group discussions that might include confidential information. To guarantee that your interactions are kept private and safe, our real-time group chat programme has been created with strong security features. Using secure communication protocols is one of the first steps in assuring the security of our platform. As an illustration, we secure client-server communication using HTTPS. By ensuring that every communication is encrypted, attackers will have a hard time intercepting and deciphering the communications. User authentication is a crucial security safeguard that we have put in place. This indicates that only those who have registered and have valid login information may access the programme. To guarantee the security and protection of user login information, we employ Firebase authentication. We employ authorization in addition to user authentication to make sure that users can only access the chat rooms that they are permitted to join. This is crucial because it guards against unauthorized people listening in on private chats. Also, we take precautions against widespread security risks including cross-site scripting (XSS) and SQL injection attacks. In order to defend against these kinds of attacks, our platform was created with security best practices in mind, including parameterized queries and input validation. 2 Before transmitting any chat messages across the network, we encrypt them completely to protect their confidentiality and integrity. The Advanced Encryption Standard (AES) algorithm is a well-known and reliable encryption standard that we employ. By doing this, it is ensured that even if a hacker intercepts the signals, they cannot be decoded and read. Lastly, to keep our programme safe from the most recent security threats, we routinely update and maintain the software. In order to find and address any possible vulnerabilities that could be present, we also carry out routine security audits. In conclusion, security was taken into consideration when developing our real-time group chat programme. We have put in place a number of security measures, such as encrypted messages, secure communication protocols, user identification and authorisation, and regular upkeep and maintenance. We will keep updating our platform with the newest security features and procedures since we are dedicated to maintaining the privacy and security of your discussions. Second, Firebase Realtime Database offers client-side validation rules that enable us to guarantee that data is structured appropriately and complies with predetermined standards. For instance, we can guarantee that usernames are distinctive and that passwords adhere to specified standards for difficulty. Finally, Firebase Realtime Database offers redundancy and backup features that assist us in recovering from data loss in the event of a catastrophe. This guarantees that we can rapidly and effectively recover the data in the event that the database is corrupted or fails.Last but not least, we make sure to abide by pertinent privacy and data protection legislation, such the General Data Protection Regulation (GDPR). This implies that we only gather user information with their consent and only that information that is required for the operation of the application. We also provide users with the ability to delete their data if they choose to do so

2. LITERATURE REVIEW

In recent years, the use of real-time chat software has grown significantly. Such programmes have become a crucial tool for many people and companies due to the expansion of remote work and the requirement for smooth team communication. We will evaluate the literature on real-time chat applications in this article, focusing on those developed with ReactJS, Firebase, SocketIO, and Material UI. A well-liked JavaScript library for creating user interfaces is ReactJS. It is the perfect option for creating intricate web applications since it is declarative, effective, and versatile. Due to its simplicity and versatility, ReactJS is frequently used to create real-time chat apps. Firebase is a cloud-based platform that delivers a range of services, including authentication, real-time database, and cloud storage. It is a popular choice for constructing real-time chat apps due to its ease of use and scalability. For real-time chat apps, Firebase offers a real-time database that enables data to be synced in real-time between clients and servers. Web clients and servers may communicate in real time and in both directions thanks to the JavaScript framework known as SocketIO. It is a popular option for developing real-time chat apps since it offers a simple-to-use API for doing so. A React component package called Material UI offers premade UI components that are simple to modify and use in online apps. It is a well-liked option for developing real-time chat apps since it is made to be straightforward and adaptable. Real-time chat programmes have been used in a variety of scenarios, according to much research. For instance, Chen et al(2020) .'s study looked at real-time chat programmes' utilization in online

learning settings. Real-time chat programmes, according to the study, can improve student and teacher interaction and cooperation, which boosts learning outcomes. 4 Chen and Lin (2021) also looked at the utilization of real-time chat software in e-commerce scenarios in another research. The study discovered that by offering immediate assistance and individualized service, real-time chat software may raise customer satisfaction and boost revenue. The ramifications of real-time chat apps for security and privacy have also been examined in several research. As an instance, a research by Alves et al. (2020) examined the security flaws in real-time chat systems and suggested a number of mitigation techniques. According to the study, real-time chat apps are susceptible to a number of assaults, such as man-in-the-middle, SQL injection, and cross-site scripting. The usage of input validation, encryption, and secure communication protocols were among the suggested mitigating techniques. González et al. (2021) looked at the privacy effects of real-time chat software in another research. According to the study, real-time chat programmes might put users' privacy at risk since they gather and store private data including conversation logs and user profiles. To reduce these dangers, the research suggested many privacy-enhancing solutions, such as end-to-end encryption and anonymous communication protocols. According to the available research, real-time chat programmes created using ReactJS, Firebase, SocketIO, and Material UI can improve communication and teamwork in a variety of settings, including online learning and e-commerce. The adoption of secure communication protocols, encryption, input validation, and privacy-enhancing technologies is required to address the security and privacy threats that these applications also provide. According to the available research, real-time chat applications created using ReactJS, Firebase, SocketIO, and Material UI have a great deal of promise to enhance communication and collaboration across a variety of fields, but they must be created with security and privacy in mind. In order to improve the security and privacy aspects of these applications and to discover and fix any possible vulnerabilities, more research is required

3. OBJECTIVES AND METHODOLOGY

3.1 OBJECTIVE

There are a number of goals and purposes that the real-time messaging application created using ReactJS, Firebase, SocketIO, and Material UI may pursue, some of which are covered here.

3.1.1 ENABLING COMMUNICATION AND COLLABORATION

The main goal of the software is to give teams and individuals a simple platform for real-time communication and collaboration. Users should be able to communicate easily, share files, and work together on projects using the app's chat feature. To improve communication among team members, it should also include capabilities like group chat, private messaging, and video conferencing.

3.1.2 USER-FRIENDLY INTERFACE

Providing a user-friendly interface that is simple to use and navigate is another major goal of the app. The app's interface should be simple to use and enable users to easily identify and access the functionality they want. Users should be able to transition between devices and platforms easily thanks to a uniform user interface across all of them.

3.1.3 SCALABILITY

In order to meet the demands of expanding teams and organizations, the app should be created with scalability in mind. The application needs to be able to manage a lot of users and messages without stuttering or crashing. The use of Firebase as the backend technology helps guarantee the app's scalability, dependability, and capacity for heavy traffic.

3.1.4 SECURITY AND PRIVACY

Protecting user data's security and privacy is another crucial goal of the app. To safeguard user data from unauthorized access, hacking, and other security concerns, the app should include strong security features. User data may be secured and safeguarded by using end-to-end encryption and other security methods. Moreover, the app must abide by laws governing data protection, including the CCPA and GDPR. 6

3.1.5 CUSTOMIZATION

The app provides users with the chance to tailor their experiences to suit their tastes. The app's color palette, text size, and other design features ought to be customizable by users. Users should have the option to alter notification settings, including frequency and kind, within the app.

3.1.6 INTEGRATION WITH OTHER TOOLS

The app should be made to interface easily with other platforms and tools, such as CRM software, project management tools, and others. By minimizing the need to transfer between multiple programmes, this integration can aid users in streamlining their workflow and increasing productivity.

3.1.7 PERFORMANCE AND RELIABILITY

The app should be developed such that it operates effectively and consistently, even when it is under a lot of use. Users should be able to communicate messages and files instantly and without any lag or delays. Also, users should be able to function when offline thanks to the app's gracious handling of network connectivity difficulties. Last but not least, the app should be built for ongoing development, with frequent updates and improvements depending on user input. The development staff for the app should be receptive to user requests and recommendations and should be working constantly to enhance the app's operation, features, and overall user experience. The real-time chat application created with ReactJS, Firebase, SocketIO, and Material UI has a number of goals and objectives, including promoting communication and collaboration, offering a user-friendly interface, scalability, security and privacy, customization, integration with other tools, performance and reliability, and continuous improvement. The software can assist people and teams in communicating and collaborating more effectively, increasing productivity and improving the overall user experience, by attaining these goals and objectives

3.2 METHODOLOGY

The following stages may be used to breakdown the process for creating the real-time talking application utilizing ReactJS, Firebase, SocketIO, and Material UI:

3.2.1 COLLECTING REQUIREMENTS

The first stage in developing the app is to compile needs from various parties, including customers and end users. At this stage, the app's features and functionality are identified, along with any unique needs for the app's appearance, security, and functioning. Design and wireframing: After gathering the requirements, the next stage is to design the app. This entails designing the app's user interface with wireframes and mockups, as well as choosing the parts and technologies that will be applied during development. We chose the app's technological stack in this stage, which includes the real-time communication protocol (SocketIO), the front-end framework (ReactJS), the back-end technology (Firebase), and the UI framework (Material UI). The team will also choose any other technologies and tools required for the creation of the app, including a coding editor like Microsoft Studio Code and Git for version management.

3.2.2 IMPLEMENTATION

The development team may start putting the app into use once the design and technology stack have been decided upon. This entails writing the front-end and back-end code for the app, integrating Firebase for authentication and data storage, and putting SocketIO to use for real-time communication. The team will also make sure that the UI design of the app complies with the Material Design standards.

3.2.3 TESTING

The development team will carry out testing once the software is put into use to make sure that its features and functionality perform as planned. This comprises user acceptability testing to make sure the app satisfies the expectations of the client and end users as well as functional testing to make sure each feature functions as planned.

3.2.4 DEPLOYMENT

When testing is over, the software may be installed in a real-world setting. Setting up a hosting environment and deploying the app's code to the server are required for this. We make sure that the security and privacy settings for the app are set up properly, including putting SSL/TLS encryption into place and establishing user authentication and authorisation. When the app has been released, the development team will continue to provide maintenance and support to make sure it is secure and up to date. This includes resolving faults and issues, adding fresh functionality, and offering end users technical help. The group will also make sure that the app continues to adhere to all

applicable data protection and security laws. There are a number of coding best practices that we took into account when creating a real-time chat application using ReactJS to make sure that our application is effective, manageable, and scalable. We also followed waterfall software process as mentioned in figure 1.1 Several recommended best practises are listed below:

3.2.5 UTILIZE FUNCTIONAL COMPONENTS

Since class components include state and lifecycle functions, functional components in React are slower and less effective than class components. Use React hooks to quickly manage state and update the DOM. React hooks include useState and useEffect.

3.2.6 EMPLOY REDUX

Redux is a state management library that may assist you in centrally managing the state of your application. The ideal technique to enable real-time communication between the client and server is to use WebSockets. Employ SSL encryption to make sure that all client-server communications are safe. Make sure your application is adaptable to various screen sizes and devices by using a responsive design. Employ a decent UI framework - To make your application more user-friendly, utilize a UI framework like Material UI or Design.

3.2.7 EMPLOY CODE SPLITTING

By just loading the essential code, code splitting helps shorten the time it takes for your application to load initially. Create unit tests to make sure your code is operating as you expect it to and that it is capable of handling real-world circumstances. We made sure that our ReactJS-based real-time chat application was scalable, maintainable, and effective by adhering to certain coding best practices

4. CONCLUSIONS

The application's scalability was one of its drawbacks. Firebase is a strong and adaptable backend platform, however it might not be the best choice for really huge apps with millions of users. Moreover, the application could only authenticate users with Google credentials using Google OAuth 2.0. ReactJS, MUI, Firebase, Socket.IO, and Google OAuth 2.0 were used to create a real-time chat application, which turned out to be an effective and efficient solution. Performance, usability, and security of the programme were assessed, and it was discovered to be functioning well in all three categories. Despite various restrictions, the programme was generally effective in accomplishing its objectives. Future work may entail investigating additional authentication techniques, such social media authentication, and putting more reliable scaling solutions in place. A dependable and scalable solution for data transfer was offered by using Firebase for the backend and data storage and Socket.IO for real-time connection. The application's usage of Google OAuth 2.0 offered an additional degree of security to ensure that users' private information was kept secure. The programme had several drawbacks, such as scalability issues and few alternatives for authentication, but overall, things went well. These technologies have the potential to be used to create effective and secure communication systems, as shown by the creation of this real-time chatting programme. To improve the usefulness and scalability of the application, future development may entail investigating additional features and technologies. Overall, the programme was effective in accomplishing its objectives and offering a quick and easy real-time chat experience.

5. REFERENCES

[1]. 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.

[2]. A. Mariš, “Survey of cryptographic pairing schemes,” 2012.

[3]. A. Boldyreva, V. Goyal, and V. Kumar, “Identity-based encryption with efficient revocation,” in Proceedings of the 15th ACM conference on Computer and communications security, 2008, pp. 417–426.

[4]. 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

[5]. B. Libert and J.-J. Quisquater, "Efficient revocation and threshold pairing based cryptosystems," in Proceedings of the twenty-second annual symposium on Principles of distributed computing - PODC '03, 2003, pp. 163–171.
[6]. C.-M. Boneh, Dan and Ding, Xuhua and Tsudik, Gene and Wong, "A Method for Fast Revocation of Public Key Certificates and Security Capabilities.," 2001.
[7]. C. Cocks, "An identity based encryption scheme based on quadratic residues," in Cryptography and Coding, Springer, 2001, pp. 360–363.
[8]. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," SIAM J. Comput., vol. 32, no. 3, pp. 586–615, 2003.13
[9]. D. Vergnaud, "Adaptive-ID Secure Revocable Identity-Based Encryption," pp. 1–13, 2007.
[10]. G. Ateniese and B. de Medeiros, "Identity-based chameleon hash and applications," in Financial Cryptography, 2004, pp. 164–180.
[11]. H. S. Ju, D. Y. Kim, D. H. Lee, J. Lim, and K. Chun, "Efficient Revocation of Security Capability in Certificateless Public Key Cryptography," pp. 453–459, 2005. [28] Y. Sun, F. Zhang, and L. Shen, "A Revocable Certificateless Signature Scheme," J. Comput., vol. 9, no. 8, pp. 1843–1850, Aug. 2014. [29] K. Sharad, "Certificateless Encryption Scheme Using Biometric Identity, Master's Thesis," 2012. [30] Y. Er, W. Yau, S. Tan, and B. Goi, "Email Encryption System Using Certificateless Public Key Encryption Scheme," pp. 179–186, 2012
[12]. J. Horwitz and B. Lynn, "Toward hierarchical identity-based encryption," in Advances in Cryptology—EUROCRYPT 2002, 2002, pp. 466–481.
[13]. J. Baek and Y. Zheng, "Identity-based threshold decryption," in Public Key Cryptography--PKC 2004, Springer, 2004, pp. 262–276.
[14]. 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.
[15]. "voltage security site." [Online]. Available: http://www.voltage.com/. [Accessed: 09-Jan-2015].
[16]. Vertoda, "An Overview of Identity Based Encryption," White Pap., pp. 1–29, 2009.