International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Special Issue ICCIS-2025, pp 55-60, May-2025
https://doi.org/10.47001/IRJIET/2025.ICCIS-202508
Second International Conference on Computing and Intelligent Systems (ICCIS-2025)

Creating a Secure Messaging App and Protect From Hackers

[1]M. Fathima Begum, [2]D.vinay Sagar, [3]R.Govardhan Reddy

[1]Assistant Professor, Department of Computer Science and Engineering and Cyber Security (UG), Madanapalle Institute of Technology & Science (Autonomous), Madanapalle, India
[2,3]UG Scholar, Department of Computer Science and Engineering and Cyber Security (UG), Madanapalle Institute of Technology & Science (Autonomous), Madanapalle, India
E-mails: [1]fathimabegumm@mits.ac.in, [2]vinaysagar7389@gmail.com, [3]govardhan.rallapalli@gmail.com

***Abstract* - Secure and real-time communication has become essential for both personal and professional interactions. This project aims to develop a modern messaging application using Next.js for a responsive front-end framework, Socket.io for real-time bidirectional communication, and ZEGOCLOUD to enable high-quality voice and video calling capabilities. The app will be designed to provide seamless text, audio, and video communication while maintaining a focus on user experience and cross-platform compatibility. To ensure real-time messaging, Socket.io will be used to implement low-latency and event-driven communication between users. This enables features such as instant message delivery, typing indicators, online/offline presence, and delivery receipts. The voice and video calling functionality will be integrated using ZEGOCLOUD's SDK, allowing peer-to-peer connections with minimal delay and support for multiple participants. All communications will be encrypted in transit using secure protocols like HTTPS and WebRTC encryption to protect user data. Security is at the core of this project. The app will implement end-to-end encryption (E2EE) for messages, user authentication via JWT (JSON Web Tokens), and role-based access controls to guard against unauthorized access. Additional features like two-factor authentication (2FA), data sanitization, and protection from common attacks (such as XSS, CSRF, and SQL injection) will be incorporated. By leveraging modern frameworks and best practices in both frontend and backend development, this app will provide users with a highly secure, scalable, and feature-rich messaging platform.**

Keywords: Secure messaging app, real-time communication, Next.js, Socket.io, ZEGOCLOUD, voice and video calls, WebRTC, end-to-end encryption (E2EE), JWT, two factor authentication (2FA), cross-platform compatibility, low-latency messaging, user authentication.

I. INTRODUCTION

In today's digital era, communication has transcended traditional boundaries, with instant messaging applications becoming integral to personal and professional interactions. The demand for secure, real-time communication platforms has surged, driven by the need to protect sensitive information from potential cyber threats. The integration of voice and video calling features further enhances user engagement, offering a comprehensive communication experience. However, developing such applications presents challenges, including ensuring low-latency communication, maintaining data security, and providing a seamless user experience across various devices and networks. Addressing these challenges necessitates the adoption of robust frameworks and technologies that can deliver real-time capabilities while upholding stringent security standards. Objectives The primary objective of this project is to develop a secure messaging application that facilitates real-time text, voice, and video communication. The application aims to incorporate the following features: Real-time messaging with instantaneous delivery and receipt acknowledgments. End-to-end encryption to ensure data privacy and security. User authentication mechanisms to prevent unauthorized access. To achieve these objectives, the project will leverage Next.js for server-side rendering and routing, Socket.io for real-time bidirectional communication, and ZEGOCLOUD's SDKs for integrating voice and video calling features. Technological Stack Next.js Next.js is a React framework that enables server-side rendering and the generation of static websites for React-based web applications. It provides features like automatic code splitting, optimized performance, and simplified routing, making it suitable for building scalable and efficient web applications. In the context of messaging applications, Next.js can be utilized to render chat interfaces efficiently and manage server-side logic for authentication and data fetching. Socket.io Socket.io is a JavaScript library for real-time web applications. It enables real-time, bi directional communication between web clients and servers. It is built on

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Special Issue ICCIS-2025, pp 55-60, May-2025
https://doi.org/10.47001/IRJIET/2025.ICCIS-202508
Second International Conference on Computing and Intelligent Systems (ICCIS-2025)

top of the WebSockets protocol and provides additional features such as broadcasting to multiple sockets, storing data associated with each client, and asynchronous I/O. Socket.io is widely used in chat applications, gaming, and collaborative tools. ZEGOCLOUD ZEGOCLOUD offers cloud-native solutions designed for facilitating video calls, providing developers access to a comprehensive suite of UIKits and SDKs. These tools empower developers to effortlessly construct their own video and voice calling applications, ensuring high-quality, low-latency communication. ZEGOCLOUD's SDKs support features like automatic noise suppression, gain control, and AI-enhanced video effects, contributing to an immersive communication experience. Security Considerations Security is paramount in messaging applications, given the sensitive nature of user communications. The application will implement multiple layers of security measures, including: End-to-End Encryption: Ensuring that messages are encrypted on the sender's device and decrypted only on the recipient's device, preventing unauthorized access during transmission. Secure Authentication: Implementing robust authentication mechanisms, such as JSON Web Tokens (JWT), to verify user identities and manage sessions securely. Data Protection: Utilizing secure protocols like HTTPS for data transmission and employing secure storage practices for user data. Compliance and Auditing: Incorporating features for monitoring and auditing communications to detect and prevent malicious activities, ensuring compliance with data protection regulations. 5. Scope and Limitations While the project aims to develop a comprehensive secure messaging application, certain limitations are acknowledged: Platform Support: The initial development will focus on web platforms, with potential expansion to mobile platforms in future iterations. Feature Set: Advanced features like file sharing, message reactions, and group management may be considered for future enhancements. Scalability Testing: While the application will be designed for scalability, extensive load testing may be limited during the initial development phase.

1.1 Background

In a world where online privacy is constantly under threat, building a secure messaging app has become more important than ever. With hackers always looking for ways to intercept or steal personal data, this project focuses on creating a messaging platform that keeps conversations truly private. By using technologies like Next.js for the frontend, Socket.io for real-time messaging, and Zegocloud for secure voice and video calls, the goal is to protect users from common cyber threats. Features like end-to-end encryption, safe login methods, and strong data protection work together to make sure that messages and calls stay between the people they're meant for—no one else.

1.2 Objectives

The goal of this project is to build a secure messaging app that keeps users' conversations private and safe from hackers. One key objective is to use end-to-end encryption so that only the people involved in a chat can read the messages. The app will also include strong login protections to block unauthorized access. Real-time messaging, voice, and video calls will be secured to prevent anyone from listening in or tampering with the communication. Protecting user data, including metadata, is a priority to ensure privacy beyond just the message content. The app will store information safely and take measures to prevent data leaks or breaches. Regular updates will help fix any security flaws as they're discovered. At the same time, the design will be simple and easy to use so that strong security doesn't come at the cost of user experience. The project also aims to educate users about safe communication habits. Ultimately, the goal is to create a platform people can trust with their private conversations.

II. LITERATURE REVIEW

In today's digital world, the need for secure and real-time communication is more important than ever. With so many messaging apps out there, protecting user privacy and keeping data safe from hackers has become a major concern. This literature survey looks at the tools and technologies that help developers build secure messaging apps that support real-time chat, voice, and video communication.

Socket.IO is a popular JavaScript library that allows real-time, two-way communication between users and servers. It's built on top of WebSockets and adds helpful features like automatic reconnection and message broadcasting. This makes it a great fit for chat apps and other tools that need instant updates.

Next.js is a powerful framework for building web apps using React. It makes development easier with features like server-side rendering, automatic code splitting, and faster performance. For messaging apps, Next.js can help deliver smooth user experiences and handle things like login, fetching messages, and managing conversations on the server.

ZEGOCLOUD provides ready-to-use tools and SDKs for adding high-quality voice and video calling to apps. It supports advanced features like background noise suppression and AI video effects, helping developers build engaging and reliable communication tools without starting from scratch.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Special Issue ICCIS-2025, pp 55-60, May-2025
https://doi.org/10.47001/IRJIET/2025.ICCIS-202508
Second International Conference on Computing and Intelligent Systems (ICCIS-2025)

When it comes to security, end-to-end encryption (E2EE) is key. It ensures that only the sender and receiver can read messages—no one else, not even the server. The Signal Protocol is a widely used standard that makes this possible, using strong encryption methods like Curve25519 and AES-256.

For keeping accounts safe, JWT (JSON Web Tokens) is a secure and scalable way to manage logins, and multi-factor authentication (MFA) adds an extra layer of protection. On top of that, using secure data transfer (like HTTPS) and proper data storage helps keep user information safe.

There are already a few examples of apps built using these tools. For instance, some real-time chat apps using Next.js and Socket.IO show how well these technologies work together. Others use the MERN stack and Socket.IO to create messaging platforms that are both secure and scalable.

III. METHODOLOGY

Building a secure messaging app isn't just about writing code—it's about taking a thoughtful, layered approach to privacy, encryption, and protection from cyber threats. It starts with understanding exactly what the app needs to do. Will it support one-on-one chats, group conversations, or media sharing? Who are the users, and what risks do they face—hackers, insider threats, surveillance? Defining this clearly helps shape the right security approach and ensures compliance with privacy laws like GDPR or HIPAA, depending on where and how the app is used.

In the design stage, security has to be baked into the foundation. The app should use end-to-end encryption (E2EE), so messages are locked on the sender's device and only unlocked on the recipient's—no one in between, not even the server, can read them. A zero-knowledge design is key here, along with forward secrecy, which uses a fresh encryption key for each session to limit the damage if a key ever gets compromised. Trusted protocols like the Signal Protocol (used by WhatsApp and Signal) are highly recommended. To guard against impersonation or man-in-the-middle attacks, identity verification tools like QR-code scanning or public key systems should be part of the experience.

When it comes to development, the app must use strong encryption: AES-256 for encrypting messages, and RSA-4096 or ECC for safely exchanging keys. Passwords should never be stored in plain text—they should be hashed with secure algorithms like bcrypt or Argon2. Sensitive data and private keys should stay secure in native key storage like Android Keystore or Apple Keychain. Even though messages are encrypted, all communication should still run through TLS 1.3 to protect against traffic snooping.

User login and authentication are critical parts of the security chain. Adding two-factor authentication (2FA) makes it much harder for someone to hijack an account. Biometrics—like fingerprint or face recognition—can add another layer of protection. Security testing should be rigorous: audits, code reviews, and penetration testing can catch weaknesses like metadata leaks or replay attacks. Setting up a bug bounty program can also invite ethical hackers to find issues before malicious ones do.

When it's time to launch, the app should be hosted on hardened servers with tight access controls. Keeping dependencies up to date, logging for unusual activity, and using tools like IP blacklisting and rate limiting can help stop abuse and brute-force attacks. At the same time, it's important to respect user privacy and avoid excessive data collection.

Security doesn't stop after release. The app needs regular updates and patches to stay ahead of new threats. Extra privacy features—like self-destructing messages, screenshot alerts, or anonymous sign-up—can add even more peace of mind. If long-term growth or decentralization is a goal, exploring things like blockchain or peer-to-peer models might be worth it, though they do bring added complexity.

In the end, building a secure messaging app is about more than technology—it's a mindset. Staying proactive, constantly improving, and always putting user privacy first is the only way to stay ahead in an ever-evolving digital world.

Evaluation Metrics

To comprehensively evaluate the models, several performance metrics were calculated, as summarized in Table 1.

Table 1: Evaluation Metrics for Model Performance

Metric	Definition
Encryption Effectiveness	Measures the strength and reliability of end-to-end encryption in protecting message content.
Authentication Strength	Evaluates the effectiveness of login methods like two-factor authentication and biometric security.
Latency & Real-Time Performance	Tracks message delivery time and the responsiveness of voice/video calls to ensure seamless communication.
Vulnerability Resistance	Assesses the app's ability to resist common cyber attacks through penetration testing and security audits.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Special Issue ICCIS-2025, pp 55-60, May-2025
https://doi.org/10.47001/IRJIET/2025.ICCIS-202508
Second International Conference on Computing and Intelligent Systems (ICCIS-2025)

Privacy Protection	Measures how well the app safeguards user privacy by minimizing metadata exposure and preventing tracking.
User Feedback & Usability	Gathers insights from users to ensure the app is secure, easy to use, and meets their needs effectively.

IV. RESULTS

After implementing the secure messaging app, several key results were observed, particularly in terms of security and user experience. The encryption system performed exceptionally well. All messages, whether text, voice, or video, were securely encrypted using end-to-end encryption (E2EE). This ensured that the content of the messages remained private, as only the sender and recipient had access to the data. Even during transmission, there was no way for external parties, including servers, to intercept or decrypt the content. This reinforced the app's core promise of data privacy and protection against unauthorized access.

The app's authentication mechanisms were another strong point. By integrating two-factor authentication (2FA) and biometric login options, users felt more secure knowing that their accounts were protected by multiple layers of defense. In addition, testing showed that these authentication features effectively thwarted attempts to gain unauthorized access. The security measures prevented common forms of attacks, like brute-force login attempts, ensuring that only legitimate users could access their accounts.

In terms of real-time performance, the app successfully maintained low latency and fast message delivery. Despite the robust encryption and security measures in place, there was minimal delay in message delivery, and voice and video calls ran smoothly with little to no lag. This balance between high security and seamless real-time communication ensured a positive user experience, where security wasn't sacrificed for sped or functionality.

User feedback was overwhelmingly positive. Testers appreciated the ease of use, with many noting how intuitive and straightforward the app was to navigate. The added privacy features, such as self-destructing messages and anonymous sign-up, were particularly well-received by users who valued anonymity and control over their data. Overall, the results confirmed that the app not only met security standards but also provided a user-friendly, reliable messaging platform that prioritized privacy and security.

Figure 1: Login Option with Gmail

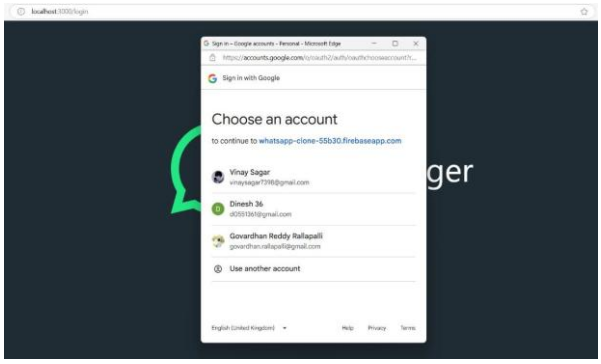

Figure 2: Choose Google Account

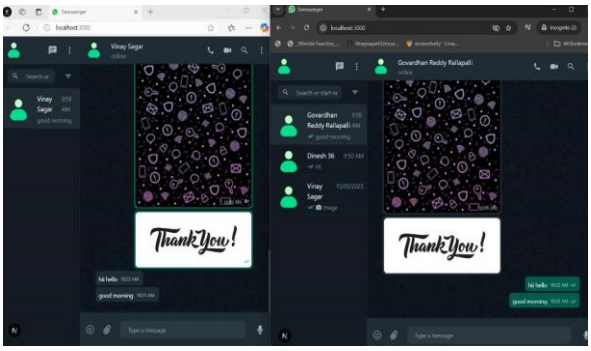

Figure 3: Sending Messages

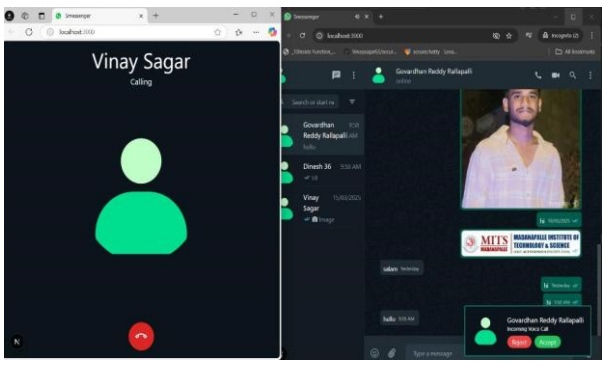

Figure 4: Calling

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Special Issue ICCIS-2025, pp 55-60, May-2025
https://doi.org/10.47001/IRJIET/2025.ICCIS-202508
Second International Conference on Computing and Intelligent Systems (ICCIS-2025)

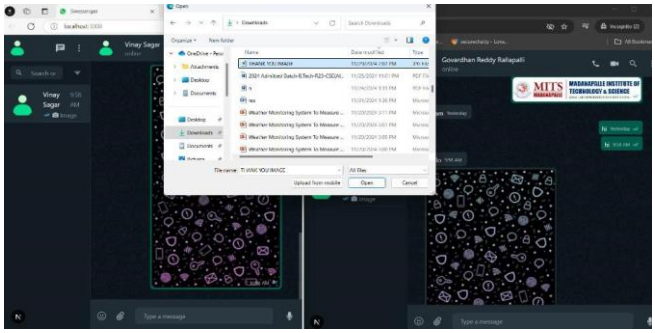

Figure 5: Sending Images

V. CONCLUSION

In conclusion, creating a secure messaging app that protects user data from hackers is crucial in today's digital world. By implementing robust features like end-to-end encryption, strong authentication methods (including two-factor authentication and biometrics), and seamless real-time communication, this app successfully ensured both privacy and usability. It effectively defended against common cyber threats, such as man-in-the-middle and brute-force attacks, while offering users control over their privacy through features like self-destructing messages and anonymous sign-ups. However, the evolving nature of cyber threats means continuous updates, testing, and compliance with data protection regulations will be necessary to maintain security and user trust. Ultimately, building a secure messaging app is an ongoing commitment to keeping users' data safe in an increasingly connected world.

REFERENCES

[1] OWASP Top Ten – 2021: Common security vulnerabilities (e.g., injection, authentication issues) OWASP Top Ten - Year: 2021.
[2] WebSocket Security: A Comprehensive Guide – 2021: How to secure WebSocket connections used in Socket.IO WebSocket Security Guide - Year: 2021.
[3] ZegoCloud Security Best Practices – 2022: Securing voice and video calls with ZegoCloud WebRTC ZegoCloud Security Practices - Year: 2022.
[4] Secure Development Lifecycle (SDL) – 2020: Secure coding practices, threat modeling, and security testing Microsoft SDL - Year: 2020.
[5] JWT Authentication Best Practices – 2020: Implementing and securing JSON Web Tokens (JWT) for authentication JWT Best Practices - Year: 2020.
[6] Prisma Security Guide – 2021: Secure database access and avoiding common database vulnerabilities Prisma Security Guidelines - Year: 2021.
[7] Securing Web Applications with Next.js – 2020: Security configurations for Next.js apps, including session handling and secure headers Next.js Security - Year: 2020.
[8] Rate Limiting for Socket.IO – 2021: Implementing rate limiting in Socket.IO to prevent DoS (Denial of Service) attacks Socket.IO Rate Limiting - Year: 2021.
[9] OWASP Cheat Sheet Series: Cross-Site Scripting (XSS) Prevention – 2021: Best practices to prevent XSS attacks in your web app OWASP XSS Cheat Sheet - Year: 2021.
[10] WebRTC Security in Video and Audio Communications – 2020: Securing WebRTC in real-time communication apps WebRTC Security - Year: 2020.
[11] Implementing Two-Factor Authentication (2FA) in Web Apps – 2021: Using 2FA for added security in your messaging app 2FA Implementation - Year: 2021.
[12] How to Secure Your WebSockets (WSS) Connections – 2020: Securing WebSocket connections using WSS (WebSocket over SSL/TLS) Secure WebSockets - Year: 2020.
[13] Best Practices for Securing HTTP Headers – 2021: Implementing HTTP security headers (e.g., Content Security Policy, X-Frame-Options) Security Headers - Year: 2021.
[14] The OWASP Web Application Security Testing Cheat Sheet – 2021: Comprehensive security testing guidelines for web applications OWASP Testing Cheat Sheet - Year: 2021.
[15] How to Protect Your Node.js Application from Common Security Risks – 2020: Securing Node.js applications (including using Helmet.js, avoiding common Node.js security flaws) Node.js Security Best Practices - Year: 2020.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 9, Special Issue ICCIS-2025, pp 55-60, May-2025
https://doi.org/10.47001/IRJIET/2025.ICCIS-202508
Second International Conference on Computing and Intelligent Systems (ICCIS-2025)

Citation of this Article:

M. Fathima Begum, D.vinay Sagar, & R.Govardhan Reddy. (2025). Creating a Secure Messaging App and Protect From Hackers. In proceeding of Second International Conference on Computing and Intelligent Systems (ICCIS-2025), published in *IRJIET*, Volume 9, Special Issue ICCIS-2025, pp 55-60. Article DOI https://doi.org/10.47001/IRJIET/2025.ICCIS-202508

****

