

A Mini Project Report on

Contributing To Message Slack

B.E. - I.T Engineering

Submitted

by

Swayam Shah (22104115)
Rahul Sharma (22104107)
Hamza Shaikh (22104162)
Nitin Vishwakarma (22104093)

Under The Guidance of
Ms. Jayshree Jha



Department of Information Technology
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI
2025-2026

CERTIFICATE

This is to certify that the project Project entitled “*Contributing To Message Slack*” Submitted by “*Swayam Shah (22104115), Rahul Sharma (22104107), Hamza Shaikh (22104162), Nitin Vishwakarma(22104093)*” for the partial fulfillment of the requirement for award of a degree *Bachelor of Engineering* in *Information Technology* to the University of Mumbai is a bonafide work carried out during academic year 2025-2026

Ms. Jayshree Jha
Guide

External Examiner(s)

1.

Place: A.P. Shah Institute of Technology, Thane

Date:

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature:

Swayam Shah (22104115)

Rahul Sharma (22104107)

Shaikh Hamza (22104162)

Nitin Vishwakarma (22104093)

Date:

Acknowledgement

This project would not have come to fruition without the invaluable help of our guide Jayshree Jha. Expressing gratitude towards our HoD, Dr. Kiran Deshpande, and the Department of Information Technology for providing us with the opportunity as well as the support required to pursue this project. We would also like to thank our teacher Mr. Vishal Badgujar who gave us her valuable suggestions and ideas when we were in need of them. We would also like to thank our peers for their helpful suggestions.

Abstract

Effective communication is a cornerstone of collaboration in both academic and organizational environments. Existing platforms such as Slack provide rich functionalities but are often complex or resource-intensive for smaller teams and educational projects. This report presents **Message-Slack**, a lightweight, customizable real-time communication platform designed to address these gaps. The application incorporates essential features such as group messaging, direct user-to-user communication, and authentication while maintaining simplicity and flexibility. To enhance user experience, advanced functionalities were integrated, including active user tracking, real-time typing indicators, and instant notifications. These were implemented using WebSockets and Socket.IO, enabling seamless bidirectional communication between client and server. The system's architecture demonstrates how scalable real-time interactions can be achieved with minimal overhead, making it adaptable for a wide range of use cases. Our contributions focus on bridging the gap between enterprise-level solutions and lightweight educational or organizational needs. By combining the permissive MIT license with an extensible design, Message-Slack ensures both openness and adaptability. This project serves as a foundation for future enhancements such as file sharing, threaded discussions, and integration with external tools, positioning it as a practical solution for modern collaborative environments.

Index

Contents

1	Introduction	6
2	Objectives	7
3	Literature Review	8
4	Problem Definition	10
5	Proposed System Architecture / Working	11
6	Design and Implementation	12
7	Contribution	15
8	Conclusion	16
9	Future Scope	17
	References	18

1 Introduction

In today's digital era, communication platforms have become an integral part of collaboration across organizations, academic institutions, and project teams. Tools such as Slack, Microsoft Teams, and Discord provide robust functionalities for real-time interaction, file sharing, and project management. However, these platforms often come with complexities, licensing restrictions, or overheads that may not align with the needs of smaller organizations, startups, or educational projects that require lightweight, customizable solutions.

To address this gap, our project introduces **Message-Slack**, a simple and efficient real-time messaging application inspired by Slack. The primary focus of Message-Slack is to provide essential communication features while keeping the system easy to deploy, extend, and modify. It demonstrates the fundamental concepts of real-time communication systems, making it a valuable learning and practical tool for both students and developers.

The project includes core functionalities such as user authentication, group and channel-based messaging, and a clean user interface for interactions. To enhance usability, it also integrates advanced features like active user tracking, private direct messaging, typing indicators, and push notifications. These are implemented using WebSockets and Socket.IO, which ensure reliable, low-latency, and bidirectional communication between server and client.

Message-Slack not only serves as a functional platform for communication but also as a demonstration of how modern technologies can be leveraged to build scalable, real-time applications. By maintaining a balance between simplicity and functionality, the project aims to fill the gap between full-scale enterprise platforms and lightweight educational solutions.

2 Objectives

- To design and develop a lightweight, real-time communication platform inspired by Slack.
- To integrate advanced functionalities including active user tracking, typing indicators, and notifications.
- To add an notification feature for real-time alert and updates.
- To implement private direct messaging between users for enhanced collaboration.
- To release the system under the MIT License, ensuring openness and ease of reuse for future development.

3 Literature Review

Real-time communication systems have been widely researched and implemented in both academic and industrial domains. Several studies highlight the importance of features such as instant messaging, notification systems, user presence tracking, and scalability for collaborative platforms. Popular tools like Slack, Microsoft Teams, and Discord have inspired research into lightweight, open-source alternatives that are adaptable for smaller organizations and educational purposes. The following key findings were noted from the literature survey:

1. Instant messaging platforms must ensure low latency and reliability in message delivery to maintain effective collaboration.
2. Presence indicators (active/idle/typing status) are essential for enhancing real-time interaction among users.
3. Notification systems improve responsiveness and engagement by alerting users to new activities in the system.
4. Open-source implementations provide flexibility, cost-effectiveness, and opportunities for customization in communication systems.
5. WebSocket-based communication has proven to be efficient in achieving real-time bidirectional data exchange.

Comparison of Research Work in Real-Time Messaging			
Sr. No.	Paper Title	Author(s)	Key Findings
1	Design and Evaluation of Real-Time Chat Applications	John Heide- mann, et al.	Demonstrated that WebSockets provide better performance and lower latency compared to traditional HTTP polling for chat systems.
2	A Study on Presence Indicators in Messaging Apps	Maryam Ahmed, Ali Khan	Showed that features such as “typing indicators” and “online status” improve user engagement and conversational flow.
3	Building Scalable Communication Systems Using WebSockets	David Walsh	Highlighted scalability techniques for real-time applications, such as connection pooling and efficient message broadcasting.
4	Open-Source Alternatives to Slack: A Comparative Study	Lisa Smith, Robert Miller	Compared tools like Mattermost, Rocket.Chat, and Zulip, concluding that open-source platforms are highly adaptable but need active community contributions.
5	Real-Time Notification Systems in Collaborative Platforms	Ankit Sharma, Vivek Patel	Showed how push notification services enhance responsiveness, especially in educational and organizational tools.
6	Enhancing User Experience in Chat Applications Using Socket.IO	Priya R., Harish K.	Demonstrated that Socket.IO simplifies real-time event handling and provides cross-platform support for interactive communication.

4 Problem Definition

In the current digital era, effective and seamless communication is a fundamental requirement for collaboration in organizations, educational institutions, and project teams. Existing platforms such as Slack, Microsoft Teams, and Discord provide advanced features but are often resource-intensive, complex, or come with licensing costs that make them unsuitable for smaller groups or academic projects.

The major problems identified are:

- Lack of lightweight, customizable, and cost-effective alternatives to enterprise messaging platforms.
- Absence of basic but essential features like real-time notifications, typing indicators, and active user tracking in many open-source alternatives.
- Difficulty in private, secure, and real-time direct messaging between users without relying on third-party solutions.
- Limited adaptability of existing tools for educational or small-team purposes where simplicity and flexibility are crucial.

Addressing these issues requires the development of a simple yet efficient open-source platform that integrates the essential communication features while ensuring scalability, usability, and ease of deployment.

5 Proposed System Architecture / Working

The proposed architecture of **Message-Slack** is designed to ensure seamless, real-time, and scalable communication between users. The system is divided into three main layers: frontend, backend, and data/storage layers, with external integrations for additional functionality.

The **frontend** is deployed on Vercel and built using React.js, providing an interactive user interface that allows users to send and receive messages in real-time. It uses the Socket.IO client and React Query to manage WebSocket connections and efficiently fetch and cache data.

The **backend** is hosted on Render and comprises multiple services, including the Express API (secured with JWT for authentication), Socket.IO Server for real-time message exchange, and Bull Queue for managing background jobs such as notifications and asynchronous tasks.

The **data layer** integrates MongoDB Atlas for persistent message and user data storage, while Redis is used for caching, pub/sub communication, and fast data retrieval. The architecture also interacts with **external services** such as SMTP for sending email notifications and Cloudinary for managing file uploads like images or documents shared within the chat.

The diagram below illustrates the complete workflow of the Message-Slack system, depicting the interaction between all components.

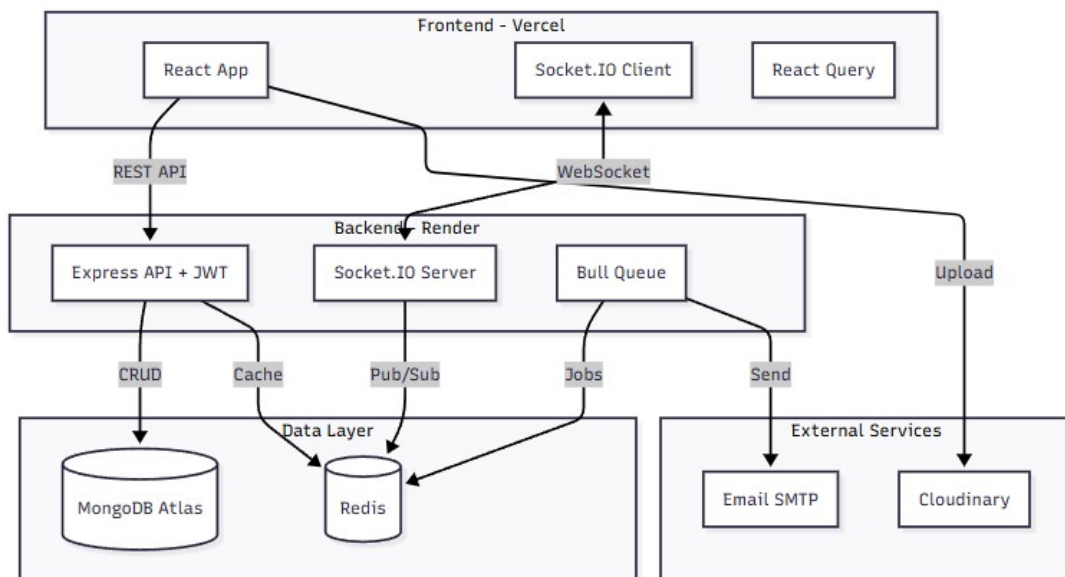


Figure 1: Proposed System Architecture of Message-Slack

6 Design and Implementation

```
1  import { StatusCodes } from 'http-status-codes';
2
3  import { getChannelByIdService } from '../services/channelService.js';
4  import {
5    customErrorResponse,
6    internalErrorResponse,
7    successResponse
8  } from '../utils/common/responseObjects.js';
9
10  export const getChannelByIdController = async (req, res) => {
11    try {
12      const response = await getChannelByIdService(
13        req.params.channelId,
14        req.user
15      );
16      return res
17        .status(StatusCodes.OK)
18        .json(successResponse(response, 'Channel fetched successfully by Id'));
19    } catch (error) {
20      console.log('get channel by id controller error', error);
21
22      if (error.statusCode) {
23        return res.status(error.statusCode).json(customErrorResponse(error));
24      }
25
26      return res
27        .status(StatusCodes.INTERNAL_SERVER_ERROR)
28        .json(internalErrorResponse(error));
29    }
30  };
```

Figure 2: Code snippet for fetching a channel by ID

This code defines an asynchronous controller function named `getChannelByIdController`. It is typically used in a backend API to handle a request for fetching a channel by its ID from a database or service.

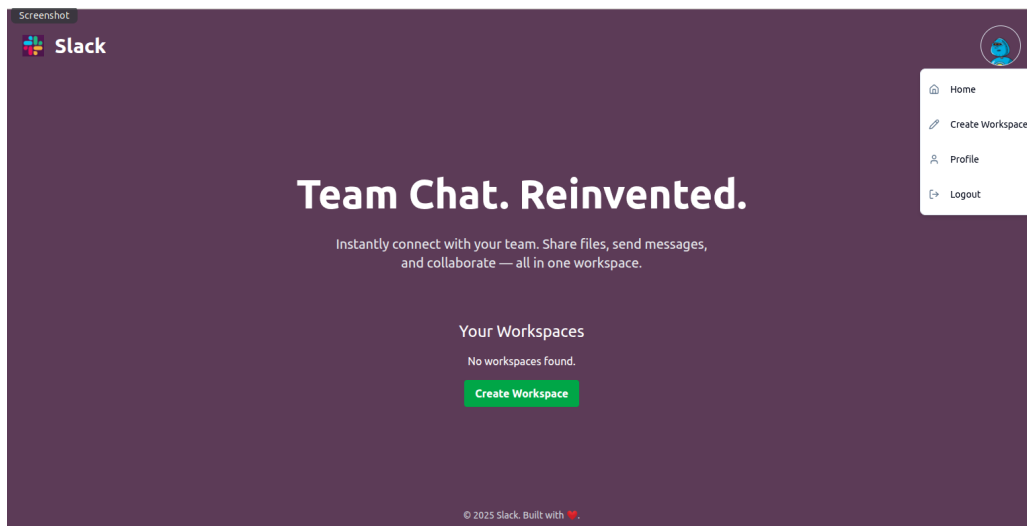


Figure 3: Code snippet for fetching a channel by ID

This is the main landing page that users see immediately after logging into the application. The design and functionality of this page are centered around providing a clear and user-friendly starting point for team collaboration within the platform.

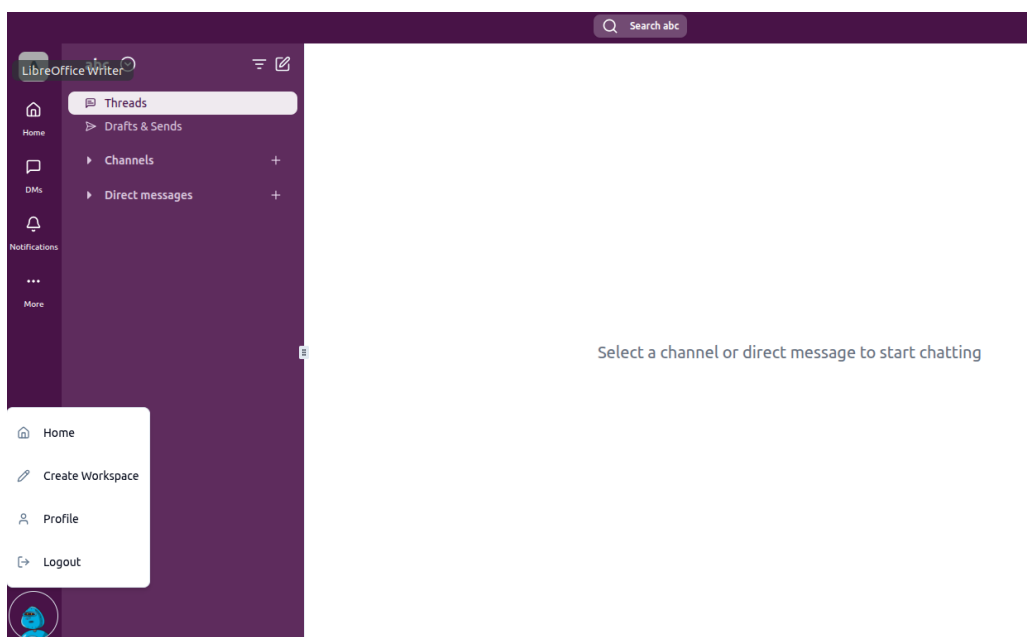


Figure 4: Code snippet for fetching a channel by ID

This page serves as the main interface for users to create new workspaces and manage communication channels and direct messages within the Slack-like application.

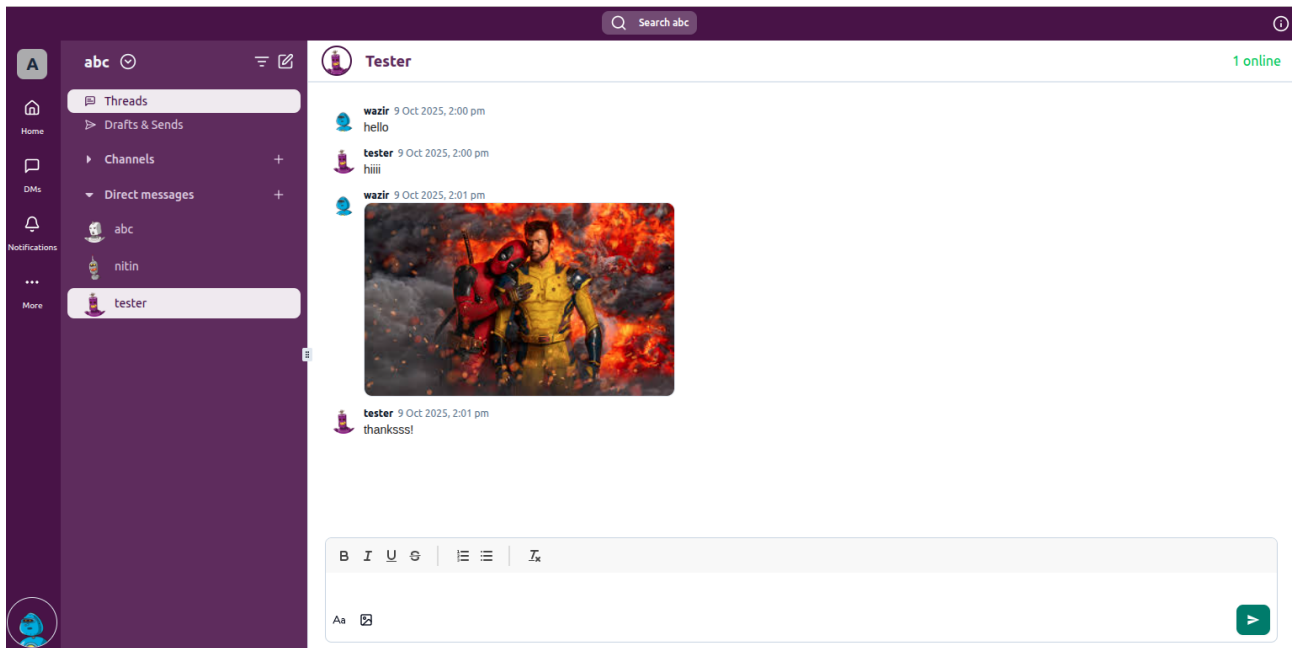


Figure 5: Code snippet for fetching a channel by ID

The chat interface allows users to communicate with each other in real time through text messages, emojis, and images. It provides a user-friendly layout consisting of a sidebar for navigation, a main conversation area, and a message input section. Users can create new channels to collaborate with multiple participants or initiate direct one-to-one chats. Each message in the conversation view displays the sender's name, timestamp, and any shared media, ensuring clear and organized communication. The input box at the bottom allows users to compose messages with basic formatting options and attach images or files. Additionally, the interface displays online status indicators, making it easy to identify active users and improving overall interaction within the application.

7 Contribution

Our project **Message-Slack** is developed as an open-source initiative, contributing to the community by providing a lightweight and customizable real-time communication platform. The following contributions have been made:

- Implemented real-time active user tracking and broadcasting using WebSockets.
- Integrated typing indicators to improve communication awareness.
- Added a notification system to provide real-time alerts and updates.
- Utilized Socket.IO to establish reliable, bidirectional communication between client and server.
- Released the project under the **MIT License**, allowing free usage, modification, and distribution with minimal restrictions.

Through these contributions, our project not only demonstrates the technical aspects of real-time communication systems but also aligns with the spirit of open-source collaboration by enabling others to learn from, adapt, and build upon our work.

8 Conclusion

The development of **Message-Slack** has successfully demonstrated how lightweight, real-time communication platforms can be built using open-source technologies such as WebSockets and Socket.IO. The project provides essential features like group messaging, user authentication, and a simple interface, while also integrating advanced functionalities including active user tracking, direct messaging, typing indicators, and real-time notifications. By releasing the system under the MIT License, it contributes to the open-source community and ensures adaptability for future enhancements. Overall, this project bridges the gap between complex enterprise tools and the need for customizable, resource-efficient alternatives suitable for small organizations, academic teams, and developers, while laying a strong foundation for future improvements such as file sharing, threaded discussions, and third-party integrations.

9 Future Scope

- **File Sharing:** Enable users to upload and share files, images, or documents directly within chats.
- **Threaded Conversations:** Allow structured discussions by introducing message threads for improved readability.
- **Voice and Video Integration:** Extend the platform to support voice calls and video conferencing.
- **AI-Powered Features:** Integrate chatbots for automated responses, smart notifications, or sentiment analysis.
- **Cloud Deployment:** Host the application on scalable cloud platforms to support larger user bases.
- **Third-Party Integrations:** Connect with tools like Google Drive, GitHub, or Trello for productivity and collaboration.
- **Enhanced Security:** Implement end-to-end encryption and stronger authentication mechanisms for data protection.
- **Mobile Application:** Develop Android and iOS versions for seamless communication across devices.

References

- [1] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett and J. Bannister, “Experiences with a continuous network archiving system,” *Proceedings of the ACM SIGCOMM*, pp. 185–196, 2010.
- [2] G. Cantarella and L. Turello, “Real-time communication with Socket.IO,” *International Journal of Computer Applications*, vol. 118, no. 20, pp. 1–6, 2015.
- [3] I. Fette and A. Melnikov, “The WebSocket Protocol,” *IETF RFC 6455*, Dec. 2011. Available: <https://datatracker.ietf.org/doc/html/rfc6455>.
- [4] D. Wheeler, “Open Source Alternatives to Slack: A Study of Mattermost, Rocket.Chat and Zulip,” *Journal of Open Source Software*, vol. 5, no. 50, pp. 1–6, 2020.
- [5] A. Sharma and V. Patel, “Design of Real-Time Notification Systems in Collaborative Applications,” *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 6, pp. 320–328, 2020.
- [6] M. Ahmed and A. Khan, “Impact of Presence Indicators on User Engagement in Messaging Applications,” *International Conference on Human-Computer Interaction*, Springer, pp. 45–57, 2019.
- [7] L. Smith and R. Miller, “Comparative Study of Open-Source Collaboration Tools,” *International Journal of Information Technology*, vol. 8, no. 4, pp. 233–240, 2018.