|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
|  | | | |
|  | |  | |  | |

# Abstract:

*This study investigates the creation and enhancement of a resilient real-time chat application platform through the integration of NextJs, Redis database, and Google OAuth NextAuth authentication. The goal of integrating these technologies is to improve the chat application's overall scalability, security, and user experience. The paper explores the nuances of using Redis as a high-performance, in-memory data store for effective data management, leveraging NextJs to create a contemporary and responsive user interface, and putting Google OAuth NextAuth authentication into practice for safe user authentication.*

*Additionally, the study looks into integrating Edge Computing into the design with the goal of reducing latency and enhancing responsiveness by moving computational operations closer to the users. The study investigates the possible advantages of dispersing computer power to the periphery, facilitating expedited data processing and mitigating reliance on centralised servers. This innovative method advances chat applications, especially in situations when real-time communication and reduced latency are essential.*

***The study includes performance assessments, a comparative analysis of the suggested architecture against conventional configurations, and implementation details that are applicable to real-world scenarios. Furthermore, the study addresses possible obstacles and factors to be taken into account while integrating Edge Computing within the framework of real-time communication platforms. This study paper offers significant insights for developers, system architects, and researchers who aim to integrate cutting-edge technology to improve the efficiency and scalability of current chat apps. This is achieved by examining these components holistically.***

# Introduction

The creation of cutting-edge chat programmes has been spurred by the constant requirement for real-time communication. This project explores the creation and deployment of a feature-rich chat programme that puts an emphasis on effective communication and user experience. This programme offers a number of important features and makes use of a contemporary technology stack.  
  
Real-time Communication: Users can connect and communicate without any delays thanks to the application's smooth real-time chat feature. This is made possible by integrating WebSockets, which guarantees fast message delivery and promotes a more lively and interesting chat experience.  
Simplified Authentication: NextAuth.js-powered Google OAuth integration offers a quick and safe way to log in. With their current Google credentials, users can sign up or log in, doing away with the need to create additional accounts and remember passwords.

Simple Dashboard Design: The application provides easy-to-use functionality for initiating chat requests via email addresses, receiving incoming requests, and having real-time conversations.   
Scalable Infrastructure: Redis is used as the database, and the application is deployed on Vercel, a well-known cloud platform, so it can withstand growing user loads without compromising stability and performance.   
This research study investigates the technologies used in this chat application's development process, focusing on their features. It will also go into the design decisions taken to produce a communication platform that is effective and easy to use.

# Research Methodology

Redis is the database.   
Redis is selected to ensure efficient message storage and retrieval in the chat application due to its high concurrency support and real-time data handling.   
  
WebSockets is the communication protocol.   
WebSockets enable instantaneous message delivery without the expense of repeated HTTP requests by enabling bidirectional, real-time communication between clients and the server.   
  
Verification: NextAuth and Google OAuth   
In order to provide a seamless user experience, Google OAuth and NextAuth simplify the implementation of authentication flows, offer secure sign-in alternatives, and streamline user authentication.   
  
Pusher in Real-Time Communication   
The instantaneous message transmission provided by Pusher's WebSocket infrastructure improves the chat application's responsiveness and enables scalability for an expanding user base.

## **Methodolodgy**

The methodology adopted for this research paper encompassed several key phases. Initially, a thorough requirement analysis was conducted to delineate stakeholder needs and user requirements. Then, depending on industry norms and compatibility, the right technologies were carefully chosen. These technologies included Redis, WebSockets, Google OAuth, Next.js, Tailwind CSS, Pusher, and Vercel. The creation of the database schema, real-time communication channels, API endpoints, authentication routines, and system architecture came next. The system design was followed in the implementation of the application, and functionality, performance, and security were checked through iterative testing. Comprehensive testing was carried out, encompassing unit, integration, end-to-end, and usability testing. To ensure compliance and peak performance, a security audit and performance evaluation under various scenarios were conducted. A thorough record of the development process was created, which included information on the system design, implementation specifics, testing procedures, and user input. Lastly, research paper dissemination of the results included possible publication in scholarly publications and presentations at conferences.

# Work Status

**January:**

1. Exploration of effective strategies for making and designing the application.

2. Quest for relevant features of the app like the landing page and the frontend and backend requirements.

3. Learn NextJs and the relevant technologies required for the projects like Node.js, Redis db.

4. Learned Routing in NextJs, Design Frontend using NextJs and Tailwind.

5. Made the login and Sign up page for the users.

**February:**

1. Learned about NextAuth Authentication and integration of the concepts of authorization etc.

2. Integration of Redis DB

3. Integration of Authentication.

4. Testing Product till now.

5. Learn WEB Sockets for fast communication.

**March:**

1. Integrate web sockets into the project.

2. Made the page for communication and adding of friend as well.

3. Test the product till this stage.

4. Tested the authentication of the user

**5.** Made the backend of the project using Node and Express in the backend.  
6. Integrated the main technologies for backend like middleware etc.

**April:**

1. Integrated and tested the backend for the project

2. Made Api for the user and for sending request and getting request .

3. Designing the application routes using Nextjs and making sure that hitting the routes is working.

4. Documented the project till this stage.

5. Designing the environment variables needed for the project.

**May:**

1. Finalizing the project and making sure that the project is working in my environment.

2. Uploading the project to Github.

3. Building the project to host the application.

4. Host the application on Vercel

5. Test the application working and maintaining the site.