JOURNAL OF BASIC SCIENCE AND ENGINEERING

BUILDING A REAL-TIME WEB-BASED CHAT APPLICATION WITH NEXT.JS, MONGODB, AND PRISMA ORM: ENHANCING USER EXPERIENCE THROUGH OPTIMIZED DECISION-MAKING

Purvesh Paunikar [1], Dhanashri Londhe[2], Sanket Pawar[3], Rohan Phulari[4], Sushant Dondge[5], Ramesh Chauhan[6]

Department of Computer Engineering, Shree Ramchandra College of Engineering, Pune University, India *[1]purveshpaunikar@gmail.com, [2]londhe.dhanashri@gmail.com., [3]sanketpawar7112@gmail.com ., [4]rohanphulari1239@gmail.com., [5]sushantdhondge1@gmail.com., [6]rc006400@gmail.com.*

Abstract— *Chat apps have become indispensable in our connected world. It meets instant communication needs and allows individuals or groups to exchange messages instantly. These applications serve many purposes, from personal communication to collaboration, customer support, and team collaboration. What they provide facilitates quick decision-making, improves connectivity, and facilitates the efficient exchange of information. Build interactive using modern technologies such as Next.js, Socket.IO, Prisma, Tailwind CSS, TypeScript and MongoDB. The application is designed and used to provide communication, response and interaction to users. Next.js is React's framework of choice due to its server-side rendering capabilities, boosting page performance, and improving SEO. The Socket.IO JavaScript library is integrated to facilitate two-way communication between client and server and enable fast messaging. Prisma is a modern repository for efficient use and safe mode, thus improving control and integrity. Tailwind CSS is used to create a clean, flexible and flexible user interface, simplifying the development process compared to its previous utility. TypeScript is a statically typed superset of JavaScript used to improve code quality and control and catch errors during development. MongoDB is a preferred NoSQL database for storing and storing conversations due to its simplicity and scalability. This article discusses the application's design, terms of use and operational aspects, and demonstrates its functionality and usability. This project demonstrates the power and diversity of connectivity technologies to create a modern and effective communication system.*

Keyword: Communication, Socket.io, Next.js, Real Time Prisma, MongoDB

INTRODUCTION

In today's world, chat applications are very useful. With the help of chat applications, people can send each other messages and also share files, images, etc. So, we built a chat application using the newest technologies available on the market, like Nextjs, which is a React Framework, Prisma ORM, Socket.io, Typescript, etc. This chat built with Next.js, Socket.io, MongoDB,

Prisma, Tailwind CSS, TypeScript, and Flask represents a powerful combination of technologies for creating a robust and feature-rich communication platform.

Next.js provides a solid foundation for building React-based web applications, ensuring efficient rendering and seamless client-side routing. Socket.io enables real-time bidirectional communication, making it ideal for instant messaging functionalities. MongoDB serves as a flexible and scalable NoSQL database, while Prisma streamlines database access with its ORM capabilities. Tailwind CSS offers a utility-first approach to styling, facilitating rapid UI development and customization.

TypeScript enhances code quality and maintainability with its strong typing system and advanced tooling support. Finally, Flask provides a lightweight and extensible backend framework in Python, enabling the implementation of RESTful APIs, user authentication, and other server-side functionalities. Together, these technologies empower developers to create a modern and responsive chat application with robust features and a seamless user experience

Next.js is a powerful React framework designed to simplify the development of web applications, including chat applications. Its key features make it particularly useful for building chat applications. Next.js allows rendering React components on the server side, which improves initial load times and SEO optimization. For a chat application, this means faster content delivery and better search engine visibility. Next.js can integrate seamlessly with technologies like Socket.io for real-time communication.

This is crucial for chat applications as it enables instant messaging and live updates without page refreshes. Next.js automatically splits code into smaller bundles, optimizing performance by loading only what's necessary. In a chat application, this helps manage the complexity of handling multiple chat rooms or conversations efficiently. Next.js offers efficient client-side routing, making navigation within the chat application smooth and responsive. Users can switch between different chat rooms or conversations without reloading the entire page. Next.js provides excellent support for TypeScript, ensuring type safety and better code organization in large-scale chat applications

MongoDB with Prisma ORM can be highly beneficial for building a chat application due to several key advantages like MongoDB offers a flexible and scalable document-based data model. In a chat application, this flexibility allows you to store messages, user profiles, chat rooms, and other related data in a format that suits your application's needs, without rigid schema constraints. It also supports for real-time updates through technologies like Change Streams or WebSocket integrations enables instant message delivery and updates in a chat application. This real-time functionality is crucial for providing a seamless chatting experience to users. Prisma simplifies database access and management by providing a type-safe and auto-generated database client. It supports MongoDB along with other databases, making it easy to perform CRUD operations, handle complex queries, and manage database schemas efficiently. Prisma ORM helps maintain data integrity by enforcing type safety and providing built-in validation

mechanisms. MongoDB's security features, such as authentication, authorization, and encryption, further enhance data security in your chat application.

Proposed System

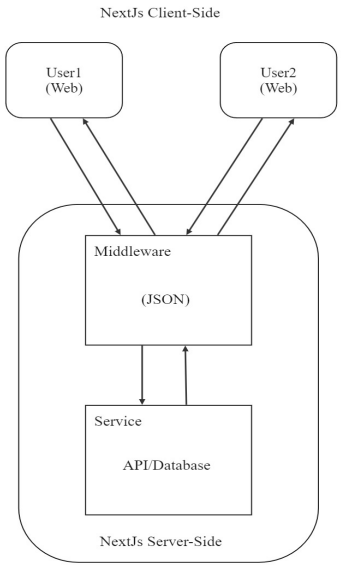


Figure1.1: Client-Server Side Block Diagram

Figure 1.1 explains how the client and server work. On the client side we have the user or we can say the user interface through which the user can access messages and other content, and on the server side we have the middleware and services. The client will communicate via JSON, which acts as an intermediary for the server. JSON stands for JavaScript Object Notation, a deep data interchange format that is readable and writable by humans and easy for computers to interpret and create. The service has three services. NextJs is used for server-side server operations, Socket.io is used for instant messaging, and MongoDB is used for data storage.

METHODOLOGY

User Authentication: Users need to be authenticated to use the chat system. This involves user registration, login, and management of user credentials. This module typically includes functionality for users to create accounts within the chat system. Users provide necessary information such as username, email, and password.

Contact Module: The contact module of the chat application serves as a vital component for managing user contacts and connections. It includes functionalities such as displaying a user's contact list with details like names, profile pictures, and statuses, allowing the addition of new contacts through search by usernames or email addresses, as well as providing the option to delete or remove contacts.

Messaging Module: The chat module is at the core of the chat application, serving as the primary platform for communication between users. It encompasses several essential functionalities, beginning with one-on-one messaging capabilities that enable users to engage in private conversations. This feature includes indicators for message status, like sent, delivered, and read statuses, ensuring a clear understanding of message progression.

Chatbot Module: The chatbot module provides great assistance. It is an automated system designed to simulate conversations with users, providing real-time responses and assistance. It can greatly enhance user interaction and functionality. The user can ask any question to the chatbot, and the chatbot will answer that question.

Aim: The aim of this research is to develop a robust and efficient real-time chat application using a combination of Next.js, Prisma, MongoDB, Socket.io, TypeScript, Tailwind CSS, and Flask. This application aims to provide users with a seamless chatting experience while demonstrating the integration of various modern technologies for web development.

Objective:

- The objective of this research is to design, implement, and evaluate a feature-rich chat application that leverages Next.js, Prisma, MongoDB, Socket.io, TypeScript, Tailwind CSS, and Flask.
- The specific objectives include developing user authentication, real-time messaging functionalities, data storage and retrieval using Prisma and MongoDB, responsive UI design with Tailwind CSS, and efficient server-side handling with Flask.
- The research aims to demonstrate the capabilities and performance of integrating these technologies for building modern web applications

Problem Statement

With the increasing demand for real-time communication platforms, there is a need for robust and scalable chat applications that can handle concurrent users efficiently while ensuring data security and seamless user experience. Existing chat solutions often lack integration with modern web development technologies, leading to performance bottlenecks and limited customization options. This research addresses these challenges by developing a comprehensive chat application using Next.js, Prisma, MongoDB, Socket.io, TypeScript, Tailwind CSS, and Flask, aiming to provide a solution that combines functionality, performance, and flexibility in a single platform.

Scope of the study

The scope of this study encompasses the design, implementation, and evaluation of a real-time chat application using Next.js, Prisma, MongoDB, Socket.io, TypeScript, Tailwind CSS, and Flask. The study will cover aspects such as user authentication, real-time messaging features, data storage and retrieval, responsive UI design, and server-side handling. Performance evaluation, scalability testing, and user experience assessment will also be included in the scope

to measure the effectiveness and efficiency of the developed chat application. The study will not delve into advanced topics such as machine learning integration or complex data analytics, focusing primarily on the core functionalities and technical aspects of the chat application.

Motivation: The motivation behind this research stems from the increasing demand for real-time communication solutions in today's digital age. Traditional chat applications often face limitations in terms of scalability, performance, and customization. By leveraging the capabilities of Next.js, Prisma, MongoDB, Socket.io, TypeScript, Tailwind CSS, and Flask, this research aims to address these limitations and deliver a modern, feature-rich chat application. The motivation also lies in exploring the integration possibilities of these technologies to create a seamless and efficient chatting experience for users, while also contributing to the advancement of web development practices

FLOW CHART

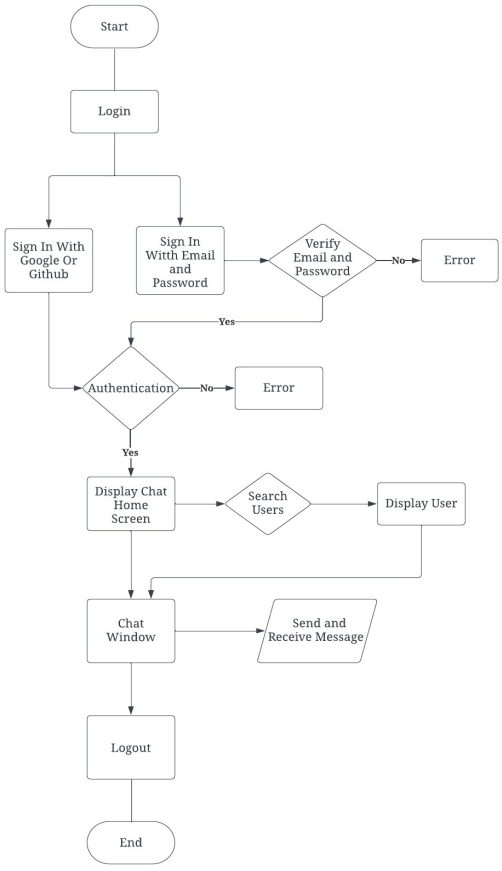


Figure1.2: Flow chart of chat system

Figure 1.2 shows the flow of chat application or how the chat system will work.

1. In first phase user have to sign in if the user is new or doesn't have account user can create account using Google and GitHub or by manually providing information and if user has account, then user can directly login to the webapp by putting correct credential.

2. After successfully login to app if user has friends or chat group, he/she can start conversation with them if user is new to application and doesn't have friends then user can send invite link to friends so that they can join and can start conversation.

3 If users click on the logout button, they will be logged out of the application and sent to the homepage, or you can say in the sign-in/signup page.

Technologies/Libraries Used

Next.js is a React framework that streamlines modern web app development, offering server-side rendering, static site generation, and file-based routing. It enhances performance and SEO by rendering React components on the server side and generating static HTML files. Next.js supports TypeScript natively, providing static typing for projects. It also optimizes builds with features like automatic code splitting and dynamic importing.

Socket.IO enables real-time, bidirectional communication between web clients and servers, crucial for applications like chat platforms and multiplayer games. It establishes persistent connections using WebSocket's and includes fallback mechanisms for environments that don't support them. Socket.IO's event-based architecture facilitates custom event handling and message broadcasting across multiple clients.

TypeScript is a strongly typed superset of JavaScript, enhancing code robustness and maintainability by catching type-related errors during compile time. It introduces features like type annotations, interfaces, Enums, and generics, allowing developers to define data shapes and identify bugs early in development.

One of the key benefits of TypeScript is its ability to enhance code readability and documentation through explicit type declarations. This makes it easier for developers to understand the expected types of variables, function parameters, and return values, leading to improved code quality and reduced debugging time.

Additionally, TypeScript integrates well with modern development tools and frameworks, including React, Angular, and Node.js, providing a seamless development experience for building scalable and reliable applications. It also supports features like code refactoring, intelligent code completion, and static analysis, enhancing productivity and code maintainability.

Overall, TypeScript is a powerful tool for building complex web applications that require type safety, scalability, and maintainability, making it a popular choice among developers in the JavaScript ecosystem.

TailwindCSS: Tailwind CSS is a utility-first CSS framework that provides a set of pre-designed utility classes to quickly style web interfaces. Unlike traditional CSS frameworks that come with predefined components and styles, Tailwind CSS focuses on providing low-level utility classes that can be combined to create custom designs without writing custom CSS code.

The core concept of Tailwind CSS revolves around utility classes like bg-blue-500 for setting background color, text-xl for text size, p-4 for padding, and so on. These utility classes follow a naming convention that reflects their purpose and can be easily applied to HTML elements directly in the markup.

One of the main advantages of Tailwind CSS is its flexibility and scalability. Developers can easily create responsive layouts, customize designs, and maintain consistency across the application by leveraging the utility classes provided by Tailwind CSS. Additionally, Tailwind CSS encourages a rapid development workflow by eliminating the need to write custom CSS code for common styling tasks.

Tailwind CSS also integrates well with other front-end frameworks and tools like React, Vue.js, and PostCSS, allowing developers to use Tailwind CSS alongside their preferred stack. Its modular architecture and theming capabilities further enhance its versatility and suitability for building modern web applications.

Overall, Tailwind CSS is a popular choice for developers who prefer a utility-first approach to styling, enabling them to create visually appealing and responsive user interfaces efficiently.

MongoDB: MongoDB is a popular NoSQL database management system that is designed for scalability, flexibility, and ease of development. Unlike traditional relational databases, MongoDB uses a document-oriented data model, storing data in JSON-like documents with dynamic schemas. This allows developers to store and retrieve data in a more natural and flexible way, making it suitable for a wide range of use cases, including web applications, content management systems, and real-time analytics platforms.

One of the key features of MongoDB is its ability to handle large volumes of unstructured or semi-structured data efficiently. It supports horizontal scaling through sharing, allowing data to be distributed across multiple servers to handle increased load and storage requirements. This makes MongoDB well-suited for applications that require high availability, performance, and scalability.

MongoDB also provides a rich query language and aggregation framework, enabling developers to perform complex queries, aggregations, and data manipulations directly within the database. It supports various indexing options, including compound indexes, text search, and geospatial indexes, to optimize query performance and data retrieval.

Additionally, MongoDB offers built-in replication and automatic failover mechanisms to ensure data durability and availability in case of hardware failures or network issues. It also integrates seamlessly with popular programming languages and frameworks, providing official drivers and libraries for languages like JavaScript, Python, Java, and Node.js.

Overall, MongoDB is a versatile and powerful database solution that offers flexibility, scalability, and developer-friendly features, making it a preferred choice for modern application development

Prisma: Prisma is an open-source modern database toolkit that provides an ORM (Object-Relational Mapping) for TypeScript and JavaScript applications. It simplifies database access and management by allowing developers to interact with databases using a type-safe and auto-generated query builder. Prisma supports multiple databases, including PostgreSQL, MySQL, SQLite, and MongoDB, and provides features like schema migrations, data modeling, and performance optimizations. It is widely used in web development for building scalable and efficient backend systems with a focus on developer productivity and type safety.

Flask: Flask is a lightweight and versatile web framework for Python that is designed to make web development simple and efficient. It is known for its minimalistic approach, allowing developers to build web applications quickly without imposing rigid patterns or dependencies. Flask provides the essential tools and libraries needed for web development, making it a popular choice for building small to medium-sized web applications, APIs, and prototypes.

With the help of Flask we have hosted internal Api endpoints.

Overall, Flask is an excellent choice for developers who prefer a lightweight and flexible web framework for building Python-based web applications. Its simplicity, extensibility, and strong community support make it a popular framework in the Python ecosystem.

Decision-Making: We have implemented a decision-making module in python which decide what action we will need to take in order to give response to user in chatbot, Python employs short-circuit logic in boolean expressions, which allows the evaluation to stop as soon as the outcome is determined. This feature is crucial for writing efficient conditions and preventing unnecessary computation.

V. RESULT

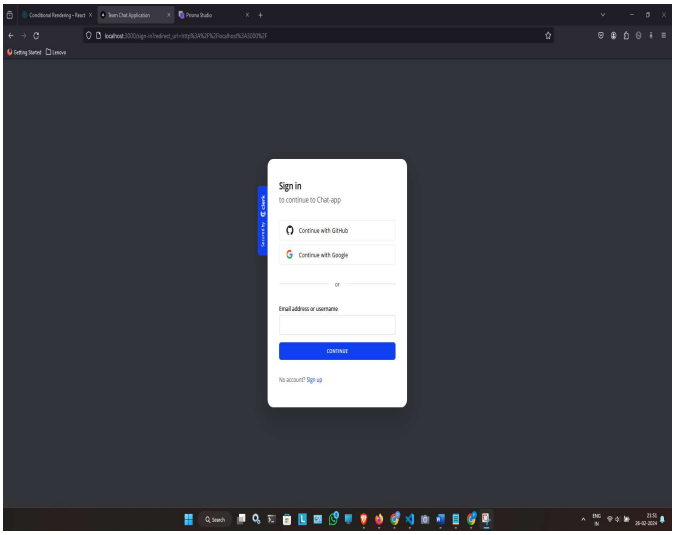


Figure1.3: Authentication Page (Signing / Signup)

The authentication page in Figure 1.3 allows users to sign in or sign up using their Google or GitHub account or enter their email address and password. After successfully signing in or signing up, the name and photo associated with their Google Account will appear on their profile.

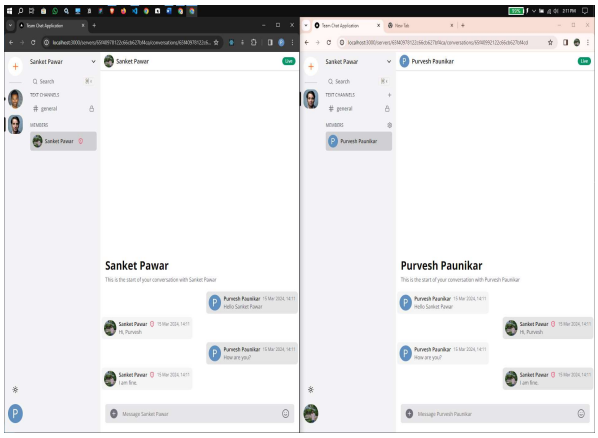


Figure 1.4: Messaging page/Chat page

In Figure 4, a message or discussion page allows users to send messages to other users. There is an input field where you can type a message, and if you press Enter after typing the message, the message will be sent to another user. There is a dark and light mode button where we can change the theme of the web chat application, there is also a button.

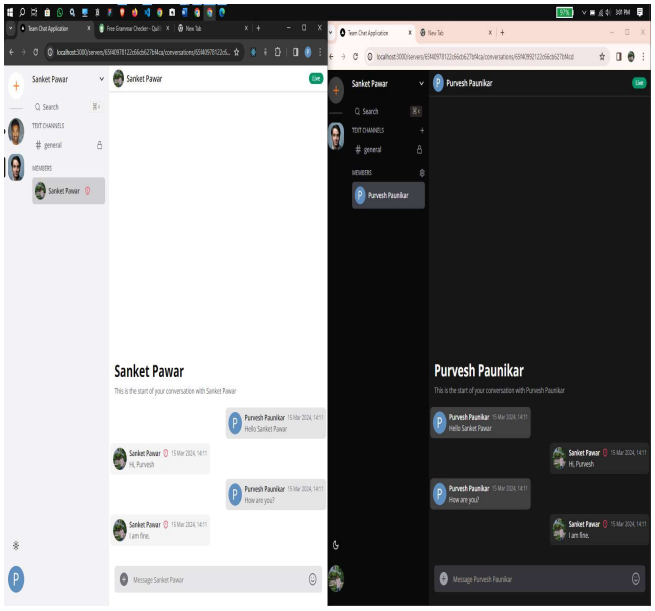


Figure1.5: Dark-Light Mode

In Figure 1.5, the user can change the theme according to his or her preference. If the user wants a dark theme, he or she can change it to dark, or if he or she prefers a light theme, he or she can change it to light theme. By default, the theme will be set according to the system theme.

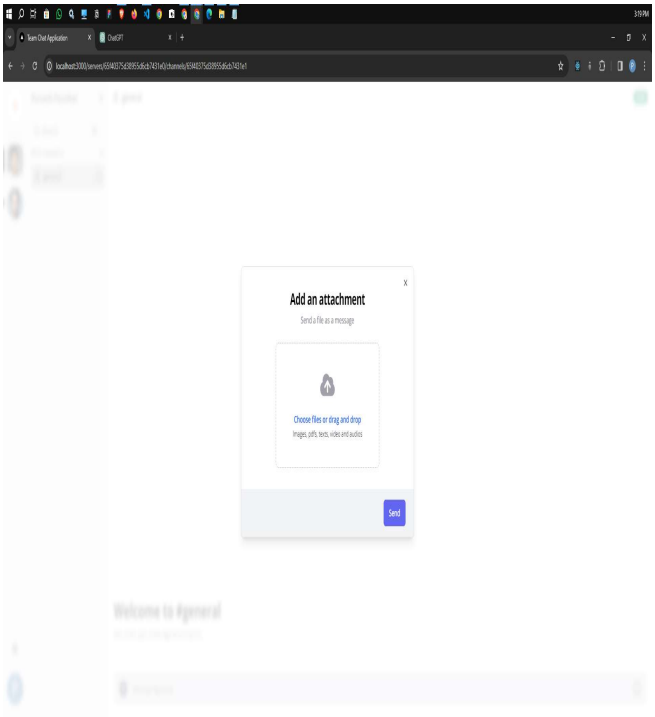


Figure1.6: File/Images Share Menu

In Figure 1.6, we can see that we can share files like images, documents (pdf, docx, ppt, etc.). You can select from a folder or you can drag and drop files.

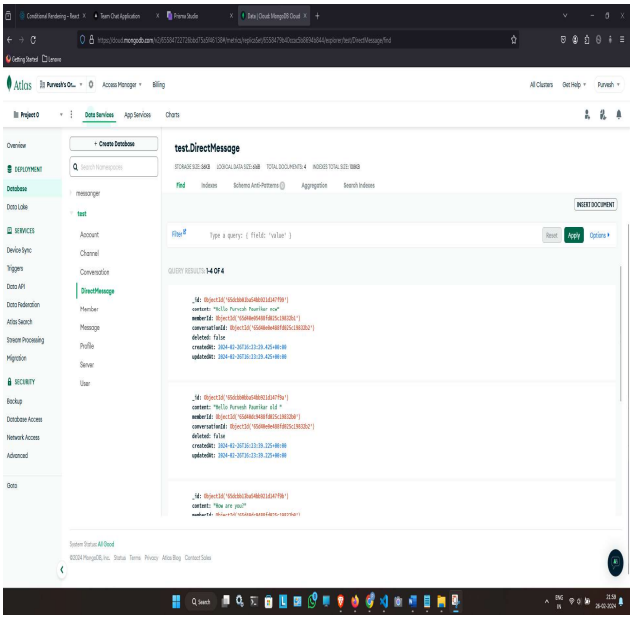


Figure1.7: Data Store in MongoDB database

In Database Figure 1.7, we can see that data such as user data, messages, and other data are stored in the MongoDB database

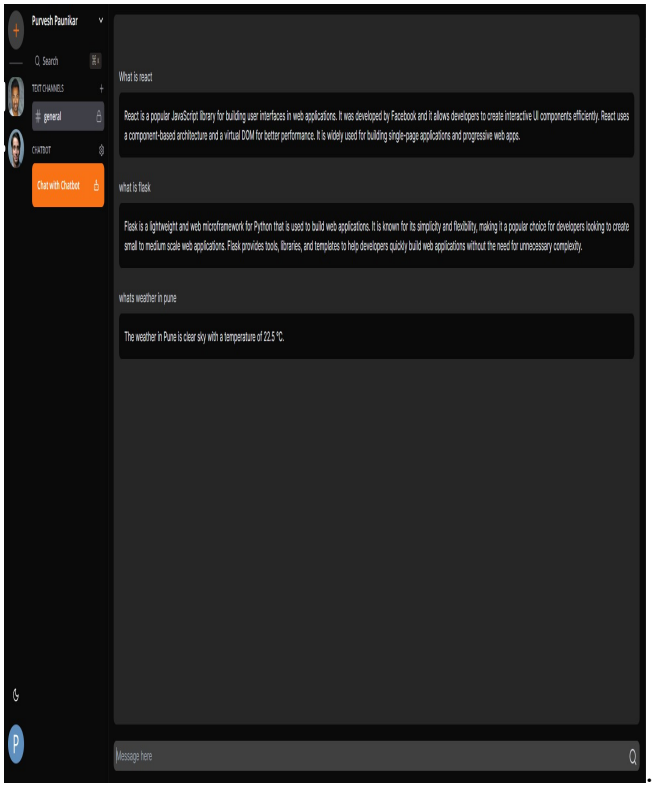


Figure1.8: Chatbot

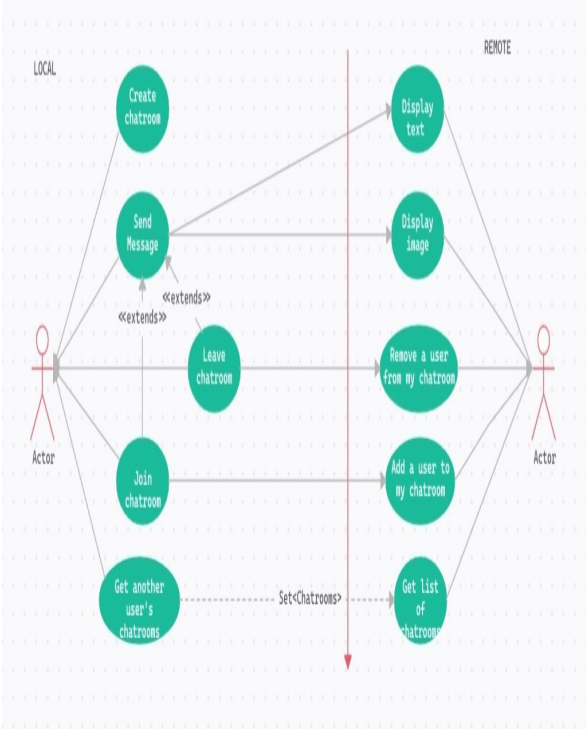
Here, we can see that we can chat with chatbot and he is able to respond us about any questions

And about live weather information and we can able to ask him also about movie information

UML DIAGRAMS

Use Case Diagram

Figure1.9: Use Case Diagram



Sequence Diagram

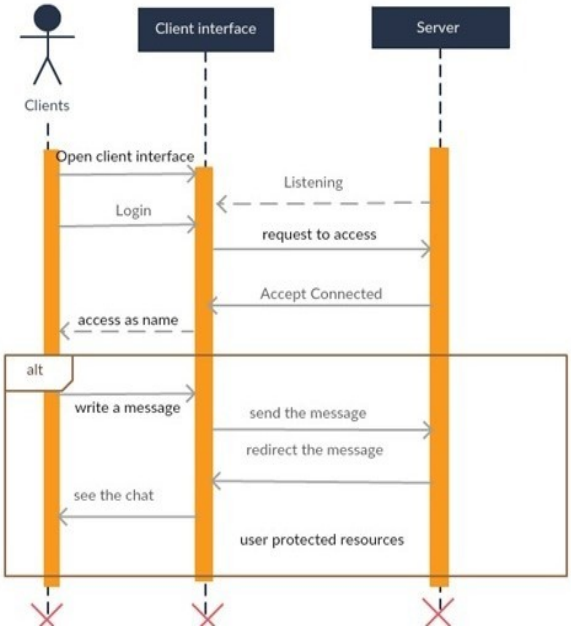


Figure1.10: Sequence Diagram

DFD Diagram

Zero Level

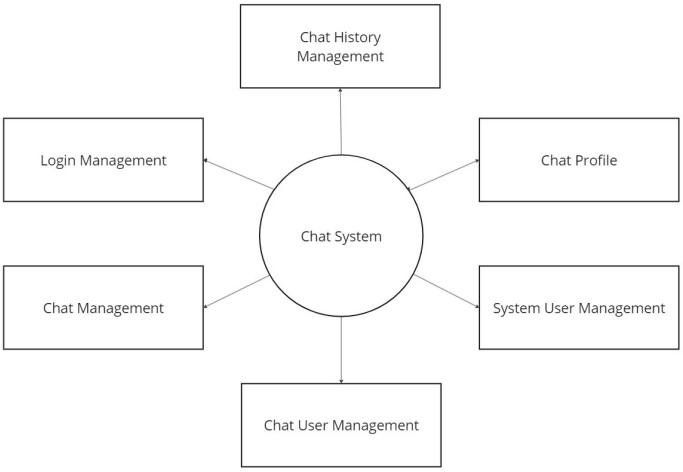


Figure1.11: Data flow Diagram level-0

First Level

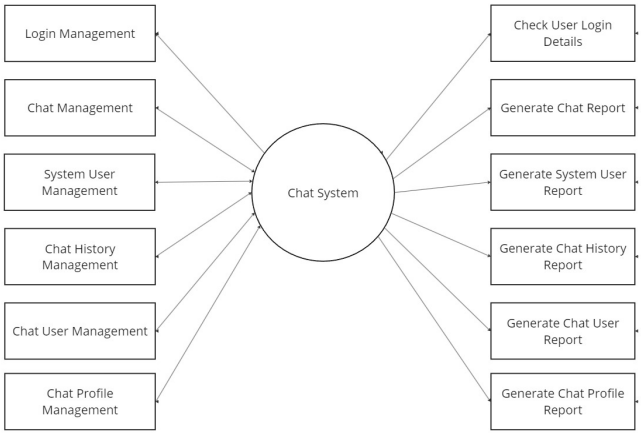


Figure1.12: Data flow Diagram level-1

Second Level

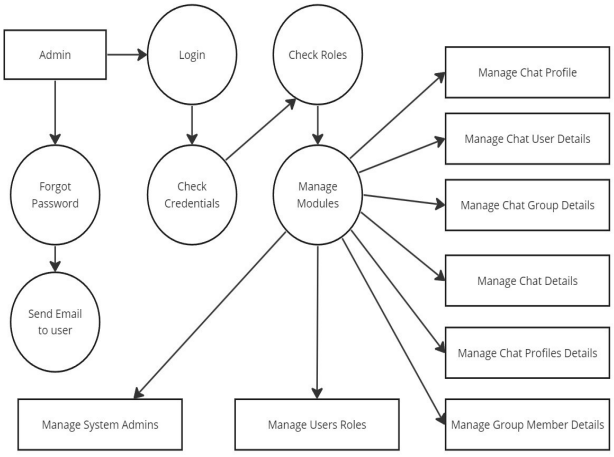


Figure1.13: Data flow Diagram level-2

Class Diagram

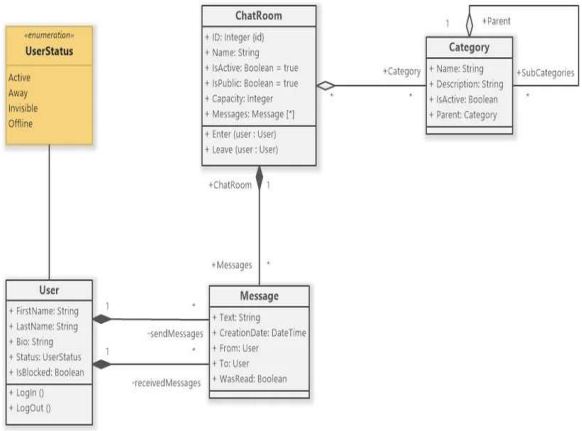


Figure1.14: Class Diagram

Activity Diagram

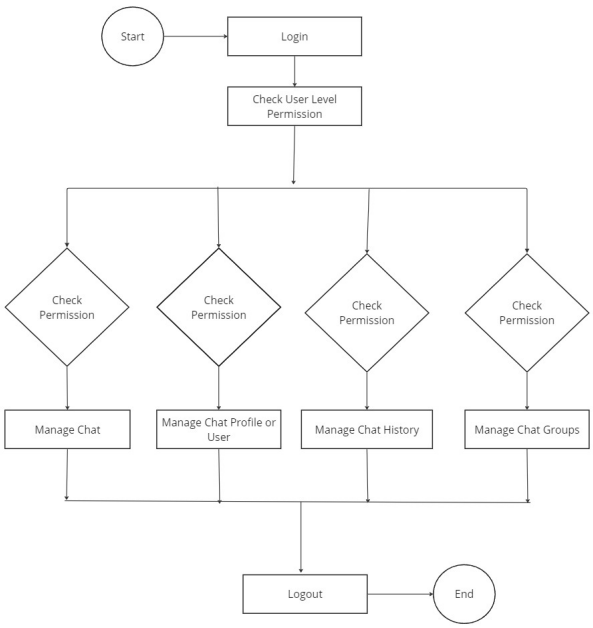


Figure1.15: Activity Diagram

ADVANTAGES

1. Real-time Communication: Socket.IO is excellent for real-time, bidirectional communication. Users can send and receive messages instantly, creating a responsive and engaging chat experience.

2. Server-Side Rendering (SSR) with Next.js: Next.js allows for server-side rendering, which improves SEO, performance, and initial load times. This means your chat application will be more discoverable and user-friendly.

3. Data Modeling and Querying with Prisma: Prisma simplifies database access with its powerful ORM. You can define your data models in a type-safe way using Prisma's schema, and easily perform CRUD operations and complex queries. This leads to faster development and fewer errors.

DISADVANTAES

Combining multiple technologies introduces complexity, especially for developers who are not experienced with all of them. Integrating Next.js, Socket.IO, Tailwind CSS, Prisma, and MongoDB requires careful planning and

While Next.js provides server-side rendering, adding Socket.IO for real-time communication can introduce additional overhead. Real-time applications require continuous connections, which can impact server performance, especially with a large number of concurrent users.

With the combination of server-side rendering, real-time communication, and complex data flow, debugging can become more challenging. Identifying issues across the frontend, backend, and database layers may require specialized knowledge.

With MongoDB and Prisma, evolving your data schema can be challenging, especially as your application grows. Changes to the schema may require data migration scripts and careful handling to avoid data loss or inconsistencies.

CONCLUSION

In conclusion, the important role of chat apps in modern communication cannot be overstated. These platforms meet the critical need for instant communication in a personal, professional and collaborative manner. Create responsive interactive applications by using Next.js with WebSockets and Socket.IO to enable two-way communication between client and server. This integration allows users to participate in real-time conversations, exchange information, and collaborate without the limitations of the request-response process. Next.js and WebSocket technology enables instant communication while also improving the confidentiality and integrity of information exchange.

REFERENCES

1. Patel, Vishal. "Analyzing the Impact of Next. JS on Site Performance and SEO." International Journal of Computer Applications Technology and Research 12 (2023): 24-27
2. Sir Nishant Reddy, Aaron A Thomas “Web-based Application for Real-Time Chatting using Firebase”, published in IEEE in 2023.
3. Rozen Berg D*, Tharunraj M, Raj Kumar B, M.R. SumalathaLakshmi Harika Palivela, Vijay Vikrama Karthikeyaa P, 2022,” WebRTC-based Decentralized Chat Application with Minimal Latency “, published in IEEE in 2022.
4. Diotra Henriyan, Devie Pratama Subiyanti, Rizki Fauzian, Dian Anggraini, M. Vicky, 2016, “Design and Implementation of Web Based Real Time Chat Interfacing Server”, published in IEEE in 2016.
5. Yinka-Banjo, Chika & Esther, Ogundeyi. (2019). Financial stock application using websocket in real time application. International Journal of Informatics and Communication Technology (IJ-ICT). 8. 139. 10.11591/ijict. v8i3.pp139-151.
6. Cadenhead, Tyson. Socket. IO Cookbook. Packt Publishing Ltd, 2015.
7. Purnomosidi, B. (2013). Pengembangan Aplikasi Cloud Computing Menggunakan Node.js.