FriendZ ChaT

A Modern Chat System

A Project Report

Submitted by: -

SUDIP CHANDRA DAS	**BWU/MCA/23/072**
BISHAL RAY	**BWU/MCA/23/073**
ANKITA MITRA	**BWU/MCA/23/093**
KOYEL MUKHERJEE	**BWU/MCA/23/094**
SREYA MONDAL	**BWU/MCA/23/076**

WebGuru InfoSystem

January 2025

INDEX

FRIENDZ CHAT

A Modern Chat System

Description: -

The **FriendZ ChaT** project is a real-time chat application designed to facilitate seamless communication between users. With the growing demand for efficient and interactive messaging platforms, this project demonstrates the integration of modern web technologies to create a scalable, user-friendly, and responsive solution. This project leverages a full-stack architecture that includes a React-based frontend, a Node.js and Express-powered backend, and WebSocket (via Socket.IO) for real-time data exchange.

Technology Used:

- **Frontend:** React, Tailwind CSS, DiasyUI, PostCSS,
- **Backend:** Node.js, Express.js, Socket.IO, BcryptJs
- **Database:** MongoDB with Mongoose for data modelling, Cloudinary.
- **Authentication**: JWT (JSON Web Tokens) for user authentication.

Road Map: -

1. **Project Initialization**
   - Set up the project for both backend and frontend
   - Define the project requirements and goals.
2. **Frontend Development**
   - Set Up Basic UI:
   - Implement State Management:
   - Develop Pages:
     - Login/Signup Page.
     - Home Page (chat dashboard).
     - Profile Page (user settings).
   - Integrate APIs.
3. **Backend Development**
   - Set Up Server.
   - Implement Authentication.
   - Set Up Database.
4. **Testing**
   - **Frontend Testing:**
     - Validate UI responsiveness and functionality.
     - Test API integration for data flow.
   - **Backend Testing:**
     - Test API endpoints using tools like Postman.
     - Verify WebSocket events for real-time messaging.
   - **User Testing:**
     - Simulate user scenarios to ensure a smooth experience.
5. **Deployment**
   - Deploy the frontend using platforms like Netlify or Vercel.
   - Deploy the backend to a server (e.g., AWS, Heroku, or Render).

- Configure environment variables for production (e.g., API keys, database URLs).

6. **Future Enhancements**
   - Add advanced features like:
     - Group chats.
     - File sharing (documents).
     - Push notifications for new messages.
     - Full access on profile section.
     - Add Calling Option.

<u>Features: -</u>

1. **User Management: -**User Authentication, Profile Management (only picture), Session Management.
2. **Chat Features: -**Real-Time Messaging, Message History, Typing Indicators, Message Status.
3. **User Interface: -**Responsive Design, Light and Dark Themes, Clean and Intuitive UI.
4. **Notifications: -**Real-Time Notifications, Unread Message Count, Secure APIs, Data Encryption.
5. **Deployment and Performance: -**Fast Loading, Cross-Browser Compatibility.

<u>Feature Solution: -</u>

- ➢ **User Authentication**: Secure login/signup with JWT for session management.
- ➢ **Real-Time Messaging**: Instant message delivery using WebSocket (Socket.IO).
- ➢ **Message History**: Display previous chats with seamless database integration.
- ➢ **Responsive UI**: Tailwind CSS-based design for desktop and mobile support.
- ➢ **Themes**: Themes are toggling from the settings.
- ➢ **Notifications**: Real-time and unread message count for active chats.

<u>Test Cases:</u>

- ➢ **Login Test**: Verify login with valid/invalid credentials.
- ➢ **Signup Test**: Test user registration with valid/invalid input.
- ➢ **Real-Time Messaging**: Check if messages are delivered instantly.
- ➢ **Responsive Design**: Test layout on desktop, tablet, and mobile devices.
- ➢ **Theme Switching**: Check seamless toggling between various themes.

<u>Project Review: -</u>

The **FriendZ ChaT** project is a robust real-time chat application, integrating modern technologies to deliver an efficient and user-friendly experience. Its architecture, built with React, Node.js, and Socket.IO, ensures seamless communication and scalability. Key features such as real-time messaging, authentication, and a responsive UI provide a strong foundation for practical use and future expansion.

<u>Conclusion: -</u>

The FriendZ ChaT project successfully demonstrates the integration of modern web technologies to create a real-time, scalable, and user-friendly chat application. By combining a React-based frontend, a Node.js/Express backend, and WebSocket (via Socket.IO) for real-time communication, it offers an efficient platform for seamless messaging.

The project excels in delivering essential features such as user authentication, responsive design, and real-time messaging. This project is not only a practical communication tool but also serves as an excellent learning resource for developers exploring full-stack development. It highlights best practices in frontend development, backend API design, state management, and database integration.

References: -

Online Documentation

Backend: -

1. **Bcryptjs -** https://www.npmjs.com/package/bcrypt
2. **Cloudinary -** https://cloudinary.com/documentation/react_integration
3. **Cookie-parser -** https://www.npmjs.com/package/cookie-parser
4. **Cors -** https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
5. **Dotenv -** https://www.npmjs.com/package/dotenv
6. **Express -** https://www.npmjs.com/package/express
7. **Jsonwebtoken -** https://www.npmjs.com/package/jsonwebtoken
8. **Mongoose** - https://www.npmjs.com/package/mongoose
9. **socket.io -** https://socket.io/docs/v4/

Devdependency:

1. **Nodemon -** https://www.npmjs.com/package/nodemon

Frontend: -

1. **Axios -** https://axios-http.com/docs/intro
2. **Lucide-react -** https://lucide.dev/guide/packages/lucide-react
3. **React -** https://react.dev/blog/2023/03/16/introducing-react-dev
4. **React-dom -** https://legacy.reactjs.org/docs/react-dom.html
5. **React-hot-toast -** https://react-hot-toast.com/docs
6. **React-router-dom -** https://reactrouter.com/home
7. **socket.io-client -** https://socket.io/docs/v4/client-api/
8. **Zustand -** https://zustand.docs.pmnd.rs/getting-started/introduction

Devdependency:

1. **eslint/js -** https://eslint.org/docs/latest/
2. **vitejs/plugin-react -** https://www.npmjs.com/package/@vitejs/plugin-react
3. **Autoprefixer -** https://www.npmjs.com/package/autoprefixer
4. **Daisyui -** https://daisyui.com/docs/install/
5. **Eslint-plugin-react-refresh -** https://www.npmjs.com/package/eslint-plugin-react-refresh
6. **Globals -** https://www.npmjs.com/package/globals-docs
7. **Postcss -** https://postcss.org/docs/
8. **Tailwindcss -** https://v2.tailwindcss.com/docs
9. **Vite -** https://vite.dev/guide/why.html

Video Resources

1. **Traverse Media**: Full-Stack Web Development Guides on YouTube.
2. **Code With Harry:** React Js Tutorials in Hindi
3. **Sheryians Coding School:** ReactJs Crash Course: Master the Basics in One Video!