ShadowChat Project Report

1. Introduction

ShadowChat is an enhanced real-time messaging application designed for secure and seamless communication with advanced administrative capabilities, collaborative coding and smart reply features. The project provides users with an intuitive platform for instant messaging while ensuring privacy and security, expanded with powerful administrative tools and real-time code collaboration capabilities.

2. Features and Detailed Implementation

2.1 Real-time Messaging using WebSockets

How WebSockets Work: WebSockets provide full-duplex communication channels over a single TCP connection. Unlike traditional HTTP requests, which require repeated polling to fetch updates, WebSockets maintain a persistent connection between the client and the server. This enables real-time message delivery with minimal latency.

Implementation in ShadowChat:

- **Backend:** Socket.io is integrated with the Node.js server to manage real-time communication for both messaging and code collaboration
- **Frontend:** The client listens for incoming messages/code room code changes and updates the chat/code room UI dynamically

Steps:

1. The server initializes a WebSocket connection using Socket.io
2. When a user sends a message, it is emitted to the recipient's socket ID
3. The frontend updates the chat window without requiring a page refresh
4. Code room events synchronize code changes across connected users

2.2 Role-Based Access Control (Implemented in V2.0)

Admin and User Roles: ShadowChat introduces a comprehensive role-based access system that differentiates between regular users and administrators, providing enhanced control and monitoring capabilities.

Admin Capabilities:

- View total number of registered users
- Monitor online users
- Track daily message statistics
- Access complete user list with management options

- Delete user accounts
- Promote users to admin status

Implementation:

- **Middleware Protection:** protectRoute and isAdmin middleware ensure secure access to admin features
- **Database Schema:** User model includes role field to distinguish between 'user' and 'admin' roles
- **API Routes:** Dedicated admin endpoints for user management and analytics

Admin API Routes:

```
router.get("/stats", protectRoute, isAdmin, getStats);

router.get("/perdaymsgs", protectRoute, isAdmin, getmessagesperday);

router.get("/allusers", protectRoute, isAdmin, getAllUsers);

router.delete("/deleteuser/:id", protectRoute, isAdmin, deleteUser);

router.put("/promoteuser/:id", protectRoute, isAdmin, promoteUser);
```

2.3 Code Room - Collaborative Coding Feature (Implemented in V2.0)

Real-time Code Collaboration: The Code Room feature enables multiple users to collaborate on code in real-time, similar to popular collaborative coding platforms.

How Code Room Works:

- Users join a shared coding environment using a unique room ID
- Real-time synchronization of code changes across all participants
- Live user presence indication showing who's currently in the room
- Monaco Editor provides a professional coding experience with syntax highlighting

Implementation:

- **Monaco Editor Integration:** Professional code editor with IntelliSense and syntax highlighting
- **Socket.io Room Management:** Users join specific rooms for isolated coding sessions
- **Real-time Synchronization:** Code changes are immediately broadcast to all room participants
- **User Presence:** Live tracking of users currently active in each code room

Code Room Features:

- Multi-language syntax support

- Real-time collaborative editing
- User presence indicators
- Room-based isolation
- Professional IDE-like experience

2.4 Smart Replies - AI-Powered Response Suggestions (Implemented in V2.0)

AI-Enhanced Communication: The Smart Replies feature leverages artificial intelligence to provide users with contextually relevant response suggestions, enhancing communication efficiency and reducing typing effort.

How Smart Replies Work:

- Analyzes the last received message in the conversation
- Generates three intelligent, human-like response options using Google's Gemini API
- Presents quick-reply buttons for seamless message composition
- Maintains conversational flow while reducing response time

Implementation in ShadowChat:

- AI Integration: Gemini API integration processes incoming messages and generates contextually appropriate replies
- Smart Prompt Engineering: Uses optimized prompt structure to ensure relevant and natural-sounding suggestions
- Real-time Processing: Smart replies are generated instantly upon receiving new messages
- User Experience: One-click reply selection streamlines conversation flow

2.5 User-Friendly Interface

A well-designed interface improves user engagement and usability. ShadowChat uses React.js with Tailwind CSS and DaisyUI for a clean and responsive design.

Implementation in ShadowChat:

- **React Components:** Modular components handle chat windows, message input, notifications, admin dashboard and code room interface
- **CSS Optimization:** Tailwind CSS continues to ensure lightweight styling with faster loading times
- **Theme Customization:** DaisyUI provides consistent theming across new features
- **Responsive Design:** Interface adapts seamlessly between messaging, admin panel, and code room views

2.6 State Management with Zustand

Why Zustand: Zustand is a lightweight state management library that simplifies global state handling without the boilerplate of Redux.

Implementation in ShadowChat V2.0:

- **Chat State Management:** Zustand stores active conversations and messages
- **User Authentication:** Maintains user session data and role information
- **Admin State:** Manages user statistics, online status, and administrative data
- **Code Room State:** Handles active code sessions and participant information

2.7 Online Status and Enhanced User Tracking

Improved User Presence: ShadowChat enhances user tracking with more granular presence information for both messaging and code room contexts.

Implementation:

- Socket.io tracks user connections across different contexts (chat, code rooms)
- Enhanced presence indicators in code rooms

3. Enhanced Tech Stack and Justification

3.1 Frontend: React.js, Tailwind CSS, DaisyUI & Monaco Editor

- **React.js:** Component-based architecture supports the expanded feature set
- **Tailwind CSS:** Utility-first approach continues to enable fast, responsive styling
- **DaisyUI:** Provides consistent theming across new admin and code room interfaces
- **Monaco Editor:** Professional code editor providing VS Code-like experience for collaborative coding

3.2 Backend: Node.js, Express.js & Enhanced Socket.io

- **Node.js:** Efficiently handles multiple concurrent connections for messaging and code collaboration
- **Express.js:** Robust routing supports new admin endpoints and code room management
- **Socket.io:** Enhanced with room management for code collaboration and improved real-time features

3.3 Database: MongoDB with Enhanced Schema

- **MongoDB:** NoSQL flexibility supports new user roles and code room data structures

- **Enhanced Collections:** Additional collections for admin analytics and code room sessions

3.4 Security: Enhanced Middleware

- **Role-based Authentication:** protectRoute and isAdmin middleware ensure secure access
- **Data Validation:** Enhanced input validation for admin operations and code room management

4. Installation & Setup

4.1 Prerequisites

- Node.js (v16 or later)
- MongoDB (local or cloud instance)
- Modern web browser with WebSocket support

4.2 Steps

1. **Clone the repository:**
2. git clone https://github.com/Luv-valecha/ShadowChat.git
3. cd ShadowChat
4. **Install dependencies:**
5. npm install
6. **Configure environment variables:**
    - Set up .env file with necessary credentials
    - Include MongoDB connection string
    - Configure Socket.io settings
7. **Start the backend server:**
8. npm run server
9. **Start the frontend:**
10. cd client
11. npm start
12. **Access the application:**
    - Navigate to the local development URL
    - Create admin account for testing administrative features

5. Deployment

ShadowChat maintains deployment compatibility with Render while supporting the enhanced feature set. The hosting solution continues to provide seamless scaling and reliability for real-time communication and collaborative coding.

Deployment Considerations:

- Environment variables for admin features
- Socket.io scaling for code room functionality
- Database optimization for enhanced analytics

6. Future Roadmap

Building on the solid foundation, future enhancements could include:

6.1 Communication Features

- Integration of voice and video calls
- Screen sharing within code rooms
- Multi-device synchronization
- AI-powered message summarization

6.2 Collaboration Enhancements

- Git integration for code rooms
- Project templates and scaffolding
- Code execution and testing environment
- Advanced debugging tools

6.3 Administrative Improvements

- Advanced analytics dashboards
- Automated moderation tools
- Custom role creation
- Audit logging and compliance features

7. Conclusion

ShadowChat represents a significant evolution from a simple messaging application to a comprehensive communication and collaboration platform powered by artificial intelligence. The addition of role-based access control provides administrators with powerful tools for user management and platform monitoring, while the innovative Code Room feature opens new possibilities for real-time collaborative programming. The integration of AI-powered Smart Replies transforms everyday conversations by providing intelligent response suggestions, making communication more efficient and engaging.

The enhanced architecture maintains security and ease of use while introducing sophisticated new capabilities that leverage both real-time collaboration and artificial intelligence. With its expanded tech stack featuring Monaco Editor integration, enhanced Socket.io implementation, and Gemini API integration for smart replies, ShadowChat positions itself as a cutting-edge platform capable of serving both casual communication needs and professional collaborative development workflows.

The platform's modern AI-enhanced architecture ensures scalability and maintainability, making it well-positioned for future enhancements and growing user bases. ShadowChat successfully bridges the gap between messaging applications, collaborative development tools, and intelligent communication systems, creating a unique and valuable user experience that adapts to modern communication expectations.