THINKSYNC: A COLLABORATIVE GROUP CHAT PLATFORM WITH INTEGRATED AI

First Synopsis

Submitted to the Faculty of Engineering and Technology

For the partial fulfilment of the requirements of

Master of Computer Applications

Supervised By:

Rajwinder Kaur

Submitted By:

Vishal (28212301618)

Class: MCA (TYP) 3rd Sem

Master of Computer Applications

Department of Computer Science

Guru Nanak Dev University Amritsar-143005 India February , 2025

Problem Definition and Possible Solutions

Problem Statement

In a rapidly advancing digital world, real-time communication and collaboration are essential for teamwork and project development. Existing chat applications often lack advanced features such as integrated AI support for instant assistance or the ability to directly generate and execute code in a collaborative environment. This results in inefficient workflows, as users have to switch between multiple tools for communication, coding, and AI-driven problem-solving.

Solution

This project aims to develop a **Group Chat Web Application with Integrated AI** that allows users to:

1. Collaborate in real-time through group chats.
2. Leverage AI for answering questions, generating code snippets, and providing on-the-fly assistance.
3. Dynamically create, deploy, and execute Node.js servers directly within the chat interface.
4. End-to-end encryption for chat messages to ensure privacy and strict input validation and output sanitization for code execution to prevent security vulnerabilities.

By integrating communication, AI-driven assistance, and live coding features in a single platform, this application will improve productivity and enhance collaborative problem-solving for developers and other user groups.

Benefits of the Proposed Solution:

1. **Efficiency**: Users can collaborate, code, and deploy in one place, reducing the need for context switching.
2. **Accessibility**: The platform is accessible to both technical and non-technical users, as AI assistance can help bridge knowledge gaps.
3. **Scalability**: The architecture ensures the platform can handle a growing number of users and projects without performance degradation.
4. **Innovation**: This solution fosters creativity and innovation by enabling users to brainstorm ideas, generate code, and instantly deploy solutions.

Introduction to Project and Features

ThinkSync is a cutting-edge group chat platform designed for seamless collaboration, enhanced by AI-powered assistance and live code execution. This platform bridges the gap between communication, coding, and deployment by integrating real-time messaging with AI-driven support and the ability to create and run Node.js servers directly within the chat. ThinkSync aims to streamline workflows for developers, teams, and communities, eliminating the need to switch between multiple tools for communication, coding, and server deployment.

By incorporating advanced AI features, users can generate code, ask questions, receive explanations, and deploy solutions on the fly. ThinkSync not only fosters creativity and innovation but also makes development more accessible and collaborative.

Key Features of ThinkSync

1. **Real-Time Group Chat**
    - Instant messaging with WebSocket technology for real-time communication.
    - Multiple chat rooms for topic-based discussions.
    - Message history storage for easy reference.
2. **Integrated AI Assistant**
    - AI-powered bot for answering questions, generating code snippets, and providing explanations.
    - Supports multiple programming languages.
    - Uses advanced NLP to understand and respond to user queries accurately.
3. **Code Sharing and Execution**
    - Built-in editor to write and execute code snippets within the chat.
    - Real-time code output for immediate feedback.
    - Supports popular languages, focusing on JavaScript and Node.js.
4. **Node.js Server Creation and Deployment**
    - Users can create and deploy Node.js servers directly from the chat.
    - Backend support for spinning up temporary containerized servers.

- Secure and isolated environments for each server instance.

5. **User Authentication and Role Management**
    - Secure login with OAuth or similar protocols.
    - Role-based access control (Admin, Moderator, Member).
    - Ensures secure and controlled access to advanced features.
6. **Scalable and Secure Architecture**
    - Cloud-based hosting for high availability and scalability.
    - End-to-end encryption for secure messaging.
    - Sandboxed code execution to prevent security vulnerabilities.

Frontend and Backend

Frontend (React.js)

The frontend of **ThinkSync** will be built using **React.js**, providing a dynamic and responsive user interface. The focus is on delivering a seamless user experience with real-time chat, AI interaction, and live code execution features.

Key Elements of the Frontend:

- **React Components**: Modular and reusable components for chat interface, AI assistant integration, and code editor.
- **Real-Time Updates**: Powered by **Socket.IO**, ensuring instant message delivery and live feedback.
- **State Management**: Using React's Context API or Redux for efficient state handling across the application.
- **UI/UX Design**: Tailwind CSS or Material-UI for a clean, modern interface.
- **Code Editor Integration**: Embedding a real-time code editor (e.g., Monaco Editor) to allow users to write and execute code within the chat. The frontend will communicate with the backend through RESTful APIs and WebSocket connections for real-time functionality

Backend (Node.js + Express.js)

The backend of **ThinkSync** will be developed using **Node.js** with **Express.js** as the framework, providing a robust and scalable server-side architecture. It will handle real-time communication, user authentication, AI service integration, and Node.js server deployment.

Key Elements of the Backend:

- **Express.js for RESTful APIs**: Facilitates communication between the frontend and the database.
- **Socket.IO for Real-Time Communication**: Enables bidirectional communication for chat and live updates.
- **AI Integration**: Connects to external AI services (e.g., OpenAI API) to process user queries and generate code.
- **Database Management**: **MongoDB** for storing user data, chat history, and configuration of temporary Node.js servers.
- **Authentication and Security**: Implements **JWT** for secure authentication and **CORS** for cross-origin protection.

.

TECHNOLOGIES USED IN PROJECT

Technology Stack (MERN Stack)

1. Frontend – React.js

- Building the user interface for real-time chat and interactive features.
- Component-based architecture for reusability and scalability.
- Integration with third-party libraries for UI/UX enhancements (e.g., Socket.IO client, Material-UI or Tailwind CSS).

2. Backend – Node.js with Express.js

- Handles server-side logic, API requests, and real-time communication.
- Express.js for building RESTful APIs and managing middleware.
- Integration of AI services and sandboxed code execution.

3. Database – MongoDB

- NoSQL database for flexible schema and fast data storage.
- Stores user data, chat history, code snippets, and server configurations.
- Ensures scalability for handling large amounts of chat data and user requests.

4. Real-Time Communication – Socket.IO

- Enables real-time bidirectional communication between the client and server.
- Used for chat messaging, code execution output, and server status updates.

5. AI Integration

- **OpenAI API (or similar AI services)** for natural language processing and code generation.
- Provides users with real-time AI assistance for queries and code suggestions.

Development Strategy for ThinkSync

A well-structured development strategy ensures that the project is delivered on time, meets user requirements, and remains scalable and secure. The development of **ThinkSync** will follow the **Agile methodology**, focusing on iterative development and continuous feedback. The strategy is broken into multiple phases as outlined below:

Phase 1: Planning and Requirements Gathering

1. Define the project's goals, features, and scope.
2. Identify key technologies (MERN stack, Socket.IO, OpenAI API, etc.).
3. Create detailed wireframes and mockups for the UI.
4. Prepare an initial database schema and ER diagram.

Phase 2: System Design and Architecture

1. Design a scalable system architecture.
    - Use a microservices-based architecture for handling real-time communication, AI integration, and server deployment.
2. Develop API design for frontend-backend communication.
3. Finalize database design (MongoDB collections and relationships).
4. Define security protocols (authentication, role-based access, etc.).

Phase 3: Frontend Development

1. Build the core UI using **React.js**.
2. Implement real-time chat functionality using **Socket.IO**.
3. Integrate AI assistant interaction with the **OpenAI API or Gemini**.
4. Build components for:
    - Code editor for writing and executing code snippets.
    - Chat interface with message history.
    - Dashboard for managing servers and user roles.
5. Ensure a responsive and user-friendly interface with **Tailwind CSS** or **Material-UI**.

Phase 4: Backend Development

1. Set up the server using **Node.js** and **Express.js**.
2. Implement RESTful APIs for CRUD operations and AI requests.
3. Enable real-time communication using **Socket.IO** on the server side.
4. Integrate with **MongoDB** for data storage.
5. Implement authentication and authorization with **JWT** and **OAuth2**.
6. Develop sandboxed code execution and Node.js server deployment services.

Phase 5: Testing and Quality Assurance

1. **Unit Testing**: Test individual components and functions.
2. **Integration Testing**: Ensure smooth interaction between frontend, backend, and third-party services.
3. **Security Testing**: Perform vulnerability assessments for secure code execution and server deployment.
4. **User Acceptance Testing (UAT)**: Gather feedback from beta testers and make improvements.

Phase 6: Deployment and Hosting

1. Deploy the backend services on **AWS, Heroku, or Vercel**.
2. Deploy the frontend on **Netlify or Vercel**.
3. Use **Docker** for containerizing Node.js server instances.
4. Set up monitoring and logging services (e.g., AWS CloudWatch or LogRocket).

Phase 7: Maintenance and Future Enhancements

1. Regularly monitor the system for bugs and performance issues.
2. Add new features based on user feedback.
3. Scale the application as the user base grows.
4. Ensure data backups and security updates.

Module Descriptions for ThinkSync

The ThinkSync platform is divided into several modules, each responsible for specific functionalities to ensure a seamless collaborative experience. Below is a detailed description of each module:

1. User Management Module

Description: This module handles user registration, authentication, and role-based access control.
Key Features:

- User sign-up, login, and logout.
- Role-based permissions (Admin, Moderator, Member).
- Password encryption and secure authentication using JWT and OAuth2.
- Profile management.

2. Chat and Collaboration Module

Description: This is the core module for real-time communication and collaboration between users.
Key Features:

- Real-time messaging using **Socket.IO**.
- Creation and management of chat rooms.
- Message history and storage in MongoDB.
- Notifications for new messages and mentions.

3. AI Assistant Module

Description: This module integrates AI to provide intelligent assistance within the chat interface.
Key Features:

- Natural Language Processing (NLP) for understanding user queries.
- Code generation and explanations using OpenAI API.
- Context-aware responses to user questions.
- Error detection and suggestions for code improvement.

4. Code Execution Module

Description: Allows users to write, share, and execute code snippets directly within the chat. **Key Features:**

- Real-time code editor (using **Monaco Editor** or similar).
- Supports multiple programming languages, focusing on JavaScript and Node.js.
- Instant code output display in the chat.
- Sandboxed environment for secure code execution.

5. Node.js Server Deployment Module

Description: Enables users to create and deploy Node.js servers from within the chat interface. **Key Features:**

- Simple configuration for server creation.
- Automatic deployment using containerization (Docker).
- Real-time server status updates.
- Isolated and secure server environments.

6. Database Module

Description: This module manages the storage and retrieval of data for the platform. **Key Features:**

- MongoDB for flexible and scalable data storage.
- Collections for users, messages, code snippets, AI requests, and server configurations.
- Efficient query optimization for real-time performance.

7. Security Module

Description: Ensures the platform is secure and user data is protected. **Key Features:**

- Secure authentication and authorization using JWT and OAuth2.
- End-to-end encryption for chat messages.
- Input validation and output sanitization for code execution.