**THINKSYNC:** A COLLABORATIVE GROUP CHAT PLATFORM WITH INTEGRATED AI

**Second Synopsis**

Submitted to the Faculty of Engineering and Technology

For the partial fulfilment of the requirements of

**Master of Computer Applications**

**Supervised By: Submitted By:**

Rajwinder Kaur Vishal (28212301618)

Class: MCA (TYP) 3rd Sem



**Master of Computer Applications**

**Department of Computer Science**

**Guru Nanak Dev University Amritsar-143005 India March, 2025**

|  |  |  |
| --- | --- | --- |
| **Sr No.** | **Topic** | **Page No.** |
| 1 | Problem solution with modular description and working | 2 |
| 2 | DFD diagram of ThinkSync | 6 |
| 3 | ER diagram of ThinkSync | 8 |
| 4 | Flowcharts of ThinkSync | 10 |
| 5 | Usecase Diagrams | 11 |
| 6 | Project Timeline | 15 |
| 7 | Conclusion (Overall progress of report) | 16 |

**Solution and Modular Description with Working**

**Problem 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.

**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.

**Working of the Project**

**1. User Authentication**

* Users sign up/login using email and password.
* Passwords are hashed and stored securely.
* JWT tokens are generated for secure authentication.

**2. Real-Time Chat**

* Users can create chat rooms or join existing ones.
* Messages are sent in real-time using Socket.io.
* Messages are stored in the database for future reference.

**3. AI Integration**

* Users can ask AI questions within the chat.
* AI processes the query and returns a response instantly.
* AI queries and responses are stored for tracking purposes.

**4. Code Execution**

* Users can write and execute code snippets inside the chat.
* The backend processes the code, executes it in a safe environment, and returns the output.
* The executed code and output are stored for collaboration and debugging.

**5. Node.js Server Deployment**

* Users can deploy Node.js servers directly from the chat.
* The backend creates and runs a Node.js server.
* Users get a URL to access their running server.

**6. Notifications and User Interaction**

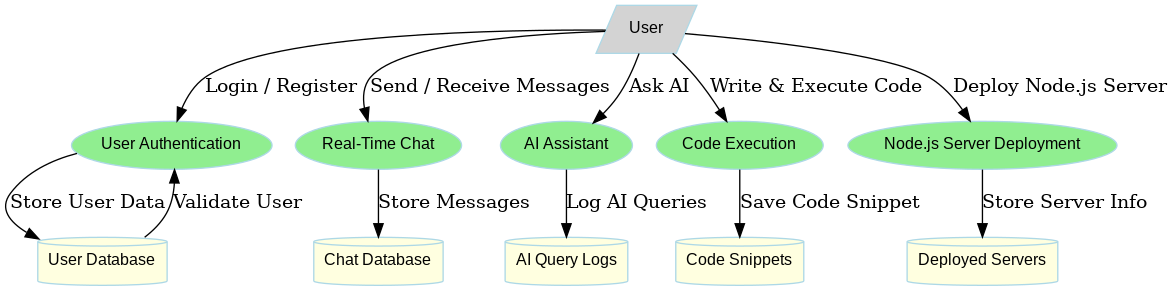
* Users get real-time notifications when:
  + A new message is sent.
  + AI responds to a query.
  + Code execution is completed.
  + A new Node.js server is deployed.

**DFD Diagram of ThinkSync**

**DFD Explanation**

The Data Flow Diagram (DFD) for ThinkSync represents the overall flow of data within the system, covering authentication, chat, AI interactions, code execution, and server deployment.

**1. External Entity:**

* **User:** The primary entity that interacts with the system. Users can register, log in, chat, use AI, execute code, and deploy servers.

**2. Processes:**

* **User Authentication:** Handles user login and registration, storing credentials securely.
* **Real-Time Chat:** Manages messaging between users and stores messages in the chat database.
* **AI Assistant:** Processes user queries and returns AI-generated responses while logging interactions.
* **Code Execution:** Allows users to write and execute code, storing results for future reference.
* **Server Deployment:** Enables users to deploy and manage temporary Node.js servers.

**3. Data Stores:**

* **User Database:** Stores registered users' information.
* **Chat Database:** Stores all messages exchanged between users.
* **AI Query Logs:** Stores AI interactions for future reference.
* **Code Snippets Database:** Stores user-submitted and executed code.
* **Deployed Servers Database:** Stores details about user-deployed Node.js servers.

**4. Data Flow:**

* Users interact with the system by sending requests to various modules.
* Each module processes the request and updates the respective database.
* The system responds by returning results to the user in real time.

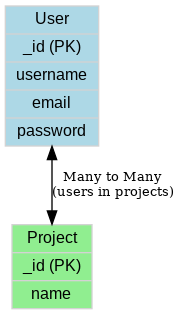
**ER Diagram of ThinkSync**

**ER Explanation**

The **ER Diagram for ThinkSync** represents the database structure, showing how different entities interact within the system.

**1. Entities and Attributes:**

* **User:** Represents registered users of the system.
  + Attributes: \_id (PK), username, email, password
* **Project:** Represents collaborative projects that users participate in.
  + Attributes: \_id (PK), name



**2. Relationships:**

* **User and Project Relationship:**
  + A many-to-many relationship exists between users and projects.
  + A single user can be part of multiple projects, and a project can have multiple users.
  + This is implemented using an array of user IDs inside the Project schema.

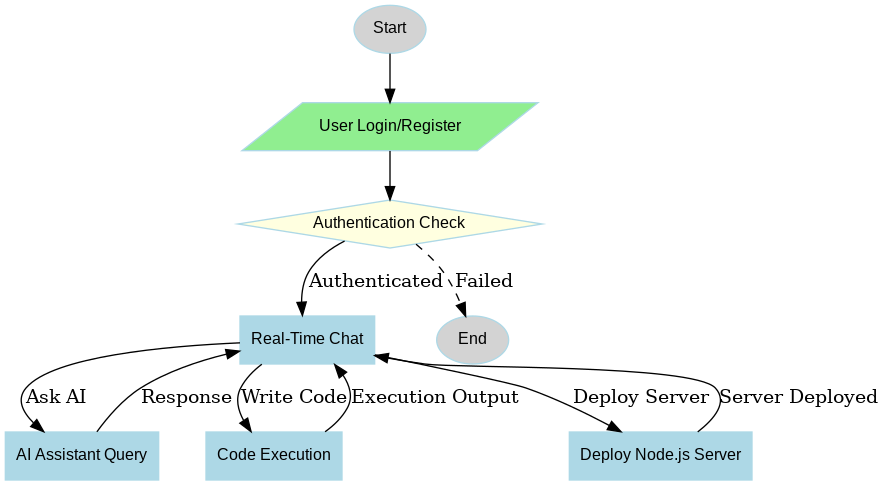
**3. Data Flow and Purpose:**

* **User authentication and management**: The User entity handles authentication and profile details.
* **Collaboration in projects**: The Project entity helps in grouping users into different projects, enabling efficient collaboration.
* **Referential Integrity**: The foreign key (users in the Project entity) ensures data consistency between users and projects.

**Flowchart of Thinksync**

**Complete System Flowchart (workflow)**

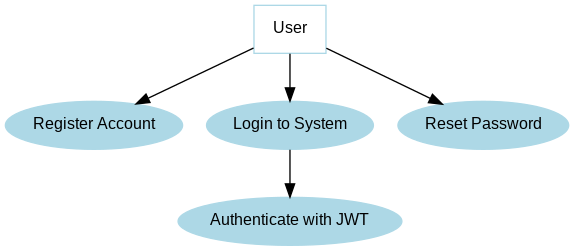
**Description:**This flowchart represents the overall workflow of ThinkSync, integrating multiple functionalities into a single system. The process follows these steps:

****

* **User Authentication** – Users either register or log in. If authentication is successful, they proceed; otherwise, they must retry.
* **Chat System** – Users can send and receive messages in real time through WebSockets.
* **AI Assistant** – Users can interact with an integrated AI to ask questions or generate responses.
* **Code Execution** – Users can write and execute code within the chat, receiving real-time outputs.
* **Server Deployment** – Users can create and deploy Node.js servers, which are validated and hosted.
* **Final Output** – Based on user actions, ThinkSync processes and returns results accordingly.

**Use Case Diagram of ThinkSync**

**1. User Authentication Use Case Diagram**



**Description:**  
This diagram represents the authentication flow in ThinkSync. It includes:

* **Register Account:** Users create a new account.
* **Login to System:** Existing users authenticate with email and password.
* **Authenticate with JWT:** The system validates user credentials and issues a secure JWT token.
* **Reset Password:** Users can request a password reset if needed.

This ensures secure access control and user identity verification in the system

**2. Chat System Use Case Diagram**

**Description:**  
This diagram illustrates the real-time messaging functionality of ThinkSync. Users can:

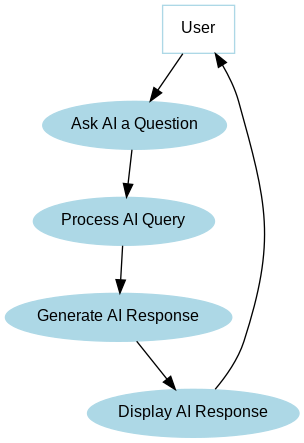
* **Send Messages:** Input text, which is transmitted via WebSockets.
* **Receive Messages:** Messages are received and displayed in the chat interface.
* **Real-time Communication:** Ensures seamless, synchronous message delivery for users.

**A diagram of a user

AI-generated content may be incorrect.**

This feature supports team collaboration and interactive discussions within ThinkSync.

**3. AI Assistant Use Case Diagram**

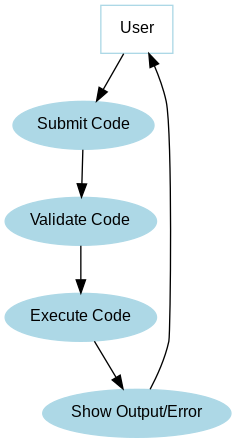
****

**Description:**  
This diagram explains the AI-powered chatbot integration in ThinkSync. Users can:

* **Ask AI a Question:** Submit queries to the AI assistant.
* **Process AI Query:** The system analyzes and forwards the question to the AI model.
* **Generate AI Response:** AI processes the query and generates an appropriate response.
* **Display AI Response:** The system returns the AI-generated response to the user.

This feature provides intelligent assistance within the chat environment.

**4. Code Execution Use Case Diagram**

****

**Description:**  
This diagram details the process of executing user-submitted code in ThinkSync. Users can:

* **Submit Code:** Input programming code within the chat.
* **Validate Code:** The system verifies syntax and correctness.
* **Execute Code:** The validated code is processed in a secure execution environment.
* **Show Output/Error:** The system returns either the program output or an error message.

This enables users to run code snippets and debug directly within the chat.

**5. Server Deployment Use Case Diagram**

**Description:**  
This diagram outlines the Node.js server deployment process. Users can:

* **Submit Server Code:** Upload their Node.js server code.
* **Validate Server Code:** The system checks for errors before deployment.
* **Deploy Server:** The validated server is hosted on the cloud/local server.
* **Monitor Server:** The system tracks server uptime and health.

**A diagram of a server

AI-generated content may be incorrect.**

* **Return Server URL & Status:** A deployment link or status is provided to the user.

This feature allows users to **deploy live servers directly from the chat.**

**Project Timeline of ThinkSync**

**Timeline**

* **Week 1-3:** Requirement gathering, research, and feasibility study.
* **Week 4-6:** UI/UX design, database design, and ER diagram creation.
* **Week 7-14:** Frontend and backend development, integrating authentication and chat system.
* **Week 15-18:** AI integration, real-time communication setup, and Node.js execution.
* **Week 19-20:** System testing, bug fixing, and performance optimization.

|  |  |  |
| --- | --- | --- |
| **Phase** | **Task Description** | **Duration** |
| **Phase 1:** Planning & Requirement Analysis | Identifying system requirements, use cases, and feasibility study. | 3 weeks |
| **Phase 2:** Design | Creating wireframes, database design, ER diagrams, and architecture. | 3 weeks |
| **Phase3:**Frontend Development | Developing UI components, authentication, and chat interface. | 4 weeks |
| **Phase 4:** Backend Development | Setting up Node.js, Express, database models, and API endpoints. | 4 weeks |
| **Phase 5:** AI & Code Execution Integration | Implementing AI chat, code execution, and AI-assisted responses. | 3 weeks |
| **Phase 6:** Server Deployment Feature | Enabling Node.js server creation and real-time execution. | 2 weeks |
| **Phase 7:** Testing & Debugging | Unit testing, integration testing, bug fixes. | 2 weeks |

**Conclusion (Overall progress report)**

The development of ThinkSync, a collaborative group chat web application with AI integration, code execution, and server deployment, has been systematically planned and executed. I have designed this project using the MERN (MongoDB, Express.js, React, Node.js) stack to ensure scalability, efficiency, and seamless real-time communication.

Through the initial stages of requirement analysis and system design, I defined the project's scope, user interactions, and database architecture. The ER diagrams, data flow diagrams (DFD), and use case models provide a structured representation of system workflows.

The implementation phase has been divided into modular components, including:

* User Authentication & Chat System (Real-time messaging using WebSockets).
* AI Assistant (Providing AI-powered assistance within chats).
* Code Execution (Allowing users to write and run code within the chat).
* Node.js Server Deployment (Enabling server setup and execution within the platform).

With a structured development timeline, I have allocated specific durations to each phase—from planning to deployment—ensuring steady progress. The Gantt chart illustrates the key milestones and expected completion dates.

Upon successful completion, ThinkSync will provide a comprehensive collaborative environment where developers, teams, and AI-driven interactions come together to enhance productivity and efficiency. Future improvements may include enhanced AI capabilities, multi-language code execution, and better server management to expand its potential.

This report has laid a solid foundation for the successful development and deployment of ThinkSync. With ongoing progress, I am confident that the project will meet its functional objectives within the planned timeline.