Software Design Document (SDD)

Project Title:
Real-Time Collaborative Online Integrated Development Environment (RCO-IDE)

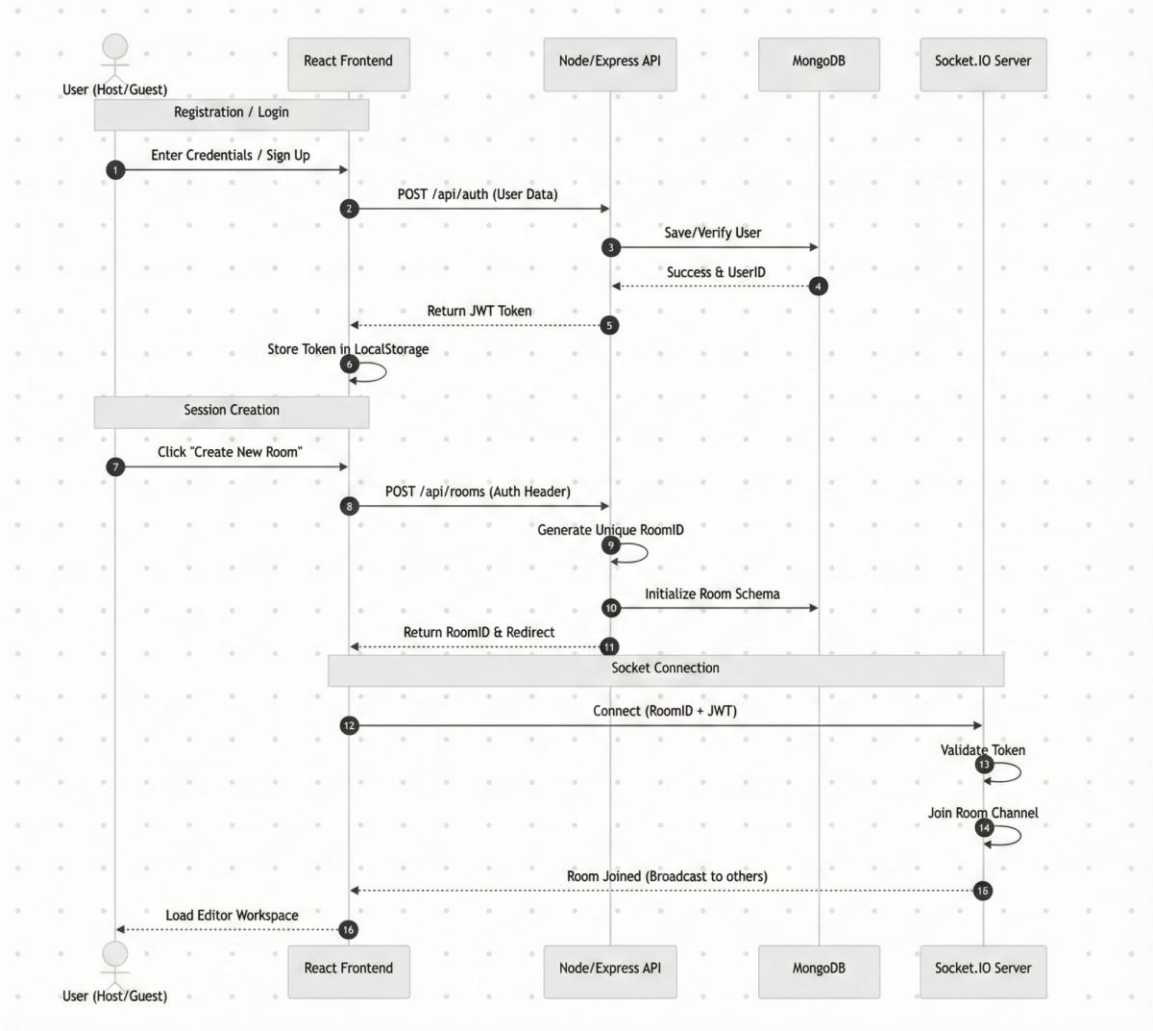
1. Introduction

This Software Design Document (SDD) provides a comprehensive and detailed description of the design architecture of the Real-Time Collaborative Online Integrated Development Environment (RCO-IDE). The document explains the overall system architecture, component-level design, data flow, database structure, and design constraints. It serves as a technical blueprint for academic evaluation and future system implementation.

2. System Architecture

The system architecture follows a layered and modular design approach. Each layer is responsible for a specific functionality, ensuring separation of concerns, scalability, and maintainability. The major layers include the Presentation Layer, Business Logic Layer, Integration Layer, and Data Layer.

Overall System Architecture Diagram:



The presentation layer interacts with users through a web-based interface. The business logic layer handles authentication, session management, and orchestration of real-time communication. The integration layer manages secure code execution, while the data layer handles persistence and caching.

3. Detailed Design

The detailed design section describes the internal working of each system component, including data flow, logic execution, and inter-module communication.

3.1 Presentation Layer Design

The presentation layer is implemented using React.js and the Monaco Editor. It provides features such as user registration, login, real-time code editing, and output visualization. JWT tokens are stored securely in the browser to maintain authenticated sessions.

3.2 Business Logic Layer Design

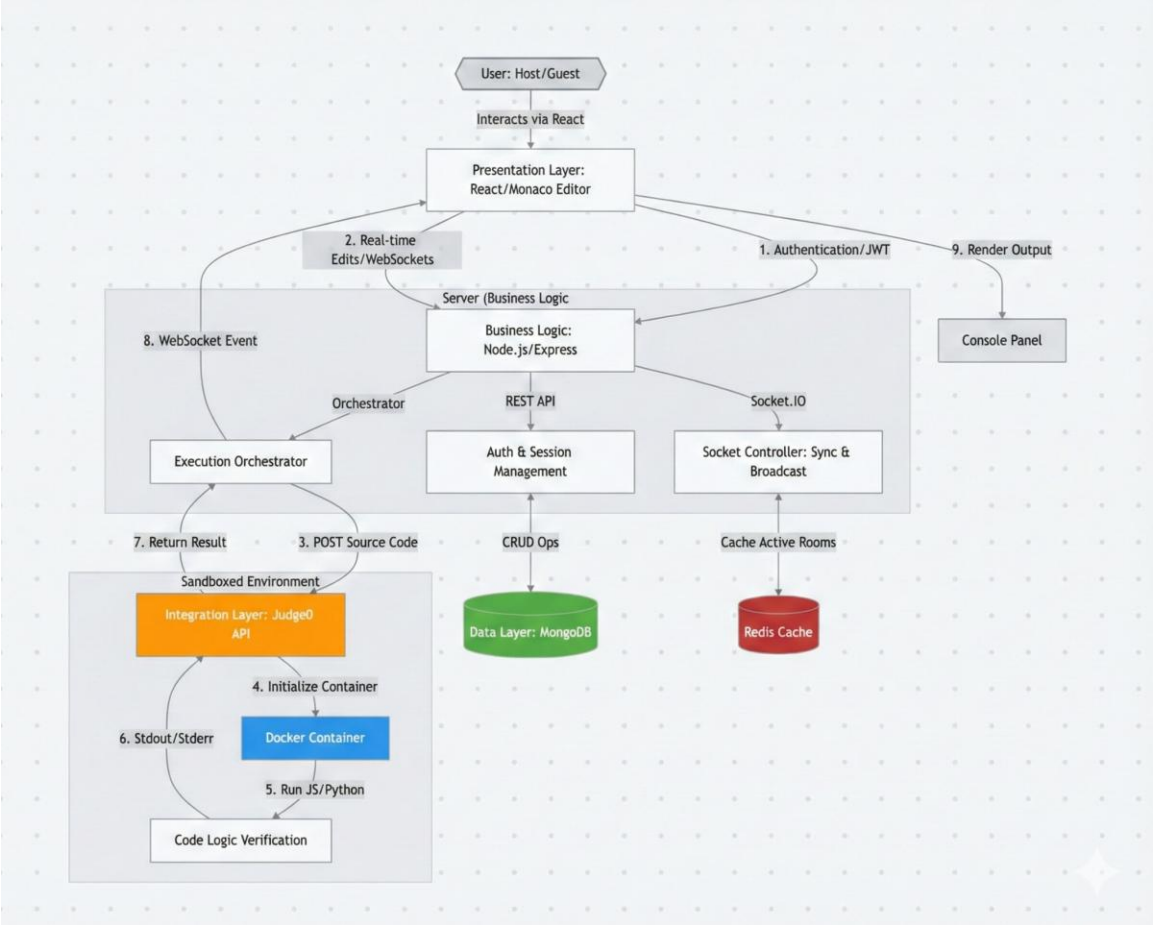
The business logic layer is implemented using Node.js and Express.js. It exposes RESTful APIs for authentication and room management and uses Socket.IO for real-time

communication. This layer coordinates code synchronization, user authorization, and execution requests.

3.3 Integration and Execution Design

The integration layer communicates with the Judge0 API to execute user-submitted code inside Docker containers. This ensures strict sandboxing, resource limitation, and prevention of malicious code execution.

Detailed Workflow / Sequence Diagram:



The sequence diagram illustrates the complete workflow from user authentication to session creation, WebSocket handshake, real-time synchronization, and workspace loading.

4. Database Design

The database design focuses on efficient storage and retrieval of user data, session information, and code snapshots.

4.1 Entity Description

User Entity:
Stores user credentials, authentication details, and profile information.

Session Entity:
Stores room identifiers, host information, active status, and permissions.

Snapshot Entity:
Stores versioned code states along with timestamps for recovery and history.

MongoDB is used as the primary database due to its schema flexibility and scalability. Redis is used as an in-memory cache to manage active socket sessions and improve real-time performance.

5. Design Constraints

Hardware Constraints:
Client systems must have a minimum of 4GB RAM. Server resources are constrained by container execution limits.

Software Constraints:
The system supports only JavaScript and Python programming languages.

Security Constraints:
All code execution must occur inside isolated Docker containers with no external network access.

Network Constraints:
Low-latency and stable internet connectivity is required for real-time collaboration.

6. Conclusion

This Software Design Document presents a detailed and structured design of the Real-Time Collaborative Online IDE. The layered architecture, secure execution model, and real-time synchronization mechanisms make the system robust, scalable, and suitable for academic and real-world applications.