High Level Design Documentation

Problem Statement

Design a scalable and reliable online judge system that facilitates the evaluation of programming solutions, provides real-time feedback to users, and fosters a competitive coding environment.

Breakdown of Key Requirements

- **Problem Management:**
  - The system must allow for the secure storage and retrieval of a diverse library of programming problems, including problem descriptions, test cases, difficulty levels, and associated topics.
  - Administrators or authorized users should have the ability to add, edit, and delete problems.
- **Code Submission and Execution:**
  - The system must provide a user-friendly interface for code submission, supporting multiple programming languages.
  - It needs a secure and isolated execution environment to run submitted code against test cases without compromising system integrity.
- **Evaluation and Feedback:**
  - The system must automatically compare code output against expected outputs for various test cases.
  - Users should receive detailed and timely feedback on their submissions, including test cases passed/failed, execution time, and error messages.
- **User Management**
  - The system must implement secure user registration and authentication processes.
  - It should maintain user profiles with records of submission history, solved problems, and performance metrics.
- **Leaderboards and Contests**
  - The system should be able to calculate and display leaderboards to rank users based on their problem-solving success and efficiency.
  - It should have the capability to host timed programming contests with well-defined start/end times and scoring rules.

Additional Considerations

- **Scalability:** The system should handle a large number of users, submissions, and test cases without performance degradation.
- **Security:** The system must be protected against malicious code, potential hacking attempts, and unauthorized access to sensitive data.

Front End (Client Side)

- **Coding Environment:** Text editor with syntax highlighting, code formatting, language selection, input/output areas.
- **Problem Library:** Organized display of programming problems with search, filtering (difficulty, topics), and problem descriptions.
- **Submission Area:** Code submission, selection of programming language.
- **Real-time Test Results:** Display of test cases passed/failed, error messages, execution time.
- **User Profiles:** Registration, login, submission history, leaderboards.

Tech Stack - Angular

Back End

- **Problem Database:** Storage of problem descriptions, test cases (input/expected output), difficulty levels, associated topics.
- **Code Compiler/Executor:** Sandboxed environment to compile and run submitted code against test cases for various supported programming languages.
- **Test Case Engine:** Manages test cases, executes them against user code, provides detailed feedback.
- **User Authentication and Database:** Securely manages user accounts, submissions, and related data.
- **Leaderboard System:** Calculates and ranks user scores based on problem-solving success and efficiency.
- **WebSockets:** For real-time updates on submission results and leaderboard dynamics. Still under consideration...
- **REST API:** For frontend-backend communication (problem retrieval, submission, user profile updates).

Tech Stack - Spring Boot, Docker

Database

Core Tables

1. **problems**
   - id
   - title
   - description
   - difficulty
   - constraints
   - created_at
2. **topics**
   - id
   - name
3. **problems_topics** (Mapping table for many-to-many relationship)
   - problem_id
   - topic_id
4. **test_cases**
   - id
   - problem_id
   - input
   - expected_output
5. **users**
   - id
   - username
   - email
   - password_hash
   - registered_at
6. **submissions**
   - id
   - user_id
   - problem_id
   - code
   - language
   - verdict
   - execution_time
   - submitted_at

Tech Stack - Postgres or MongoDB
AWS RDS(If needed)

Cloud Services

Cloud Provider: AWS

Key AWS Services

- **Amazon EC2 (Elastic Compute Cloud):** Virtual servers for the backend application instances. Choose appropriate instance types based on processing power and memory.
- **Amazon RDS (Relational Database Service):** Managed database service (e.g., PostgreSQL or MySQL) for storing problems, users, submissions, etc.
- **AWS Elastic Load Balancer:** Distributes traffic across multiple EC2 instances for scalability.
- **Amazon S3 (Simple Storage Service):** Cost-effective storage for test cases (optional, test cases could also be stored in the database).
- **Amazon CloudFront (Optional):** Content delivery network (CDN) for caching static assets (images, CSS, scripts) to improve frontend load times globally.
- **Amazon Route 53:** For DNS management and domain name routing.
- **AWS CodeBuild (Optional):** For setting up a continuous integration/deployment (CI/CD) pipeline to automate code deployment.

Time Required

Front End : 2 Week
Backend : 2 Week
Database : 1 Week
Cloud Deployment : 1 Week