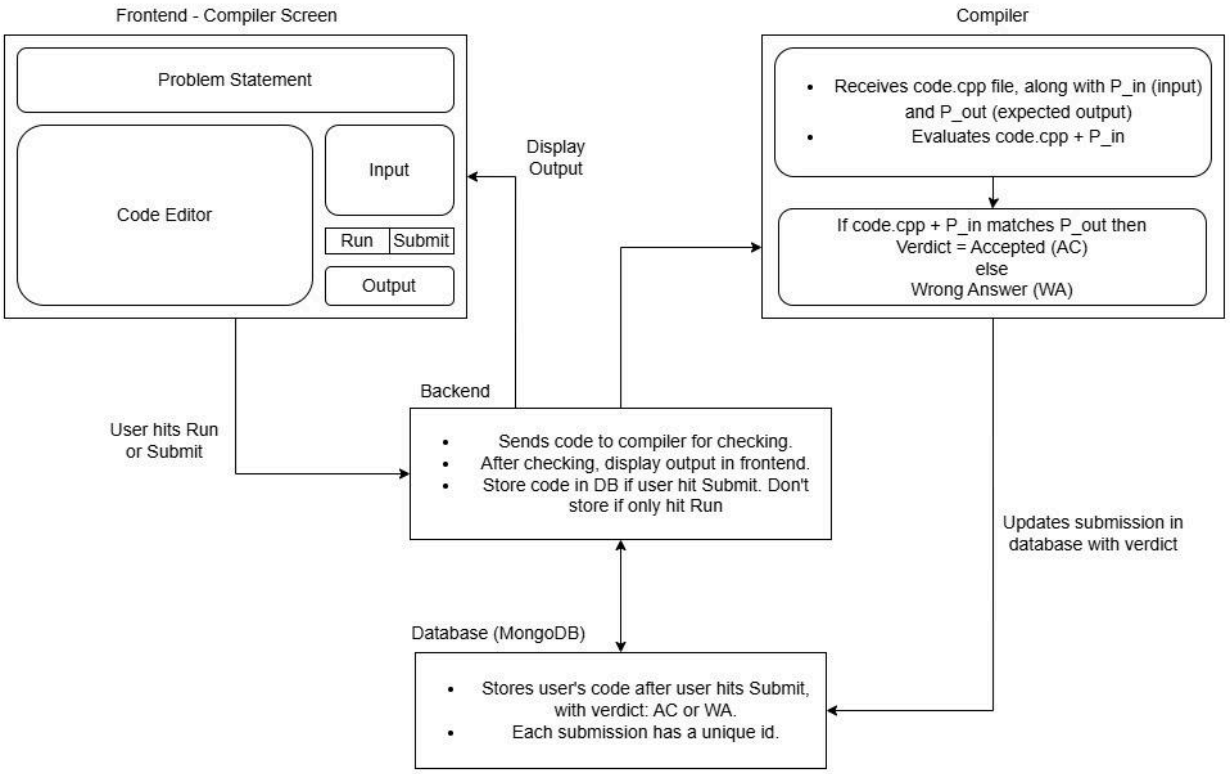
Online Judge

Online Judge is a MERN based full stack web application that allows users to register themselves on the platform, and solve problems by writing and executing code in its compiler. It can also host coding competitions in a fast and secure manner. It also includes advanced features such as leaderboards, submission history, and AI-generated code explanations to support learning, performance tracking, and interview preparation.

System Architecture and Design:



1. Frontend - React.js

   React is component based and its reusable structure will suit us for pages like problem list, problem page, leaderboard, user profile and history of submissions. React router has easy routing for screen transitions.

   It has fast updates via virtual DOM so it is suitable for real-time verdict display. Its ecosystem consists of Monaco for code editor and Axios or Fetch for API, which will be very helpful.

2. Backend - Node.js + Express.js

   Good for many concurrent requests (code submission and problem fetching). It has easy integration with MongoDB using Mongoose. It will handle user authentication using JWT, check user submissions by calling the compiler and then display verdict at frontend.

3. Database - MongoDB

   Flexible schema allows easy updates, like adding new fields such as problem difficulty, topic tags, etc. Schema will contain - user, problem, submission, test_case.

4. Code Execution Enginer - Docker + Node.js

   Docker provides secure, isolated sandboxes. It will prevent malicious code from affecting our system. It can run multiple containers concurrently across a job queue. Each submission will have its own container.

5. Plagiarism Detection - MOSS API

   It is an established tool for detecting code similarity and can be integrated to flag plagiarism.

6. AI Integration - OpenAI GPT-4 or Gemini

   Will provide AI analysis of code, including time and space complexity, and make notes by explaining the code interview-style, for future reference of user.

User Flow -

1. User Interaction (Frontend – Compiler Screen)
    - The user opens a problem and sees the problem statement, code editor, and input/output panels.
    - User writes code in the editor and either clicks:
        - Run (to test code with sample input)
        - Submit (to evaluate code against hidden test cases)

2. Frontend to Backend Communication
    - On clicking Run or Submit, the frontend sends:
        - User's code
        - Problem ID
        - Input (if running custom inputs or predefined samples)
        - Language selected

3. Backend Processing
    - Backend receives the request and sends the code + input to the compiler.
    - Waits for the compiler’s verdict (AC/WA) and output.

4. Compiler Logic
    - Compiler receives the code file along with:
        - P_in (input)
        - P_out (expected output)
    - Executes the code inside an isolated sandbox (Docker).
    - Compares output from code execution to P_out.
        - If they match → Verdict = Accepted (AC)
        - Else → Verdict = Wrong Answer (WA)
    - Returns output + verdict back to backend.

5. Backend Post-processing
    - If Run:
        - Simply returns output to frontend.
        - Does NOT store anything in DB.

    - If Submit: Stores the submission in MongoDB:
        - User ID
        - Problem ID
        - Submitted code
        - Verdict (AC/WA)

    - Timestamp
    - Unique Submission ID

6. Frontend Output Display
    - Displays the output and verdict to the user after receiving it from backend.

7. Database Role (MongoDB)
    - Stores submitted code.
    - Each submission stored with a unique ID, linked to:
        - Problem
        - User
        - Verdict
        - Time of submission

Frontend pages:

- Home - Landing page with intro and links to problems, login/register
- Register/login - user authentication forms (JWT-based login)
- Dashboard - user profile, stats, top 10 recent submissions, calendar showing streak
- Problem list - All problems with filter for difficulty and topic tags
- Problem page - problem description, editor, custom input/output, submit/run
- Leaderboard - user ranking based on number of problems solved or contest ranking
- AI analysis page - shows explanation, complexity, and notes for selected submission
- Contest (optional) - contest dashboard, problem list, timer, scoreboard

Schemas:

User schema:

- _id: ObjectId,
- username: String,
- email: String,
- passwordHash: String,
- role: String, ('user' or 'admin')
- submissions: [ObjectId],
- createdAt: Date

Problem schema:

- _id: ObjectId,
- title: String,
- statement: String,
- difficulty: String, ('Easy', 'Medium', 'Hard')
- tags: [String], (topic tags)
- constraints: String,
- createdBy: ObjectId, (admin user)
- testCases: [ObjectId],
- createdAt: Date

Testcase schema:

- _id: ObjectId,
- problemId: ObjectId,
- input: String,
- expectedOutput: String

Submission schema:

- _id: ObjectId,
- userId: ObjectId,
- problemId: ObjectId,
- code: String,
- language: String,
- verdict: String, ('AC', 'WA', 'TLE')
- executionTime: Number, (in ms)
- memoryUsed: Number, (in KB)
- aiExplanation: String, (Optional GPT response)
- submittedAt: Date

API Endpoints:

For Authentication :
POST - /api/auth/register - register a new user
POST - /api/auth/login - logs in (returns JWT token)

For User :
GET - /api/user/id - gets user profile
PUT - /api/user/id - updates user profile

For Problems :
GET - /api/problems - get list of all problems
GET - /api/problems/problem_id - get specific problem
POS/PUT/DELETE - /api/problems - create/update/delete a new problem (admin access only)

For Submissions :
GET - /api/submissions/userid - get list of submissions by user
GET - /api/submissions/submission_id - get a specific submission

For AI Analysis :
POST - /api/aianalysis/id - analyze submission using GPT

For Testcases :
POST - /api/testcase/problem_id - create testcases for problem_id (admin access only)
PUT - /api/testcase/problem_id - update testcases for problem_id (admin access only)
GET - /api/testcase/problem_id - get testcases for problem_id

For leaderboard :
GET - /api/leaderboard - get leaderboard list