Software Design Documentation

Competitive Coding Arena

Date: 29 Dec 2024

Written By: Sandeep

Contributors: Lohit, Yash, Harsh

Table of Contents

---

1. Introduction

1.1 Purpose

The goal of this project is to design and implement a platform similar to Codeforces, providing users with a competitive programming experience. The platform will allow users to:

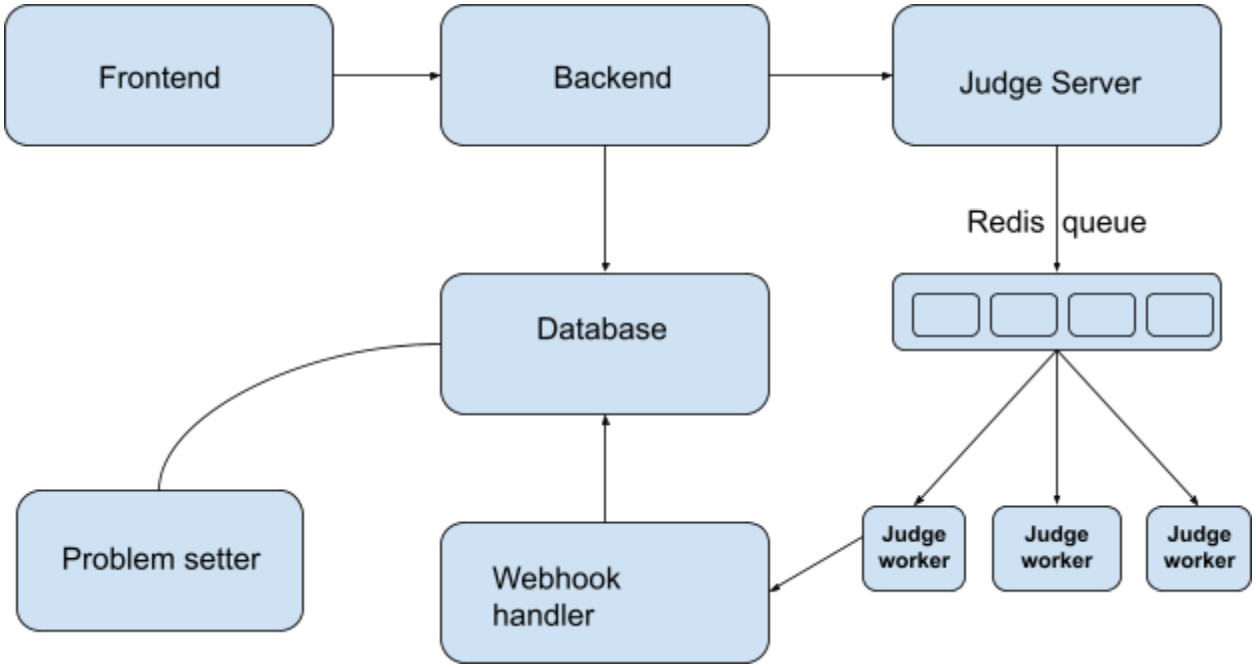
- Read blogs
- Solve algorithmic challenges.
- Participate in contests.
- View live leaderboards.
- Evaluate code submissions in real time.

---

2. System Overview

The Competitive Coding Arena is a multi-tiered platform with components for user authentication, posting blogs, problem management, code evaluation, and leaderboard. Key features include:

- User authentication and profiles.
- Problems Component
- Real-time contest management.
- Secure and scalable code execution.
- Leaderboard tracking and historical data.



---

3. System Components

3.1 Frontend

- **Technology:** React.js
- **Features:** Blogs (private and public), User registration, problem navigation, code editor, leaderboard view, contest component

3.2 Backend

- **Technology:** Python (Flask/FastAPI)
- **Features:** Authentication, problem management, leaderboard updates, API for frontend interaction, Connection with Code Exceution Engine

3.3 Code Execution Engine (Judge)

- **Technology:** Docker-based sandboxing.
- **Features:** Secure, isolated execution of code submissions and scalable.

3.4 Database

- **Technology:** TBD

- **Features:** User profiles, Blogs details, problem metadata, submission tracking, contest data.

3.5 Object Store

- **Technology:** AWS S3 or Google Cloud Storage
- **Features:** Storage for test cases, problem assets, and solution archives.

3.6 Problem Setter

- **Technology:** Flask/FastAPI
- **Features:** Helps admin to set problems, create contest, write blogs etc.

---

4. Detailed Design

4.1 Frontend

- Pending…

4.2 Backed

- Support All API for frontend
- User auth
- Connection with db
- Connection with Judge

Detailed Implemention:

Pending…

4.3 User management (login/signup)

- **Registration/Login**: Support for email-based and social authentication (e.g., Google, GitHub).
- **Profiles**: Public profiles displaying user statistics, submissions, and achievements.
- **User Roles**: Roles such as Admin, Participant and level of user
-

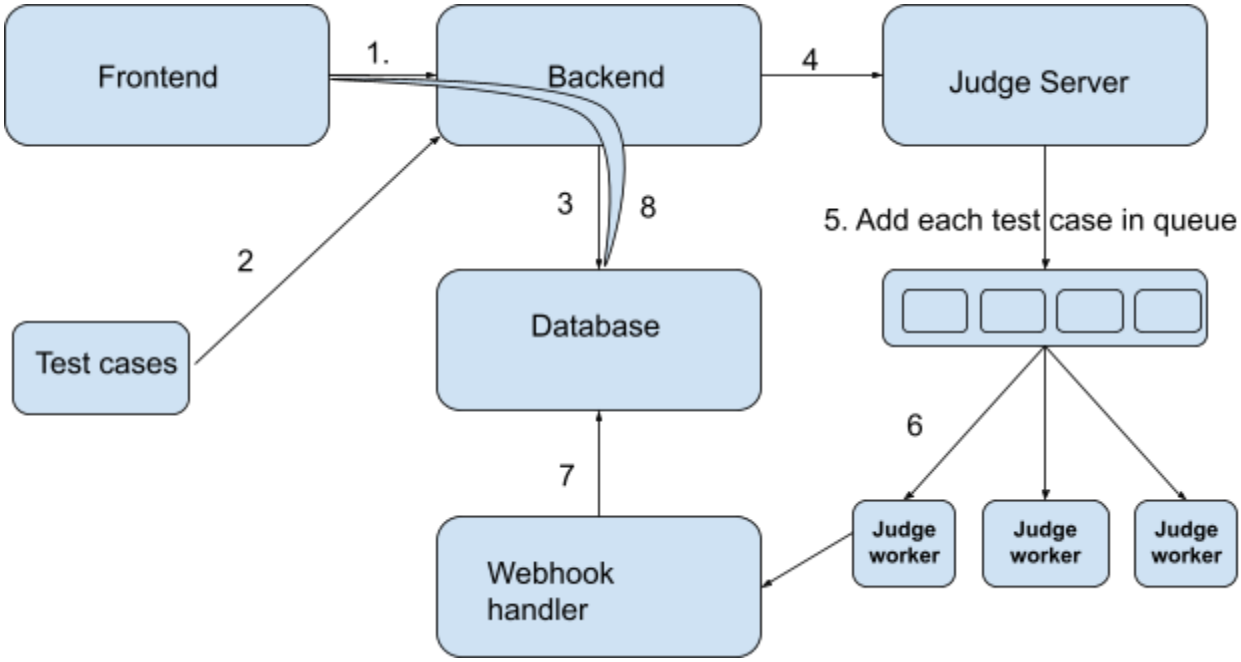
4.4 Blogs

- Blogs metadata will be stored in database and content in Object Stores
- Blog object will be of type:

    - Title
    - Description
    - Tags
    - Owner
    - Date Created
    - Versions published
    - Comments
        - Thread
            - User, comment
    - Content (string)
        - Markdown language
- Backend will support 4 APIs
    - /list_all_blogs (title, owner, metadata, date)
    - /blog_details?blog_id
    - /add_comment?blog_id
    - /admin/create_blog

4.5 Test Cases and Code Evaluation

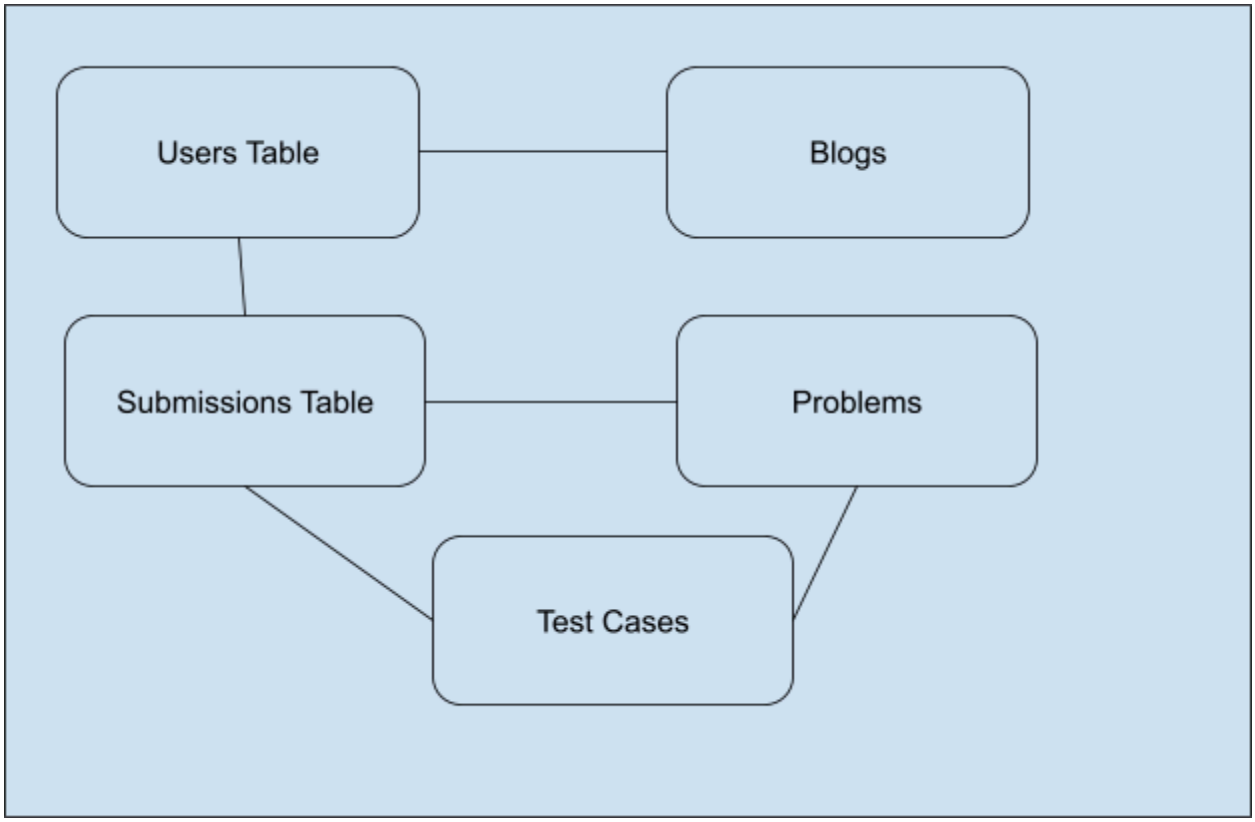
- Test cases will be stored and categorized into sample and hidden sets.
- Evaluation pipeline:
    1. Request from frontend with problem_id, language, user_id, user_code
    2. Fetch all test cases for problem_id
    3. Create entry of submission in Database with status pending
    4. Send Request to Judge
    5. Judge will loop over all the test cases and create entry for each test case in redis queue
    6. Judge worker will execute the submitted code against one test case at a time.
    7. Update the Database entry after each test case completes
    8. Frontend will continuously check status of submission in database after every 2 secs with 10 retires.
    9. If user does not get status, show error code submission in queue



4.6 Judge Server Implementation

- Judge API server
  - Support API so that Server can receive request from Backend
  - Support API to create entry for each test case in redis queue
  - Support API to update Database after executing(using Webhhok handler)
- Judge Redis Queue
  - Create a redis cluster
  - Expose a writer endpoint (under vpc)
  - Expose a reader endpoint (under vpc)
- Judge Worker to execute code
  - Use Docker containers to isolate code execution.
  - Restrict system resources (CPU, memory, disk I/O) to prevent abuse.
  - Monitor for infinite loops and malicious code.
- Deployment
  - Use services like Kubernetes to automate deployment and scaling of workers

4.7 Database



4.7 Caching

- Implement caching using Redis for better performance
-

4.3 Ease of Problem Setting, Boilerplate Generation

- Provide a graphical interface for problem setters to upload descriptions, constraints, and test cases.
- Automatically generate boilerplate code for popular languages (e.g., Python, Java, C++).

4.4 Storing Test Cases: Files vs Object Stores vs Databases

- **Files:** Suitable for local development but lacks scalability.
- **Object Stores:** Preferred for scalability and cost-effectiveness.
- **Databases:** Useful for metadata, but not optimal for large test case storage.
- **Decision:** Use object stores for storing large test case files and a database for indexing and metadata.

4.8 Real-time Leaderboards

- Create most efficient live leaderboards
- Implementation: pending…

---

5. Rollout Plan

Phase 1:

Frontend to show markdown blogs

- [ ] Create landing Page (Lohit)
- [ ] Markdown viewer (Yash)
- [ ] Code viewer (harsh)
- [ ] Frontend deployment (Sandeep)

Phase 2:

- [ ] User auth (using google firebase)
- [ ] Store blogs in database (google firestore)
- [ ] Improve UI
- [ ] Blogs access using level of user

Phase 3:

- [ ] Create Backend, APIs
- [ ] Create Database
    - [ ] Users table
    - [ ] Blogs table
- [ ] Integrate with frontend

Phase 4:

- [ ] Add problems section
- [ ] Code editor
- [ ] Support code submission to database
- [ ] Support for comments in Blogs

Phase 5:

- [ ] Design simple Judge to execute code

Phase 6:

- [ ] Improve Judge using advanced tech

Phase 7:

- [ ] Add support for contests

Phase 8:

- [ ] UI to ease problem setters
- [ ] Admin console

Phase 9:

- [ ] Real time leaderboard

Phase 10:

- [ ] Design a monitoring system for platform
- [ ] Make system more scalable

Phase 11:

- [ ] Get users onboard