# **Online Coding Platform**

## **Project Overview**

Build a functional prototype of an online coding assessment platform that allows administrators to create coding challenges and candidates to solve them. This project will evaluate your proficiency in full-stack development, system design, API implementation, and database management.

## **Key Requirements**

### **Admin Portal**

1. **User Management**: Registration and authentication system with role-based access control
2. **Challenge Management**: Create, edit, and delete coding challenges with descriptions and constraints
3. **Test Case Management**: Upload and manage both public and hidden test cases for each challenge
4. **Candidate Tracking**: View and filter registered candidates and their submissions
5. **Performance Analytics**: View basic statistics on challenge completion rates and execution times

### **Candidate Portal**

1. **User Management**: Registration and authentication system for candidates
2. **List Coding Challenges**: View available coding challenges with sorting/filtering options.
3. **Load Coding Challenge**: Fetch a question and sample test cases
4. **Testing System**: Run code against custom test case with real-time feedback
5. **Submission System**: Submit solutions for evaluation against hidden test cases. You need not have to provide submission results in real time. You can use an async task queue to process submissions.
6. **History Tracking**: View status of previous submissions with results and performance metrics

## **Technical Requirements**

### **Core Technology Stack**

* **Backend**: FastAPI with Python 3.9+
* **Database**:
  + SQL for user data and challenge metadata
  + MongoDB for storing code submissions and test results
* **Authentication**: JWT-based token authentication with proper security measures

### **Execution Environment**

* **Code Execution**: Integration with Judge0 API (free tier) for secure code execution
* **Security**: Implement proper input validation and rate limiting
* **Hidden Test Cases:** The size of hidden test cases can be very large. So, it is recommended to score hidden test cases file in AWS S3 Bucket.
* **Asynchronous Processing**: Implement job queues using Kafka or Celery or RabbitMQ or Async python queues for code execution

### **Optional Enhancements (Bonus Points)**

* **Real-time Updates**: WebSocket implementation for live submission status updates
* **Docker Containerization**: Containerize the application for easy deployment

## **Assessment Criteria**

Your submission will be evaluated based on:

1. **Architecture Design**: Overall system design, API structure, and data flow
2. **Code Quality**: Clean, maintainable code following modern best practices
3. **Security Implementation**: Proper authentication, authorization, and input validation
4. **Database Design**: Schema design, queries, and overall data management approach
5. **Feature Completeness**: Implementation of all core requirements
6. **Documentation**: Clear API documentation and setup instructions

## **Submission Guidelines**

1. Create a GitHub repository with your implementation
2. Include a comprehensive README.md with:
   * System architecture overview
   * API documentation
   * Setup instructions
   * Any additional notes on implementation decisions
3. Include seed data or scripts to populate the database with sample challenges
4. Submit the GitHub repository URL by the deadline

## **Timeline**

* Expected time commitment: 1-2 days
* Code review and feedback: Within one week of submission

## **Note**

This assessment is designed to evaluate your technical skills and problem-solving approach. We value clean, maintainable code over perfect implementation of all features. Focus on demonstrating your strengths and documenting any architectural decisions you make.

**COMPILER APIS**

1. List available Languages

curl --request GET \

--url https://judge0-ce.p.rapidapi.com/languages \

--header 'x-rapidapi-host: judge0-ce.p.rapidapi.com' \

--header 'x-rapidapi-key: 6d9678a2a5msh0b1e9ffb088ee62p125a17jsncdaf210209c8'

1. Create a submission

curl --request POST \

--url 'https://judge0-ce.p.rapidapi.com/submissions?base64\_encoded=true&wait=false&fields=\*' \

--header 'Content-Type: application/json' \

--header 'x-rapidapi-host: judge0-ce.p.rapidapi.com' \

--header 'x-rapidapi-key: 6d9678a2a5msh0b1e9ffb088ee62p125a17jsncdaf210209c8' \

--data '{"language\_id":52,"source\_code":"I2luY2x1ZGUgPHN0ZGlvLmg+CgppbnQgbWFpbih2b2lkKSB7CiAgY2hhciBuYW1lWzEwXTsKICBzY2FuZigiJXMiLCBuYW1lKTsKICBwcmludGYoImhlbGxvLCAlc1xuIiwgbmFtZSk7CiAgcmV0dXJuIDA7Cn0=","stdin":"SnVkZ2Uw"}'

1. Get status of submission

curl --request GET \

--url 'https://judge0-ce.p.rapidapi.com/submissions/2e979232-92fd-4012-97cf-3e9177257d10?base64\_encoded=true&fields=\*' \

--header 'x-rapidapi-host: judge0-ce.p.rapidapi.com' \

--header 'x-rapidapi-key: 6d9678a2a5msh0b1e9ffb088ee62p125a17jsncdaf210209c8'

1. Create a Batched Submission

curl --request POST \

--url 'https://judge0-ce.p.rapidapi.com/submissions/batch?base64\_encoded=true' \

--header 'Content-Type: application/json' \

--header 'x-rapidapi-host: judge0-ce.p.rapidapi.com' \

--header 'x-rapidapi-key: 6d9678a2a5msh0b1e9ffb088ee62p125a17jsncdaf210209c8' \

--data '{"submissions":[{"language\_id":46,"source\_code":"ZWNobyBoZWxsbyBmcm9tIEJhc2gK"},{"language\_id":71,"source\_code":"cHJpbnQoImhlbGxvIGZyb20gUHl0aG9uIikK"},{"language\_id":72,"source\_code":"cHV0cygiaGVsbG8gZnJvbSBSdWJ5IikK"}]}'

1. Get status of Batched submission

curl --request GET \

--url 'https://judge0-ce.p.rapidapi.com/submissions/batch?tokens=dce7bbc5-a8c9-4159-a28f-ac264e48c371%2C1ed737ca-ee34-454d-a06f-bbc73836473e%2C9670af73-519f-4136-869c-340086d406db&base64\_encoded=true&fields=\*' \

--header 'x-rapidapi-host: judge0-ce.p.rapidapi.com' \

--header 'x-rapidapi-key: 6d9678a2a5msh0b1e9ffb088ee62p125a17jsncdaf210209c8'