**Software Requirements**

**Specification**

**for**

**Online Coding and Contest Platform**

**Version 1.0 approved**

**Prepared by Chandan Vispute**

**Members : Aniket Aher (202101103083)  
Aditya Danve (202101103160)  
Abdullah Al Kaseri (202101103085)**

**Date : 14-07-2025**

**Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.**

**Software Requirements Specification for Online Coding and Contest Platform Page ii**

# Table of Contents

[Table of Contents ii](#_Toc8033)

[Revision History ii](#_Toc8034)

[1. Introduction 1](#_Toc8035)

[1.1 Purpose 1](#_Toc8036)

[1.2 Document Conventions 1](#_Toc8037)

[1.3 Intended Audience and Reading Suggestions 1](#_Toc8038)

[1.4 Product Scope 1](#_Toc8039)

[1.5 References 1](#_Toc8040)

[2. Overall Description 2](#_Toc8041)

[2.1 Product Perspective 2](#_Toc8042)

[2.2 Product Functions 2](#_Toc8043)

[2.3 User Classes and Characteristics 2](#_Toc8044)

[2.4 Operating Environment 2](#_Toc8045)

[2.5 Design and Implementation Constraints 2](#_Toc8046)

[2.6 User Documentation 2](#_Toc8047)

[2.7 Assumptions and Dependencies 3](#_Toc8048)

[3. External Interface Requirements 3](#_Toc8049)

[3.1 User Interfaces 3](#_Toc8050)

[3.2 Hardware Interfaces 3](#_Toc8051)

[3.3 Software Interfaces 3](#_Toc8052)

[3.4 Communications Interfaces 3](#_Toc8053)

[4. System Features 4](#_Toc8054)

[4.1 System Feature 1 4](#_Toc8055)

[4.2 System Feature 2 (and so on) 4](#_Toc8056)

[5. Other Nonfunctional Requirements 4](#_Toc8057)

[5.1 Performance Requirements 4](#_Toc8058)

[5.2 Safety Requirements 5](#_Toc8059)

[5.3 Security Requirements 5](#_Toc8060)

[5.4 Software Quality Attributes 5](#_Toc8061)

[5.5 Business Rules 5](#_Toc8062)

[6. Other Requirements 5](#_Toc8063)

[Appendix A: Glossary 5](#_Toc8064)

[Appendix B: Analysis Models 5](#_Toc8065)

[Appendix C: To Be Determined List 6](#_Toc8066)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# 1. Introduction

## 1.1 Purpose

The purpose of this project is to design and develop a robust, scalable, and user-friendly **Online Coding and Contest Platform** that enables users to practice coding problems, participate in programming contests, and evaluate their solutions in real-time. The platform aims to serve students, educators, and competitive programmers by providing a virtual environment that supports code submission, automatic evaluation against test cases, and accurate leaderboard generation.

The system will allow:

* **Participants** to register, solve problems, submit code in supported languages, and view their results and rankings.
* **Administrators** to manage users, add/edit coding problems, create contests, and monitor submission logs.
* **Judging System** to automatically compile and execute submitted code securely, apply time/memory limits, and validate correctness based on predefined test cases.

This platform will mimic the functionality of widely used online judges such as Codeforces, LeetCode, and HackerRank, with a custom backend judging mechanism and a modular design that supports extensibility, maintainability, and educational use.

## 1.3 Intended Audience and Reading Suggestions

This document is intended for developers, testers, project managers, faculty coordinators, and evaluators. Readers should start with the Introduction and proceed to the Overall Description and System Features., such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.

## 1.4 Product Scope

The **Online Coding and Contest Platform** is a web-based application designed to facilitate the learning, practice, and assessment of programming skills through problem-solving and time-bound coding contests. The platform will support multiple programming languages, allow code submission and automatic evaluation, and maintain real-time leaderboards and user profiles.

This software will serve the dual purpose of being a **learning environment** for students and an **assessment tool** for educators, institutions, and recruiters. By providing a standardized, scalable, and secure system for code evaluation, the platform aims to encourage competitive programming, enhance technical skills, and prepare users for technical interviews or competitive exams.

**Key Objectives and Benefits:**

* Enable **registered users** to solve coding problems, participate in live or virtual contests, and view detailed submission results.
* Allow **administrators** to manage users, contests, problems, and system configurations.
* Provide **real-time judging** of code submissions with constraints such as Time Limit Exceeded (TLE), Wrong Answer (WA), and Accepted (AC).
* Maintain **detailed user profiles**, track progress, and rank users based on performance.
* Support **testcase-based problem validation**, custom judging logic, and secure code execution.

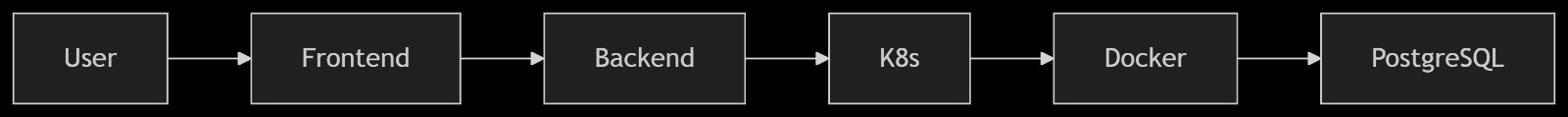
This product aligns with institutional goals of:

* Improving student engagement in programming practice.
* Supporting coding competitions at local or national levels.
* Enabling educators to evaluate coding skills objectively and efficiently.
* Preparing students for real-world coding interviews and hackathons.

If a separate Vision and Scope document is prepared, this product scope will align with the strategic goals mentioned therein and act as a technical realization of that vision.

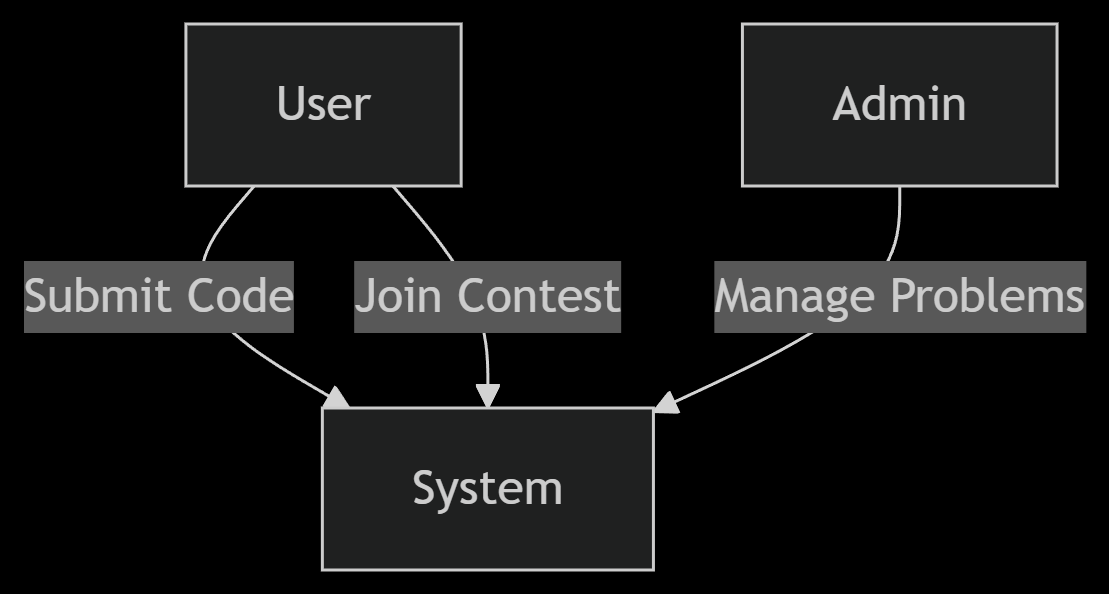
# 2. Overall Description

## 2.1 Product Perspective This platform is a standalone, web-based coding platform similar to Codeforces and LeetCode. It has a modern microservice-oriented architecture that includes the following components: 1. Frontend (React): Provides an intuitive UI for users to submit code, view contests, and receive results. 2. Backend API (Node.js): Handles user interactions, submission queuing, and communication with database and judging services. 3. PostgreSQL: Stores user data, submissions, contests, and system logs. 4. Kubernetes (K8s): Orchestrates and scales Docker containers that run submitted code securely. 5. Docker: Executes submitted code in isolated, language-specific containers. 6. WebSocket: Enables real-time communication for returning submission results and leaderboard updates. System Flow: User → Frontend → Backend → K8s → Docker → PostgreSQL This approach ensures secure execution, scalability, and real-time feedback for coding contests.



System Architecture Diagram:

This platform is a new, standalone product designed to emulate and enhance the features of leading platforms like Codeforces, LeetCode, and HackerRank. It is tailored specifically for educational institutions and independent learners.  
It will consist of three main components:  
1. Web frontend (React)  
2. Backend API (Node.js with MongoDB)  
3. Judging engine (container-based secure code execution system)  
All three communicate over RESTful APIs and sockets where needed.



▼ Diagram: User and Admin Interaction with System

## 

Figure X: End-to-End Python Code Execution Flow in Distributed Judging Architecture

## 2.2 Product Functions

• User registration and login (including OAuth options)  
• Code editor with syntax highlighting  
• Submit, run, and view results of code submissions  
• Real-time scoreboard and problem ranking  
• Contest creation and scheduling  
• Problem management (add/edit/delete)  
• Profile statistics and historical performance  
• Admin dashboard for system and user management  
• Secure backend code judging mechanism

## 2.3 User Classes and Characteristics

• Participants: Can register, solve problems, submit code, join contests, and view rankings. Most users fall in this category.  
• Admins: Have full control over the system. They can manage users, problems, and contests.  
• Judges/Moderators: May assist in verifying problems or reviewing flagged content.  
All users must be authenticated. Participants may range from students to professionals, so interfaces are kept user-friendly.

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

## 2.4 Operating Environment

• Web-based frontend compatible with modern browsers (Chrome, Firefox, Edge)  
• Backend: Node.js with Express, MongoDB for database  
• Judging server: Linux-based containers (Docker)  
• Deployment: Ubuntu 20.04+ with NGINX as reverse proxy, hosted on cloud platforms (e.g., AWS, GCP)

## 2.5 Design and Implementation Constraints

• Secure sandboxing for code execution using Docker  
• Must use open-source stack (Node.js, MongoDB, React)  
• Support for multiple languages (C++, Python, Java initially)  
• High availability and load tolerance expected during live contests  
• Judgment engine must restrict runtime and memory usage per submission

## 2.6 User Documentation

• Web-based user guide  
• Admin documentation for managing contests and system users  
• Help section within the platform  
• Troubleshooting FAQs and support contact

## 2.7 Assumptions and Dependencies

• Users will have access to stable internet and modern browsers  
• All third-party packages (like Ace editor, Docker) are properly installed  
• Cloud hosting is available for high-scale contests  
• Users are familiar with basic programming concepts

# 3. External Interface Requirements

## 3.1 User Interfaces

The platform provides a responsive web UI developed in React:  
• Home Page – displays available contests, login/register options.  
• Dashboard – post-login interface showing attempted problems, contests, and rankings.  
• Code Editor – integrated with syntax highlighting, input/output, and result pane.  
• Admin Panel – interface to manage users, problems, and contests.  
• Mobile-friendly and follows accessibility standards.

## 3.2 Hardware Interfaces

The system interacts with:  
• Web clients via HTTP/S through browser-based interfaces.  
• Judging server hardware through Docker container execution and APIs.  
There is no special hardware interface; the system will run on standard cloud-hosted or on-premise servers.

## 3.3 Software Interfaces

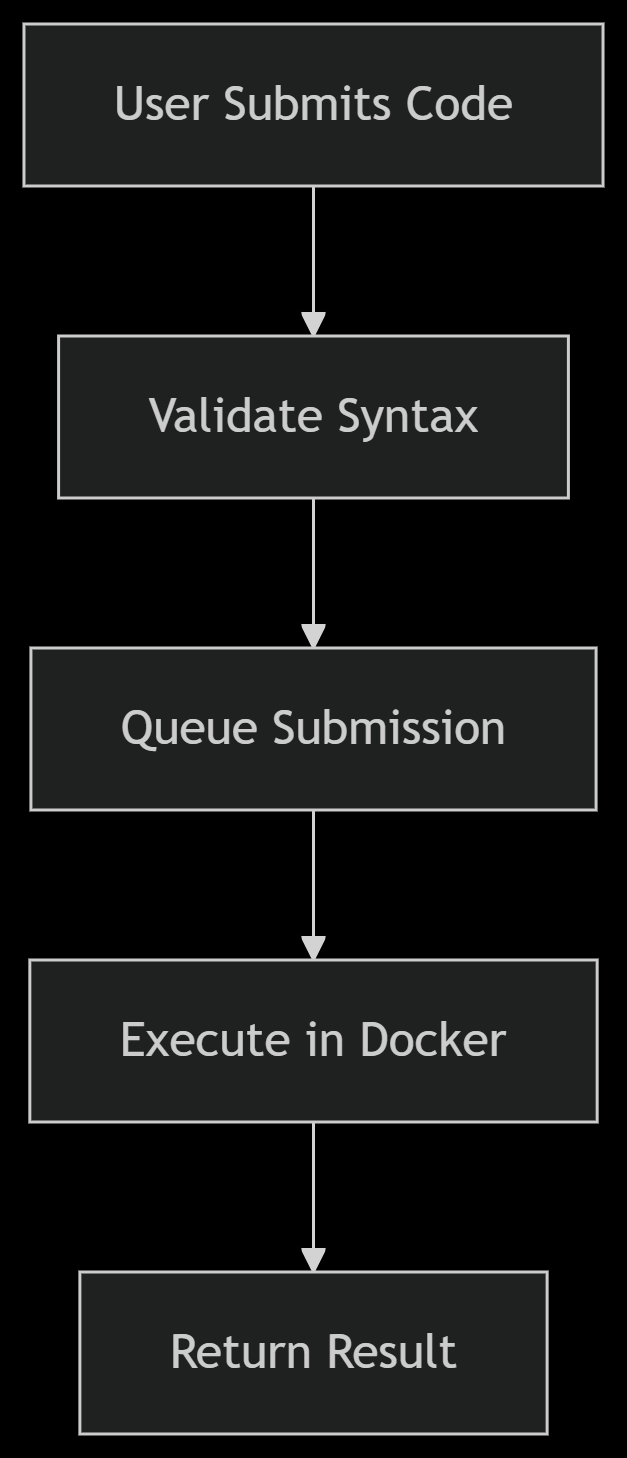
• MongoDB (v6+) – Database for storing users, problems, contests, and submissions.  
• Node.js + Express – RESTful backend server interfacing with frontend and database.  
• Docker Engine – for containerized code execution.  
• Socket.IO – for real-time contest leaderboard updates.  
• Third-party libraries – such as Ace Editor, bcrypt for security, JWT for authentication.

## 3.4 Communications Interfaces

• HTTP/HTTPS – primary communication protocol for frontend-backend.  
• WebSocket – used for real-time leaderboard updates during contests.  
• REST API – structured endpoints used for data exchange (e.g., GET /problems, POST /submit).  
• TLS Encryption – ensures secure communication over HTTPS.

# 4. System Features

## 4.1 System Feature 1



▼ Diagram: Code Submission Workflow

Code Submission and Judging  
  
4.1.1 Description and Priority  
This feature is HIGH priority. It allows users to submit code and get results after evaluation.  
4.1.2 Stimulus/Response Sequences  
- User writes code and clicks "Submit"  
- System sends code to the backend  
- Backend compiles and runs code in isolated container  
- Judging engine compares output to test cases  
- Result (AC/WA/TLE/RE) is returned to user  
4.1.3 Functional Requirements  
REQ-1: System shall compile and execute code securely in Docker containers.  
REQ-2: System shall return output and status (Accepted, Wrong Answer, Time Limit Exceeded, Runtime Error).  
REQ-3: System shall impose limits on time and memory usage.

## 4.2 System Feature 2 (and so on)

Contest Management  
  
4.2.1 Description and Priority  
This feature is HIGH priority. Admins can create and schedule contests.  
4.2.2 Stimulus/Response Sequences  
- Admin selects "Create Contest", adds problems, sets time and duration.  
- Participants see upcoming contests and join.  
- Real-time leaderboard updates during contest.  
4.2.3 Functional Requirements  
REQ-4: System shall allow admin to create, edit, and delete contests.  
REQ-5: System shall track participants, submissions, and scores during contest.  
REQ-6: System shall update leaderboard in real-time.

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

• Judging results should be available within 3 seconds of submission.  
• The platform must handle at least 500 concurrent users during a contest.  
• Leaderboards must update in near real-time (<1s latency).

## 5.2 Safety Requirements

• Judging containers must restrict access to system resources (CPU, memory, file system).  
• Submissions should be scanned for malicious code patterns.  
• System backups must be scheduled daily.

## 5.3 Security Requirements

• JWT-based user authentication.  
• Passwords stored using bcrypt hashing.  
• Only admin users have access to contest creation and problem management.  
• All communications secured with HTTPS.

## 5.4 Software Quality Attributes

• Usability: Intuitive UI with clear workflows.  
• Maintainability: Modular backend and frontend architecture.  
• Reliability: Auto-restart services on crash; retry logic for submissions.  
• Portability: Docker-based deployment for portability across servers.

## 5.5 Business Rules

• Only registered users can participate in contests.  
• A user may not resubmit the same solution more than 5 times in a minute.  
• Problems can only be edited if not yet published.

# 6. Other Requirements

• Database must store submission history with timestamps and verdicts.  
• UI should support both dark and light themes.  
• Platform should support localization (English initially, others in future).

# Appendix A: Glossary

• AC – Accepted  
• WA – Wrong Answer  
• TLE – Time Limit Exceeded  
• RE – Runtime Error  
• JWT – JSON Web Token  
• REST – Representational State Transfer

# Appendix B: Analysis Models

• See attached class diagram (to be added)  
• Data Flow: User -> Submit -> Judge Engine -> Result  
• Entity Relationships: User -< Submission >- Problem

# Appendix C: To Be Determined List

• Whether to support third-party login (Google, GitHub)  
• Selection of scalable hosting (AWS/GCP)  
• Final set of supported programming languages