NED University — CISD

Competitive Coding Arena — SRS

Software Requirements Specification (SRS)

Competitive Coding Arena

Version 1.0

Prepared by

Syed Taha Ali Naqvi (CS-23063)
Muhammad Saad (CS-23148)
Zain Javed (CS-23060)
Aarif Mahdi (CS-23057)

Department of Computer & Information Systems Engineering
NED University of Engineering & Technology

Instructor: Ms. Fakhra Aftab

Submission Date: 21 October 2025 — Tuesday

Contents

Chapter 1
Introduction

1.1 Document Purpose

The purpose of this document is to describe the software requirements specification for the Competitive Coding Arena web application. It provides a detailed outline of system functionality, user interactions, and non-functional requirements, serving as a foundation for design and development.

1.2 Product Scope

The Competitive Coding Arena provides users with an interactive environment for coding competitions. It includes user authentication, matchmaking, real-time code execution, and leaderboard ranking. The platform motivates users to improve their problem-solving abilities while engaging in friendly competition.

1.3 Intended Audience and Document Overview

This document is intended for:

Developers and designers — to implement the system accurately.

Testers — to validate system functions against requirements.

Instructor Ms. Fakhra Aftab — for review and evaluation.

System administrators — to deploy and maintain the web platform.

The SRS is organized as follows:

Chapter 1: Project introduction and definitions.

Chapter 2: Overall description of system functionality and constraints.

Chapter 3: Specific requirements — functional, interface, and behavioral.

Chapter 4: Non-functional requirements and quality attributes.

Appendix A: Data dictionary.

1.4 Definitions, Acronyms and Abbreviations

CCA – Competitive Coding Arena

PDF – Portable Document Format

SRS – Software Requirements Specification

HTTP/HTTPS – Hypertext Transfer Protocol / Secure

DFD – Data Flow Diagram

Graph – A non-linear data structure consisting of nodes and edges

GUI – Graphical User Interface

Latex – Document formatting tool

MongoDB – NoSQL database program

DB – Database

UI – User Interface

API – Application Programming Interface

JWT – JSON Web Token

CRUD – Create, Read, Update, Delete

1.5 Document Conventions

The document follows the IEEE 830 SRS structure.

Mandatory requirements are written in bold with “SHALL”.

All content is single-spaced with 1-inch margins; headings use consistent numbering.

1.6 References and Acknowledgments

IEEE Std 830-1998 — Recommended Practice for Software Requirements Specifications.

Project guidance by Ms. Fakhra Aftab, Department of CIS, NED University.

Chapter 2
Overall Description

2.1 Product Perspective

The Competitive Coding Arena is a standalone web-based system designed using the MERN stack. The client communicates with the backend server through GraphQL and WebSockets for real-time events. The server interacts with the MongoDB database to store user profiles, matches, and problem data.

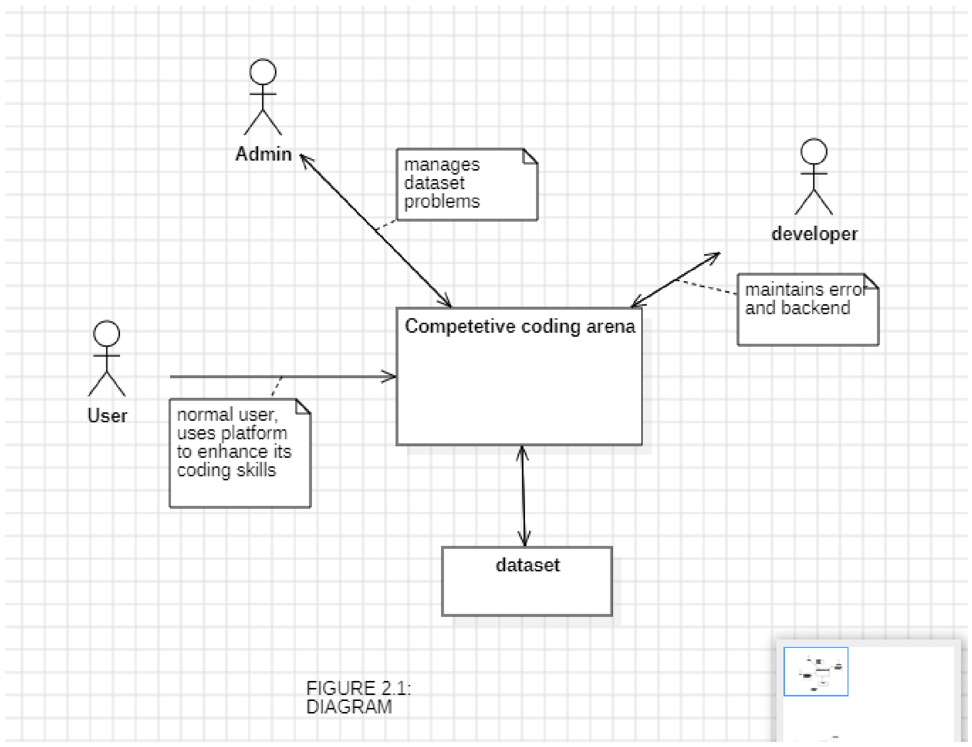

Figure 2.1: System Architecture Diagram

2.2 Product Functionality

User Authentication: Users can register or log in using their credentials. On successful login, a cookie stores the username for session tracking.

Leaderboard: Displays all users and their statistics such as points and ranks, sorted in descending order of points.

Profile Management: Users can view and edit their personal information such as username, email, and password.

Competition Mode: Enables real-time matches. When a user clicks “Start Competition,” the system searches for an available opponent. If found, a coding battle begins with a 10-minute timer.

Code Editor: Provides a built-in JavaScript editor where users can write, run, and submit solutions. Problems increase in difficulty as the user progresses.

Ranking System: Players start with a Rookie rank and level up when their points exceed 50.

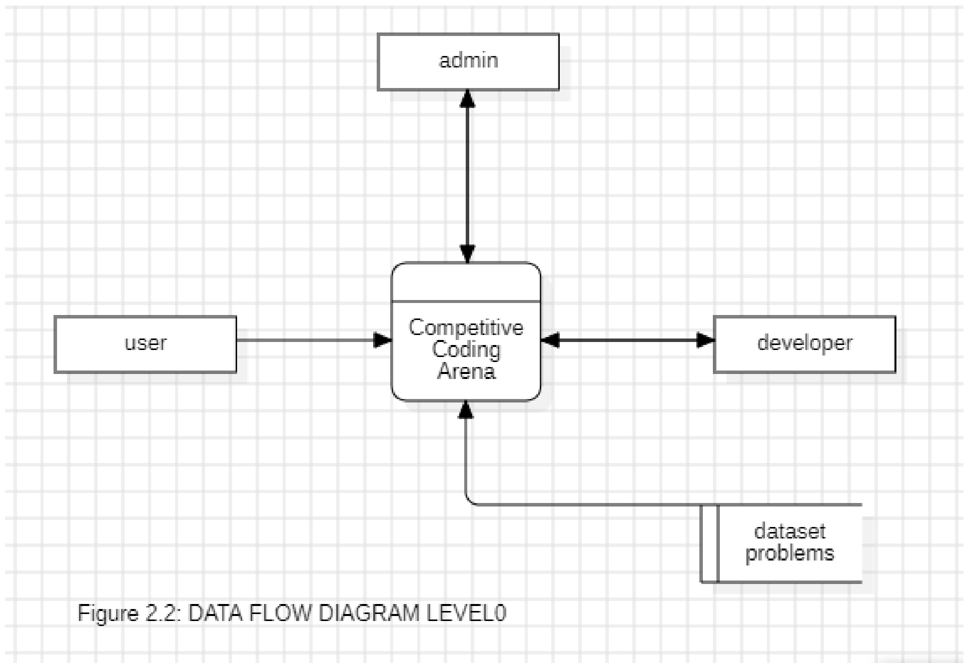

Figure 2.2: Functional Overview Diagram

2.3 Users and Characteristics

Registered User: Can log in, view leaderboard, start competitions, and edit profile.

Guest User: Limited access; can only view leaderboard.

Administrator: Manages coding problems, monitors matches, and handles user reports.

2.4 Operating Environment

Platform: Web-based; accessible via Chrome, Firefox, or Edge.

Server OS: Windows Server.

Database: MongoDB(NOSQL).

Network: Local or online deployment with stable internet connection.

2.5 Design and Implementation Constraints

Code execution limited to JavaScript only.

Competition timer fixed at 10 minutes.

One-on-one competition structure.

Internet connectivity required.

2.6 User Documentation

The final system will include a user manual, quick start guide, and FAQs on how to register, start competitions, and check rankings.

2.7 Assumptions and Dependencies

Users have stable internet access.

Browser supports JavaScript and WebSockets.

Server and database are hosted on a reliable platform.

WebSocket server will handle multiple concurrent users.

Chapter 3
Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Login Page: Authenticates users by role.

User Profile: Allows user to manage their profile.

Start Game: Allows user to design and start new challenges.

Leaderboard: Display rankings of all users based on their progress and results.

3.1.2 Hardware Interfaces

No specific hardware beyond a standard computer or web server is required.

3.1.3 Software Interfaces

Web Framework: Node.js

Database: MongoDB (NoSQL)

Operating System: Windows

3.1.4 Communications Interfaces

All communication uses HTTPS.

Qraqpql handle data exchange between client and server.

CSV imports/exports for external integration.

3.2 Functional Requirements

FR-1: The system shall allow users to register and log in using a unique username, email, and password.

FR-2: The system shall maintain user sessions using cookies for authentication.

FR-3: The system shall allow users to view a leaderboard sorted by total points.

FR-4: The system shall allow users to start a new competition and search for available opponents.

FR-5: The system shall begin a match when two players are paired.

FR-6: The system shall display a JavaScript-based code editor to each player.

FR-7: The system shall evaluate submitted code and update points for correct answers.

FR-8: The system shall update the leaderboard and player rank dynamically.

FR-9: The system shall provide the ability for users to edit their profile details.

FR-10: The system shall determine the winner based on points and difficulty of problems solved.

3.3 Behavior Requirements

After login, a user may view the leaderboard, start a competition, or edit their profile. In a match, the system matches two players, assigns identical problems, and evaluates solutions. The player with more points after 10 minutes wins. After login, the admin can manage the problem dataset by updating, adding, or removing problems.

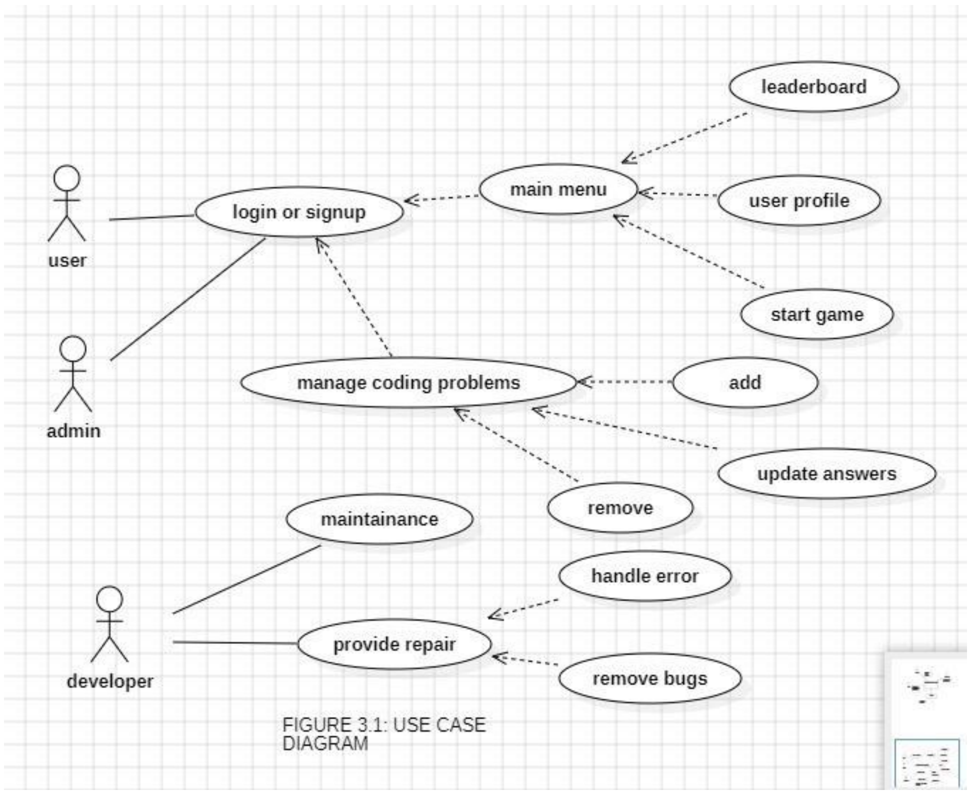

Figure 3.1: Behavior Flow Diagram

Chapter 4

Other Non-Functional Requirements

4.1 Performance Requirements

Matchmaking occurs within 5 seconds.

Code execution within 2 seconds per test.

Handle at least 100 concurrent users.

Leaderboard updates within 3 seconds.

Page load under 2 seconds.

4.2 Safety and Security Requirements

Secure HTTPS communication.

JWT-based authentication.

Input validation to prevent code injection.

Role-based data access for users and admins.

4.3 Software Quality Attributes

4.3.1 Reliability

High uptime using redundant backend services.

4.3.2 Usability

Intuitive interface with minimal navigation steps.

4.3.3 Maintainability

Modular React and Node.js structure.

4.3.4 Portability

The system can be deployed on any standard web server supporting the chosen stack.

4.3.5 Scalability

Cloud deployment for horizontal scaling.

Chapter 5
Other Requirements

FR-11: The system shall automatically log out users after 30 minutes of inactivity to maintain security.

FR-12: The system shall allow users to view their past competition history, including opponents, scores, and results.

FR-13: The system shall display live opponent activity (e.g., “opponent submitted code,” “opponent solved problem”) via WebSocket updates.

FR-14: The system shall store and display problem difficulty levels (Easy, Medium, Hard) with associated point values.

FR-15: The system shall allow admins to upload, update, or delete coding problems through a web interface.

FR-16: The system shall send a notification to the user when a match result is declared.

FR-17: The system shall allow users to reset their password via email verification.

FR-18: The system shall prevent duplicate logins (same account active on multiple de- vices).

Chapter A
AppendixA- Data Dictionary

Entity / Field	**Description**
user	Contains username, email, password, rank, and points.
match	Includes match_id, player1, player2, timer.
problem	Stores problem_id, difficulty, and title.
submission	Tracks submission_id, code, and verdict.

Chapter B
AppendixB- Database Tables

Users Table

Attribute	Data Type	Key	Description
username	STRING	Primary Key	User’s unique username.
displayname	STRING		Publicly displayed name of the user.
password	STRING		Hashed password of the user.
points	INT		Total points earned by the user.
rank	STRING		Rank of the user in the system.
session_token	BOOLEAN		Boolean value for logged in session.
total_matches	ARRAY		Total number of matches played by the user.

Challenges Table

Attribute	Data Type	Key	Description
id	INT	Primary Key	Unique identifier of the challenge.
problem_ Statement	STRING		Detailed description of the problem statement.
Function name	STRING		Name of the function that the user must implement.
test cases	OBJECT-ID		Test cases used to evaluate the user’s solution.
difficulty	STRING		Difficulty level of the challenge (e.g., easy, medium, hard).

Matches Table

Attribute	Data Type	Key	Description
id	INT	Primary Key	Unique identifier of the match.
Players-Info	OBJECT--ID		Usernames of all players in the match along with their points.
player _won_ Username	STRING	Foreign Key $\longrightarrow$ (Users.username)	Username of the player who won the match.
status	STRING		Current status of the match (e.g., completed, pending).
matchDate	DATETIME		Date and time when the match was played.
endTime	DATETIME		Time at which the match ended.