Software Requirements Specification

for

Online Lab Platform

Version 1.1.0 approved

Prepared by <Aaryan Agrawal, Kanishk Singh Dayma, Mayank Dahiya, Devash Saini, Dheeraj, Vyankatesh Deshpande >

<3-03-2025>

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
SRS v1.0.0	09-02-2025	base Version	
SRS v1.1.0	03-03-2025	updated version	

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to define the requirements for an Online Coding Lab platform. The platform will provide a collaborative environment for coding practice, problem-solving, and virtual lab sessions for students and instructors.

1.2 Document Conventions

This document follows the IEEE SRS template. Priorities are defined as follows:

- High (H): Critical features for the system's operation.
- Medium (M): Important but not essential features.
- Low (L): Optional features for enhanced functionality.

1.3 Intended Audience and Reading Suggestions

The intended audience includes:

- **Developers**: For understanding system features and constraints.
- **Project Managers**: For planning and resource allocation.
- **Testers**: For designing test cases.
- **Users (Students and Instructors)**: For understanding system capabilities

1.4 Product Scope

The Online Coding Lab will facilitate virtual coding sessions, support multiple programming languages, and provide features for code compilation, submission, and assessment. The platform aims to enhance the learning experience by fostering problem solving.

1.5 References

- Class Notes
- https://ieeexplore.ieee.org/document/278253

2. Overall Description

2.1 Product Perspective

The proposed online lab platform serves as a modern, customized replacement for Moodle, which is currently utilized in IIT Jodhpur. Moodle is a robust, widely adopted learning management system (LMS) used for handling courses, assignments, and communication between students and educators. However, certain limitations such as its complexity, lack of flexibility for lab-specific tasks, and cumbersome interface may necessitate a tailored solution.

2.2 Product Functions

- **User Management**
  - Register and manage user roles: students, instructors, and administrators
  - Provide secure login and role-based access control
  - Enable profile management
- **Lab Participation and Submissions**
  - Allow students to access assigned labs and perform lab tasks
  - Facilitate submission of lab assignments with file upload support
- **Communication and Collaboration Tools**
  - Support announcements, and messaging
- **Lab Content Creation and Management**
  - Allow instructors to create, edit, and schedule lab activities
  - Organize labs by courses and categories

2.3 User Classes and Characteristics

1. Student Users (Primary User Class)

- **Frequency of Use:** Frequent, especially during active lab sessions.
- **Functions Used:**
  - Access and participate in lab session
  - Submit assignments
  - View grades and feedback
  - Communicate with instructors
- **Technical Expertise:** Basic familiarity with web applications
- **Security Level:** Limited access, restricted to their assigned courses and submissions

2. Instructor Users (Primary User Class)

- **Frequency of Use:** Moderate to frequent, depending on course requirements

- **Functions Used:**
    - Create and schedule lab sessions
    - Manage lab content and assignments
    - Grade submissions and provide feedback
    - Communicate with students via announcements or discussion boards
- **Security Level:** Higher privileges to manage courses and student evaluations

2.4 Operating Environment

The platform will be cloud-hosted (university servers). Users can access it via modern browsers on Windows, macOS, Linux, and mobile devices. The backend is built with Node.js, using MongoDB, and the frontend is developed with React.js.

2.5 Design and Implementation Constraints

A large number of users logging simultaneously may crash the platform (A major drawback that still exists in Prutor today)

2.6 User Documentation

The platform will include:

- A user guide explaining navigation, course selection, and coding exercises.
- In-platform tooltips and FAQs for quick guidance.
- Documentation will be available online in PDF.

2.7 Assumptions and Dependencies

- The platform assumes stable internet access for cloud hosting.
- Future updates might require additional server resources as user load increases.

3. External Interface Requirements

3.1 User Interfaces

The Online Coding Lab platform will provide a structured and intuitive user interface catering to students, instructors, and administrators. The key UI components include:

- **Dashboard**:
    - Displays available **courses**, enrolled courses, assignments.
    - Quick access to coding challenges.
- **Course Selection Page**:
    - Lists all available courses with descriptions and enrollment options.

- **Assignment & Coding Interface**:
    - **Problem Statement Panel**: Displays question descriptions, constraints, and examples.
    - **Code Editor**:
        - Supports multiple programming languages.
        - Provides features like error detection and debugging.
    - **Submission Panel**:
        - Displays test case results, execution time, and memory usage.
        - Allows re-submission.
- **Navigation Menu**:
    - Access to dashboard, courses and assignments.
- **Common UI Elements**:
    - **Standard Buttons**: Submit, Reset, Run, Enroll, and Help.
    - **Keyboard Shortcuts**: Quick actions for running code, saving progress, and navigating sections.
    - **Dark/Light Mode**: User-selectable themes for enhanced readability.

3.2 Hardware Interfaces

- **Device Compatibility**: Accessible via web browsers on **desktops, laptops and tablets**.
- **Server Requirements**:
    - Cloud-based or on-premise infrastructure with multi-core processors and scalable storage (eg. IITJ servers).
    - Secure environment for storing user data and course materials.
- **Supported Input Devices**:
    - Keyboard and mouse for text input and navigation.
    - Touchscreen support for tablet users.

3.3 Software Interfaces

- **Operating System Compatibility**: Works on Windows, macOS, and Linux.
- **Database**:
    - **MongoDB** for managing user data, course materials, assignments, and submissions.
- **Backend Framework**:
    - **Express.js (Node.js)** for API handling, user authentication, and database interaction.
- **Frontend Framework**:
    - **React.js** for building the interactive user interface.
- **Code Execution Engine**:
    - Supports languages like Python, C++, Java, and JavaScript.
- **External APIs**:

- **Authentication**: OAuth or Firebase Authentication for secure login.
- **Compilation & Execution**: Jdoodle's API for compilation and execution.
- **Cloud Storage**: MongoBD for storing course files and user submissions.

3.4 Communications Interfaces

- **HTTP/HTTPS Protocol**:
  - RESTful APIs for frontend-backend interactions.
  - WebSocket for real-time updates and collaboration features.
- **Authentication & Security**:
  - JWT-based authentication for secure user sessions.
- **Networking Requirements**:
  - Optimized for **low-latency** requests to provide quick execution feedback.

4. System Features

4.1 Classroom Management

Description and Priority

This feature allows instructors to create, manage, and delete virtual classrooms. Students can join classrooms using a unique code provided by the instructor. Each classroom contains coding assignments and evaluation tools.

Priority: **High**

- **Benefit:** 9
- **Penalty:** 8 (Without this, the platform is unusable)
- **Cost:** 5 (Moderate implementation complexity)
- **Risk:** 3 (Some risk due to user management and security concerns)

4.1.1 Stimulus/Response Sequences

User Action	System Response
Instructor logs in	System verifies credentials and redirects to the instructor dashboard
Instructor clicks "Create Classroom"	System displays a form for classroom details
Instructor fills out details and submits	System creates the classroom and generates a unique join code

Student logs in	System verifies credentials and redirects to the student dashboard
Instructor deletes a classroom	System prompts for confirmation and then removes the classroom along with associated assignments

4.1.2 Functional Requirements

- **REQ-1:** The system shall allow instructors to **create classrooms** with a unique name and description.
- **REQ-2:** The system shall generate a unique classroom code that students can use to join.
- **REQ-3:** The system shall allow students to join a classroom by entering a valid classroom code.
- **REQ-4:** The system shall display a **list of enrolled students** to the instructor.
- **REQ-5:** The system shall allow instructors to **delete classrooms**, **removing all associated assignments** and submissions.
- **REQ-6:** The system shall prevent students from joining a classroom if the code is invalid or the classroom is closed.

4.2 Assignment Management

4.2.1 Description and Priority

Instructors can **create**, **assign**, and **manage** coding exercises within their classrooms. Students can **view assignments**, **submit their solutions.**

Priority**: High**

- **Benefit:** 9 (Core feature for learning and assessment)
- **Penalty:** 8 (Without this, students cannot complete coursework)
- **Cost:** 6 (Requires backend logic for managing assignments)
- **Risk:** 4 (Potential complexity in grading and evaluations)

4.2.2 Stimulus/Response Sequences

User Action	System Response
Instructor clicks "Create Assignment"	System displays a form for entering assignment details

Instructor fills out details and submits	System saves the assignment and notifies students
Student views assignment list	System displays assignments for the enrolled classroom
Student clicks on an assignment	System displays the assignment details and a code editor
Student submits solution	System validates, compiles, and runs the code, providing real-time feedback
Instructor manually grades an assignment	System updates the student's score and feedback

4.2.3 Functional Requirements

REQ-7: The system shall allow instructors to **create assignments** with problem statements and sample test cases.
REQ-8: The system shall allow students to **submit solutions to assignments**.
REQ-9: The system shall support real-time code compilation and execution for student submissions.
REQ-10: The system shall provide **auto-grading** based on predefined test cases.
REQ-11: The system shall allow instructors to manually override auto-graded scores.
REQ-12: The system shall provide students with feedback on their submissions.
REQ-13: The system shall allow instructors to **view student submission history**.

4.3 Code Compilation and Execution

4.3.1 Description and Priority

Students can **write**, **compile**, and **run code** in a secure, browser-based environment. The platform supports multiple programming languages and provides real-time execution results.

Priority**: High**

- **Benefit:** 10 (Key feature enabling coding practice)
- **Penalty:** 9 (Students cannot test solutions without it)
- **Cost:** 7 (Requires server-side execution logic)
- **Risk:** 5 (Security concerns with executing user-submitted code)

4.3.2 Stimulus/Response Sequences

User Action	System Response
Student selects a programming language	System loads the appropriate environment
Student writes code in the editor	System provides syntax highlighting and autocomplete
Student clicks "Run"	System compiles and executes the code in a sandboxed environment
System detects an error in code	System displays error messages with debugging tips
Student clicks "Submit"	System validates the code against test cases and stores the submission

4.3.3 Functional Requirements

REQ-15: The system shall **support multiple programming languages** (e.g., Python, Java, C++).
REQ-16: The system shall allow students to **write** and **execute code** in an integrated editor.
REQ-17: The system shall provide **real-time compilation** and **execution** results.
REQ-18: The system shall display **error messages** and **debugging information** if the code fails.
REQ-19: The system shall provide an option to save submissions.

4.4 Grading and Feedback

4.4.1 Description and Priority

The platform enables **automatic** and **manual grading** of student submissions. Instructors can provide feedback, and students can track their progress.

Priority: Medium

- **Benefit:** 8 (Improves learning experience)
- **Penalty:** 7 (Students rely on feedback for improvement)
- **Cost:** 5 (Requires grading logic and UI integration)
- **Risk:** 3 (Low technical complexity)

4.4.2 Stimulus/Response Sequences

User Action	System Response
Student submits an assignment	System runs test cases and auto-grades the submission
Instructor manually reviews submission	System allows grading overrides and provides comment options
Student views graded assignment	System displays score, feedback, and test case results
Student requests re-evaluation	System notifies the instructor for review

4.4.3 Functional Requirements

REQ-20: The system shall **automatically grade student submissions** based on test cases.
REQ-21: The system shall allow instructors to **manually override grades** and provide feedback.
REQ-22: The system shall display **student scores.**

4.5 Security and Authentication

4.5.1 Description and Priority

The system ensures that users access only their assigned roles and protects submitted code and grades from unauthorized access.

Priority**: High**

4.5.2 Functional Requirements

REQ-24: The system shall require users to **authenticate before accessing the platform**.
REQ-25: The system shall support **role-based access control** (Instructor, Student, Admin).

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The system shall meet the following performance criteria:

a) **Response Time**

- Page load time shall not exceed 2 seconds under normal conditions
- Code compilation and execution results shall be returned within 3 seconds for basic programs
- Maximum response time for any operation shall not exceed 5 seconds
- System shall support concurrent usage by at least 200 users without performance degradation

b) **Scalability**

- System shall maintain performance levels while handling up to 1000 simultaneous users
- Database shall support storage of up to 10,000 code submissions per day
- Platform shall support addition of new programming languages without system redesign

c) **Availability**

- System uptime shall be maintained at 99.9% during academic hours (8 AM - 8 PM)
- Scheduled maintenance shall only occur during off-peak hours
- System shall include automatic failover capabilities for critical components

5.2 Safety Requirements

a) **Data Protection**

- Regular automated backups of all user data and submissions
- Backup retention features shall be there.

b) **Error Prevention**

- System shall prevent execution of potentially harmful code
- Resource limits shall be enforced for all code execution
- System shall maintain isolation between different users' code execution environments

5.3 Security Requirements

a) Authentication and Authorization

- Multi-factor authentication for administrative access
- Role-bas
- ed access control (RBAC) implementation
- Password requirements: minimum 8 characters, including uppercase, lowercase,

numbers, and special characters
- Session timeout after 30 minutes of inactivity

b) Data Security

- Regular security audits shall be conducted every 6 months
- Compliance with educational data protection regulations

c) System Security

- Regular vulnerability scanning and penetration testing

5.4 Software Quality Attributes

a) Usability

- System shall support multiple languages (internationalization)
- Navigation shall require no more than 5 clicks to reach any feature
- Consistent UI/UX patterns across all pages

b) Maintainability

- Modular architecture for easy component updates
- Comprehensive API documentation
- Clear separation of concerns in code architecture
- Automated testing coverage of at least 80%

c) Reliability

- Mean Time Between Failures (MTBF): TBD
- Mean Time To Repair (MTTR): TBD
- Automated error logging and monitoring
- Graceful degradation of non-critical features during high load

5.5 Business Rules

a) Academic Integrity

- All code submissions shall be timestamped
- System shall maintain audit logs of all grading actions
- Plagiarism detection tools shall be integrated
- Historical submission records shall be maintained for the duration of the course

b) Administrative Control

- Only authorized instructors can create and modify assignments
- Department heads shall have oversight access to all courses in their department
- Teaching assistants shall have limited grading privileges as assigned by instructors
- System administrators shall not have access to modify grades or academic content

c) Compliance

- System shall adhere to academic institution's data retention policies

- All actions shall be logged for audit purposes
- System shall support export of data in standard formats
- Compliance with accessibility requirements for educational software

6. Other Requirements

6.1 Database Requirements

- **NoSQL Database (MongoDB/Firebase)**: Used for unstructured data like **real-time notifications, logs, and chat messages** and structured data like **users, courses, assignments, and submissions**..
- **Performance Requirements**: Queries should return results in **less time**, even under high load.
- **Automatic Backups**: Daily backups with **a 30-day retention period** to ensure data safety.
- **ACID Compliance**: All transactions must ensure **Atomicity, Consistency, Isolation, and Durability**..

6.2 Legal and Compliance Requirements

- **Academic Integrity Enforcement**: Implement **plagiarism detection** for assignment submissions.

6.3 Reuse and Extensibility Objectives

- **Multi-Language Coding Support**: The coding environment should support **Python, Java, C, C++** .

Appendix A: Glossary

SRS – Software Requirements Specification

RBAC – Role-Based Access Control

LMS – Learning Management System

ACID – Atomicity, Consistency, Isolation, Durability (database principles)

IDE – Integrated Development Environment (for coding assignments)

Appendix B: Analysis Models

- **Class Diagram** – Represents **users, courses, assignments, and submissions**.
- **Use Case Diagrams** – Shows interactions of **students, instructors, and admins** with

the platform.

- **State Transition Diagram** – Defines states like **course enrollment, assignment submission, auto-grading**.

Appendix C: To Be Determined List

1. **Exact list of supported programming languages** in the coding module.
2. **Storage Limits**: Maximum **file size for submissions and total user storage allocation**.
3. **Grading Policies**: Criteria for **automatic** of coding and non-coding assignments.