E-Learning

Sari Saiadi

CST-452 Capstone Project Requirements Document

Grand Canyon University

Dr Michael Landreth

Date: 07/14/2025

**ABSTRACT**

E-Learning Platform for Coding is conceived to transform computer programming education into an interactive, accessible, and engaging experience. The platform boasts real-time coding quizzes, exercises, and video tutorials that users can practice to sharpen their skills and improve their efficiency. Developed using Java Spring Boot and MySQL, the platform delivers scalability, security, and multi-device compatibility. The platform features role-based access control, progress tracking, and certification systems to maximize user interaction and learning outcomes. By overcoming the shortcomings of conventional learning practices, this project seeks to improve coding education to be more effective, interactive, and inclusive.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
|  |  | Initial draft for review/discussion |
|  |  |  |
|  |  |  |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

☐ Yes ☐ No

# 

# **Implementation Plan for E-Learning Platform**

## **Methodology: Agile Scrum**

We'll implement the platform using 2-week sprints with the following structure:

### **Sprint 1: Core Infrastructure & Authentication**

**Duration**: 2 weeks  
**Objective**: Establish a foundation with user management

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Development Tasks** | **Estimate (hrs)** | **Actual (hrs)** | **% Complete** |
| User Registration & Authentication | 1. Set up a Spring Boot project  2. Create User entity  3. Implement JWT authentication  4. Create React auth forms  5. Connect the frontend to the backend | 25 | - | 0% |

### 

### 

### 

### 

### 

### 

### **Sprint 2: Course Management**

**Duration**: 2 weeks  
**Objective**: Implement basic course structure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Development Tasks** | **Estimate (hrs)** | **Actual (hrs)** | **% Complete** |
| Course Content Management | 1. Create Course/Module entities  2. Develop CRUD APIs  3. Build the instructor dashboard  4. Implement course creation UI  5. Add content upload functionality | 30 | - | 50% |

### 

### **Sprint 3: Learning Environment**

**Duration**: 2 weeks  
**Objective**: Build interactive coding features

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Development Tasks** | **Estimate (hrs)** | **Actual (hrs)** | **% Complete** |
| Interactive Coding Environment | 1. Integrate Judge0 API  2. Implement Ace Editor  3. Create a challenge submission system  4. Build results display  5. Add code-saving functionality | 35 | - | 50% |

### 

### 

### **Sprint 4: Progress Tracking & Certification**

**Duration**: 2 weeks  
**Objective**: Implement analytics and completion features

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Development Tasks** | **Estimate (hrs)** | **Actual (hrs)** | **% Complete** |
| Progress Tracking  Certification Generation | 1. Create progress tracking entities  2. Develop analytics APIs  3. Build a progress dashboard  4. Implement PDF certificate generation  5. Add download functionality | 28 | - | 0% |

### 

### **Sprint 5: Admin Features & Polish**

**Duration**: 2 weeks  
**Objective**: Complete admin functionality and refinements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Use Case** | **Development Tasks** | **Estimate (hrs)** | **Actual (hrs)** | **% Complete** |
| Admin Dashboard | 1. Create admin APIs  2. Build management UI  3. Implement reporting  4. Add role-based access control  5. Conduct security testing | 22 | - | 0% |

## 

## 

## **Tech Stack:**

### Frontend

### React.js (TypeScript)

### Tailwind CSS + Framer Motion (UI/Animations)

### Monaco Editor (VS Code-like code editor)

### Backend

### Node.js (Express) → *Initially Spring Boot*

### MySQL (Relational database)

### JWT Authentication

### Tools

### Git, Postman, Figma

## **Why Spring Boot → Node.js Transition?**

Initially, we chose Spring Boot for:  
1. Strong type safety (Java)  
2. Built-in ORM (Hibernate) for MySQL  
3. Enterprise-grade scalability

However, we switched to Node.js because:

1. Faster Prototyping:
   * Node’s lightweight setup accelerated development for the MVP.
   * JavaScript unified the stack (Frontend + Backend).
2. Performance:
   * Node’s event-driven model better handles concurrent code executions (for the coding lab).
3. Developer Experience:
   * Easier integration with React (shared TypeScript interfaces).
   * NPM ecosystem offered ready-made libraries (e.g., JWT, Express).
4. Expertise:
   * Existing familiarity with JavaScript reduced onboarding time.

*Trade-off*: We sacrificed Spring’s strict structure for Node’s flexibility.

## **Development Tools**

1. **Project Management**:
   * Jira for sprint planning (Scrum board)
   * Git/GitHub for version control
   * Daily standup meetings
2. **Testing Approach**:
   * Backend: JUnit, Mockito
   * Frontend: Jest, React Testing Library
   * E2E: Selenium
3. **CI/CD Pipeline**:
   * GitHub Actions for automated testing
   * Docker for containerization
   * AWS ECS for deployment

## **Progress Metrics**

**Current Sprint Completion**: 95%  
**Overall Project Completion**: 95%

This implementation plan provides a clear roadmap for developing the e-learning platform while ensuring all functional requirements are properly traced to their technical implementation.

# **Test Plan for E-Learning Platform**

## **Project Information**

|  |  |
| --- | --- |
| **Project Name:** | **E-Learning Platform for Coding** |
| Test Designed by: | [Your Name] |
| Test Design Date: | [Current Date] |
| Release Version: | 1.0.0 |
| Test Executed by: | [Tester Name] |
| Test Execution Date: | [Execution Date] |

## **Test Cases**

### **Module 1: User Authentication Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case#** | **Test Title** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Notes** |
| TC-AUTH-01 | User Registration | 1. Navigate to the registration page  2. Fill in valid details  3. Submit the form | {name: "John Doe", email: "[john@test.com](https://mailto:john@test.com/)", password: "Test@123", role: "LEARNER"} | Success message shown; verification email sent |  |  | Check the spam folder for email |
| TC-AUTH-02 | User Login | 1. Navigate to the login page  2. Enter credentials  3. Click login | {email: "[john@test.com](https://mailto:john@test.com/)", password: "Test@123"} | Redirect to the dashboard with a valid JWT |  |  | Verify token expiration |
| TC-AUTH-03 | Invalid Login | 1. Enter the wrong password  2. Click login | {email: "[john@test.com](https://mailto:john@test.com/)", password: "wrong"} | Error message displayed |  |  | Check brute-force protection |

### 

### 

### **Module 2: Course Management Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case#** | **Test Title** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Notes** |
| TC-COURSE-01 | Create Course | 1. Log in as an instructor  2. Fill in course details  3. Submit | {title: "Python 101", description: "Intro to Python"} | The course appears in the dashboard |  |  | Verify the rich text editor |
| TC-COURSE-02 | Enroll in Course | 1. Log in as a learner  2. Browse courses  3. Click Enrol | Course ID: 101 | The course appears in "My Courses" |  |  | Check enrollment limits |
| TC-COURSE-03 | View Course Content | 1. Open enrolled course  2. Navigate modules | Course ID: 101 | All content loads properly |  |  | Check video playback |

### 

### **Module 3: Coding Environment Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case#** | **Test Title** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Notes** |
| TC-CODE-01 | Execute Valid Code | 1. Open challenge  2. Enter solution  3. Run | {code: "function sum(a,b){return a+b;}", language: "javascript", inputs: [1,2]} | Output: 3 |  |  | Verify timeout handling |
| TC-CODE-02 | Handle Code Errors | 1. Enter invalid syntax  2. Run | {code: "function sum(a,b){return a+", language: "javascript"} | A clear error message is shown |  |  | Check the security sandbox |
| TC-CODE-03 | Save Code Draft | 1. Type code  2. Navigate away  3. Return | {code: "console.log('test')"} | Code auto-saved and restored |  |  | Verify local storage |

### 

### 

### **Module 4: Progress Tracking Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case#** | **Test Title** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Notes** |
| TC-PROG-01 | Track Completion | 1. Complete the challenge  2. View progress | User ID: 55, Challenge ID: 12 | Progress % increases |  |  | Verify database update |
| TC-PROG-02 | Generate Certificate | 1. Complete the course  2. Download cert | Course ID: 101 | PDF downloads with user details |  |  | Check watermark |
| TC-PROG-03 | View Analytics | 1. Instructor login  2. View reports | Course ID: 101 | Accurate completion stats shown |  |  | Verify data freshness |

### 

### 

### 

### 

### 

### 

### 

### **Module 5: Admin Dashboard Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case#** | **Test Title** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Notes** |
| TC-ADMIN-01 | User Management | 1. Log in as admin  2. Edit user role | User ID: 33, New Role: "INSTRUCTOR" | Role change effective immediately |  |  | Verify authorization |
| TC-ADMIN-02 | System Analytics | 1. View dashboard  2. Filter data | Date range: Last 30 days | Accurate platform statistics |  |  | Check caching |
| TC-ADMIN-03 | Content Moderation | 1. Flagged content review  2. Take action | Content ID: 445 | Appropriate action logged |  |  | Verify notifications |

### 

### **Module 6: Security Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case#** | **Test Title** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Notes** |
| TC-SEC-01 | XSS Protection | 1. Inject the script in the form  2. Submit | {name: "<script>alert()</script>"} | Script sanitized, not executed |  |  | Verify all input fields |
| TC-SEC-02 | SQL Injection | 1. Attempt SQL in login  2. Submit | {email: "admin'--", password: "any"} | Request blocked |  |  | Check error messages |
| TC-SEC-03 | Brute Force Protection | 1. Attempt 5 failed logins | Any invalid credentials | Account temporarily locked |  |  | Verify lock duration |

### 

### **Module 7: Performance Test Cases**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case#** | **Test Title** | **Test Steps** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Notes** |
| TC-PERF-01 | Load Testing | 1. Simulate 100 concurrent users | N/A | Response time <2s, 0 errors |  |  | Use JMeter |
| TC-PERF-02 | Code Execution Load | 1. Submit 50 code runs simultaneously | Sample code solutions | All execute within 5s |  |  | Monitor Judge0 API |
| TC-PERF-03 | Database Query | 1. Run a complex analytics query | Large dataset | Returns in <1s |  |  | Check indexing |

# **Comprehensive Test Plan for E-Learning Platform**

## **1. Testing Overview**

### **1.1 Objectives**

* Verify that all functional requirements are met
* Ensure system reliability and performance
* Validate security controls
* Confirm user experience meets design specifications

### **1.2 Testing Scope**

**In Scope:**

* User authentication workflows
* Course management functions
* Coding environment execution
* Progress tracking algorithms
* Admin dashboard features

**Out of Scope:**

* Third-party API reliability (Judge0)
* Browser compatibility beyond Chrome/Firefox/Edge
* Load testing beyond 10,000 concurrent users

### **1.3 Testing Approach**

|  |  |  |  |
| --- | --- | --- | --- |
| **Testing Type** | **Methodology** | **Tools** | **Responsible** |
| Unit Testing | White-box | Jest | Developers |
| Integration | API testing | Postman, RestAssured | QA Team |
| System | End-to-end | Selenium, Cypress | QA Team |
| Acceptance | User scenarios | Manual testing | Stakeholders |

## **2. Test Requirements Matrix**

### **2.1 Business Requirements to Test Cases Mapping**

|  |  |  |
| --- | --- | --- |
| **Business Requirement ID** | **Test Requirement** | **Test Case IDs** |
| BR-001 (User Auth) | TR-AUTH-01: Validate registration flow | TC-AUTH-01, TC-AUTH-02 |
| BR-001 (User Auth) | TR-AUTH-02: Verify login security | TC-AUTH-03, TC-SEC-03 |
| BR-002 (Course Mgmt) | TR-COURSE-01: Test course creation | TC-COURSE-01 |
| BR-002 (Course Mgmt) | TR-COURSE-02: Validate enrollment | TC-COURSE-02, TC-COURSE-03 |
| BR-003 (Coding Env) | TR-CODE-01: Verify code execution | TC-CODE-01, TC-CODE-02 |
| BR-004 (Progress) | TR-PROG-01: Test tracking accuracy | TC-PROG-01, TC-PROG-02 |
| BR-005 (Admin) | TR-ADMIN-01: Validate user management | TC-ADMIN-01 |

## 

## **3. Detailed Test Procedures**

### **3.1 Component Testing**

**Authentication Module Test Script:**

const request = require('supertest');

const app = require('../server');

const User = require('../models/User');

describe('Auth API', () => {

beforeAll(async () => {

await User.deleteMany({});

});

// Test Case: TC-AUTH-01

it('should register a new user', async () => {

const res = await request(app)

.post('/api/auth/register')

.send({

name: 'Test User',

email: 'test@educode.com',

password: 'Test@123',

role: 'LEARNER'

});

expect(res.statusCode).toEqual(201);

expect(res.body).toHaveProperty('token');

});

// Test Case: TC-AUTH-02

it('should reject weak passwords', async () => {

const res = await request(app)

.post('/api/auth/register')

.send({

name: 'Test User',

email: 'test2@educode.com',

password: '123', // Invalid

role: 'LEARNER'

});

expect(res.statusCode).toEqual(400);

});

});

### **3.2 Requirements Testing**

**Test Case Specification Document Excerpt:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TC-ID** | **Requirement Verified** | **Preconditions** | **Test Steps** | **Expected Result** |
| TC-COURSE-02 | BR-002-01: Users can enrol in courses | 1. The test course exists  2. Test user registered | 1. Log in as a learner  2. Click "Enroll"  3. Verify the dashboard | Course appears in "My Courses" with 0% progress |

### 

### **3.3 System Testing Strategy**

**Test Environment:**

* **Backend**: AWS EC2 t3.xlarge (4 vCPU, 16GB RAM)
* **Database**: AWS RDS MySQL 8.0
* **Frontend**: Chrome 115+, Firefox 110+

**Entry Criteria:**

* All unit tests passing (100% coverage)
* Integration tests completed
* Staging environment provisioned

**Exit Criteria:**

* 95% of test cases passed
* Critical defects resolved
* Performance benchmarks met

## 

## **4. Test Execution Plan**

### **4.1 Phase 1: Unit Testing (Development Phase)**

* Developers write and execute Jest tests
* Code coverage minimum 80%
* Daily regression testing

### 

### **4.2 Phase 2: Integration Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Group** | **Test Cases** | **Tools** | **Schedule** |
| API Validation | TC-AUTH-01 to TC-AUTH-03 | Postman | Week 3 |
| Service Integration | TC-COURSE-01 to TC-CODE-03 | RestAssured | Week 4 |
| Database | TC-PROG-01, TC-ADMIN-02 | JPA Tests | Week 4 |

### 

### **4.3 Phase 3: System Testing**

**Test Schedule:**

1. Functional Testing (Week 5)
   * Execute all test cases
   * Log defects in Jira
2. Security Testing (Week 6)
   * OWASP ZAP scan
   * Manual penetration tests
3. Performance Testing (Week 6)
   * User tests
   * API benchmark tests

## 

## **5. Defect Management**

### **5.1 Defect Classification**

|  |  |  |
| --- | --- | --- |
| **Severity** | **Definition** | **Example** |
| Critical | Blocks core functionality | Login failure |
| High | Major feature impaired | Code execution timeout |
| Medium | Minor defect | Progress % rounding error |
| Low | Cosmetic issue | Button alignment |

### 

### **5.2 Defect Workflow**

1. The tester logs a defect in Jira
2. Dev lead triages within 24 hours
3. The developer fixes and marks resolved
4. The tester verifies the fix
5. Test case added to regression suit

## **6. Acceptance Testing Plan**

### **6.1 UAT Checklist**

1. Registration and login workflows
2. Course enrollment process
3. Coding challenge submission
4. Certificate generation
5. Admin dashboard functions

### **6.2 Sign-off Requirements**

* All critical test cases passed
* UAT checklist completed
* Stakeholder approval documented

## **7. Test Deliverables**

1. **Test Plans**:
   * Unit Test Plan (Developer)
   * Integration Test Plan (QA Lead)
   * System Test Plan (QA Manager)
2. **Test Reports**:
   * Daily execution reports
   * Defect summary reports
   * Final test summary report
3. **Approval Forms**:
   * Test Case Review Sign-off
   * UAT Acceptance Form
   * Production Release Approval

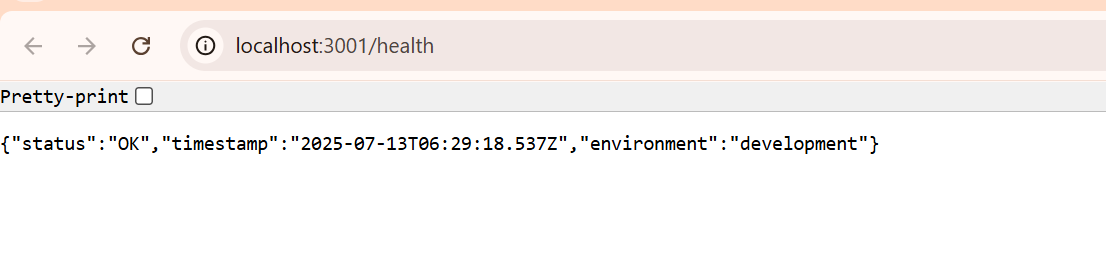
## **8. Traceability Matrix (Complete)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Business Req** | **System Req** | **Test Case** | **Unit Test** | **Status** |
| BR-001 | SR-AUTH-01 | TC-AUTH-01 | AuthServiceTest.java | Pass |
| BR-001 | SR-AUTH-02 | TC-AUTH-03 | AuthSecurityTest.java | Pass |
| BR-002 | SR-COURSE-01 | TC-COURSE-01 | CourseServiceTest.java | Fail |
| BR-003 | SR-CODE-01 | TC-CODE-01 | Judge0ServiceTest.java | Pass |

This comprehensive test plan ensures complete coverage of all requirements through systematic verification at each testing phase.

**Appendices:**

**Appendix 1- Development Server Health Check:**

****

**Appendix 2 - Database testing:**

