Test Plan
Software Quality Engineering

Presented By

Muhammad Waqas (22I - 2469)

Abdul Raffay (22I - 2546)

Abdullah Mansoor (22I- 8808)

To

Mam Uzma

Table of Contents

Test Plan: Campus Connect

IEEE 829 Format

1. Test Plan Identifier

- TP-USS-2024-01
- Version: 1.0
- Last Updated: December 9, 2024

2. References

- Campus Connect Vision and Requirements Documentation, Version 2.3 (Last updated: June 15, 2024)
- Spring Boot Application Source Code (SDA_Project, Repository: Github, Branch: main, Commit: uploaded porject, as of June 20, 2024)
- IEEE 829-2008 Test Plan Standard
- Software Quality Engineering Course Requirements, dated December 2024

3. Introduction

This test plan outlines the testing strategy and approach for the System test plan of University Social System, a comprehensive web-based platform that facilitates social interactions and academic management within a university environment. The system includes various modules such as user authentication, course management, multimedia handling, and administrative functions. Testing activities will be carried out within the allocated QA team resources and budget constraints as per the Software Project Plan.

This test plan complements the ongoing review processes, including code reviews and design inspections. Changes to this plan or to the test schedule will follow the established project change control procedures.

Scope

This plan encompasses testing of all major components including:

- User Authentication and Authorization
- Course Management System
- Social Interaction Features

- Content Management (Audio, Video, Books)
- Administrative Functions
- Event Management
- Student-Teacher Interactions

4. Test Items

The Enrollment Management module must be tested and validated by December 8, 2024, to align with the university's enrollment period.

The following components will be tested:

1. **Core System Components:**
   - User Authentication Module
   - Admin Management System
   - Course Catalog System
   - Enrollment Management
   - Event Management System

2. **Controllers:**
   - AdminController
   - AudioController
   - BookController
   - CourseCatalogController
   - CourseController
   - EnrollmentController
   - EventController
   - IndexController
   - LessonController
   - ModuleController
   - NewsController
   - NotificationController
   - PhotoController
   - PostController
   - QuizController
   - ScheduleController
   - StudentController
   - TeacherController
   - VideoController

3. **Database Interactions**
4. **API Endpoints**
5. **User Interface Components**

5. Software Risk Issues

High-Risk Areas:

1. **Data Security and Privacy**
   - User authentication and authorization
   - Personal information protection
   - Academic records integrity

2. **System Performance**
   - Concurrent user access
   - Large media file handling
   - Real-time notification delivery

3. **Integration Points**
   - Multiple controller interactions
   - Third-party service dependencies
   - Database consistency

4. **Technical Complexity**
   - Multi-user session management
   - Real-time updates
   - File upload/download operations

 Data Security and Privacy:
Ensuring compliance with privacy laws, robust user authentication, and protection of sensitive academic records.

- **System Performance:**
Validating scalability under heavy concurrent usage, efficient handling of large multimedia files, and timely real-time notifications.

- **Integration Points:**
Maintaining stable interactions between multiple controllers, ensuring reliability with third-party services, and preserving database integrity.

- **Technical Complexity:**
Managing multi-user sessions, supporting real-time data updates, and enabling secure, efficient file upload/download operations.

6. Features to be Tested

1. **User Authentication and Authorization**
   - Login/Logout functionality (H)
   - Role-based access control (H)
   - Password management (H)
   - Session handling (H)
2. **Course Management**
   - Course creation and modification (H)
   - Enrollment processing (H)
   - Course material distribution (M)
   - Grade management (M)
3. **Content Management**
   - File upload/download (M)
   - Media streaming (M)
   - Document management (M)
   - Content accessibility (H)
4. **Social Features**
   - User interactions (M)
   - Event management (M)

- Notifications (M)
- Message handling (L)

5. **Administrative Functions**
    - User management (H)
    - System configuration (H)
    - Report generation (M)
    - Access control management (H)

Explanation:

The features are described in terms of what the user wants to do (login, manage courses, access content, interact socially, and perform administrative tasks), rather than technical details.

Each feature has a risk rating to indicate potential impact or complexity from the user's viewpoint. High-risk items are typically those that affect security, privacy, essential functionalities, or critical data. Medium and low risks are assigned to less critical functionalities or those with fewer potential user-facing problems.

7. Features Not to be Tested

1. **External Systems**
    - **University ERP integration**:
      Not tested this cycle because integration will occur in a future release.
    - **Payment gateway systems**:
      Not in the current project scope; planned for a later version.
    - **External email servers**:
      Existing stable functionality already verified in prior releases, considered low-risk for this iteration.
2. **Hardware-specific Features**
    - **Printer compatibility**:
      No immediate user requirement for printing; deferred testing until actual need arises.
    - **Specialized hardware integrations**:
      Not included in this release as these integrations are targeted for a subsequent hardware deployment.

3. **Browser-specific Features**
    - **Legacy browser support**:
      Will not be tested because the project has officially discontinued legacy browser support due to low usage and high maintenance costs.
    - **Mobile-specific optimizations**:
      Planned for a dedicated mobile release; current focus is on desktop functionality only.

8. Approach

Testing Levels and Coverage:

1. **Unit Testing**
    - *Focus:* Validate individual classes and methods against defined functionality.
    - *Approach:* Use JUnit for testing Java classes and Mock objects to simulate dependencies.
    - *Metrics Collected:* Code coverage (percentage of methods/lines tested), unit test pass/fail rate.
    - *Rationale:* Early defect detection at the smallest testable unit.
2. **Integration Testing**
    - *Focus:* Verify proper interaction between modules (e.g., controllers, API endpoints, database layers).
    - *Approach:* Utilize Postman/REST Assured for API endpoints, ensure stable controller interactions and validate database queries and transactions.
    - *Metrics Collected:* Integration test pass/fail rate, defect density related to interfaces, response times for API calls.
    - *Rationale:* Ensure modules work together as intended, uncover interface or data flow issues.
3. **System Testing**
    - *Focus:* Assess the complete, integrated system against the requirements from an end-to-end perspective.
    - *Approach:* Perform scenario-driven workflows, run Selenium-based UI tests, execute performance tests with JMeter, and conduct security tests (e.g., OWASP-related checks) to ensure system reliability and compliance.

    - *Metrics Collected:* End-to-end defect density, performance metrics (response time, throughput), security vulnerability counts.
    - *Rationale:* Validate that the entire system meets functional and non-functional requirements before user acceptance.

4. **User Acceptance Testing (UAT)**
    - *Focus:* Confirm that the system satisfies user needs and business requirements in realistic use scenarios.
    - *Approach:* Conduct scenario-based testing and workflow validation using real-world user tasks.
    - *Metrics Collected:* UAT defect count, user satisfaction rating (qualitative), completion rates for user scenarios.
    - *Rationale:* Ensure that end-users can effectively and efficiently perform their intended tasks with the final product.

Tools and Technologies:

- **JUnit:** For automated unit testing. No special training beyond standard developer familiarization.
- **Postman/REST Assured:** For API verification. Testers receive brief training as needed.
- **Selenium:** For UI regression and automation. Training provided if testers are new to web automation frameworks.
- **SonarQube:** For static code analysis and code quality metrics. QA and development leads will ensure team members understand how to interpret SonarQube reports.
- **JMeter:** For performance and load testing. Some testers may require basic training in creating test scripts and analyzing load test results.

Configuration Management and Environments:

- All tests will run against a controlled set of environments (Dev, Test, Staging) managed by the CM team.
- Hardware/Software configurations will be documented, and each test environment will mimic the production setup as closely as possible.
- Versions of components and third-party libraries will be tracked in a configuration management repository to ensure consistency and traceability.

Regression Testing Strategy:

- Regression tests will be performed at all levels (Unit, Integration, System) after critical defect fixes and before major releases.

- Priority will be given to high-risk areas identified earlier (e.g., authentication, payment-related features, performance-critical functions).
- Severity and priority of defects will guide the extent of regression testing; high-severity issues trigger more comprehensive regression cycles.

Metrics and Reporting:

- Metrics such as defect density, code coverage, test execution rate, and performance benchmarks will be collected at their relevant testing levels.
- Weekly test reports will summarize progress, defect trends, and readiness for the next test phase.
- Test results will be shared with stakeholders in regular review meetings.

Handling Ambiguous or Untestable Requirements:

- Any unclear or incomplete requirements will be flagged to the Product Owner and Business Analysts.
- Such requirements will be placed on hold, documented in a tracking system, and re-tested only after clarification and confirmation.

Meetings and Communication:

- Regular stand-ups and weekly progress meetings will align QA, Development, and Product teams.
- A change control board meeting will review scope changes and their impact on the test strategy.

Special Conditions and Coverage Requirements:

- New features will be tested fully before release.
- Certain non-critical integrations and legacy features may be partially tested or deferred if agreed upon by stakeholders, documented accordingly.

Tools and Technologies and rest might change and could be later updated.

9. Item Pass/Fail Criteria

Pass Criteria:

- **Unit Level:**
    - At least 90% code coverage as indicated by the coverage tool.

    - All planned unit test cases executed with no critical or high-severity defects remaining.
    - Minor or low-severity defects may remain if they do not impact core functionality.
- **Integration/System Level:**
    - All critical and high-priority test cases pass successfully.
    - No unresolved critical or high-severity defects in integrated components.
    - Performance metrics meet or exceed the defined acceptable thresholds (e.g., response times, throughput).
    - Security and compliance requirements are met, with no known high-severity vulnerabilities.
- **Master (Overall) Level:**
    - All lower-level test plans (unit, integration, system) have been completed according to their defined criteria.
    - A specified percentage of test cases (e.g., 95%) are executed and passed across all test levels.
    - Any remaining minor defects are documented, understood, and agreed upon by stakeholders as acceptable for release.

Fail Criteria:

- Presence of unresolved critical or high-severity defects at any level (unit, integration, system) that impact core functionality.
- Failure to meet minimum performance criteria (e.g., critical transactions consistently exceed required response time).
- Presence of high-risk security vulnerabilities that could compromise data or user privacy.
- Inability to achieve the defined minimum test coverage targets or complete the specified percentage of planned test cases.

Criteria might update later.

10. Suspension and Resumption Requirements

Suspension Criteria:

- **Critical Security Vulnerability Discovered:**
  Testing will be suspended immediately if a vulnerability could compromise user data or system integrity. Proceeding under these conditions risks generating misleading results and wasting resources.
- **System-Wide Performance Issues:**
  If the application's performance degrades to the point where valid test results cannot be obtained (e.g., response times consistently exceed thresholds by a wide margin), further testing is halted to prevent false defect reports.
- **Database Connectivity Failures:**
  Persistent database connectivity issues invalidate test results, as many features depend on reliable data access. Testing stops until the database is stable.
- **More Than 40% Test Case Failures:**
  If over 40% of the executed test cases fail, it indicates a systemic problem. Continuing tests at this point would likely identify "ghost errors" caused by underlying defects rather than meaningful new issues.

Resumption Requirements:

- **Critical Issues Resolved:**
  All identified critical or high-severity defects, including security vulnerabilities, must be fixed or mitigated.
- **System Stability Verified:**
  Performance and stability must return to acceptable baselines, verified by a targeted set of regression tests.
- **Database Connectivity Restored:**
  Database issues must be resolved, ensuring that data-related tests can produce reliable results.
- **Root Cause Analysis Completed:**
  A brief investigation must confirm the underlying cause of the failures and ensure no "ghost errors" remain. This may include resetting or reconfiguring the test environment to a known good state before resuming testing.

11. Test Deliverables

1. **Test Documentation**

- **Test Plan Document:** The overarching plan detailing testing scope, approach, resources, and schedule.
- **Test Design Specifications:** Detailed specifications describing the test conditions, expected results, and methods to ensure coverage.

- **Test Cases and Test Scripts:** Step-by-step instructions for executing tests, including preconditions, inputs, and expected outcomes.
- **Test Data Sets:** Prepared data used for executing the test cases.

2. **Test Execution Artifacts**

- **Test Execution Reports:** Summaries of test runs, indicating which cases passed, failed, or were blocked, and any associated comments.
- **Defect (Problem) Reports:** Documentation of discovered defects, including severity, steps to reproduce, and references to corrective actions taken.
- **Error and Execution Logs:** Logs captured during test execution, including system, server, and tool-generated outputs for troubleshooting.
- **Performance and Security Assessment Results:** Outputs from load testing tools, vulnerability scanners, and any reliability metrics gathered.

3. **Tools and Outputs**

- **Automated Test Scripts and CI/CD Configurations:** The code and configurations that run the tests automatically in the continuous integration pipeline.
- **Simulators and Stubs:** Any simulators, mock services, or stubs used to emulate external dependencies during testing.
- **Static and Dynamic Analysis Reports:** Outputs from code quality tools (e.g., SonarQube) and runtime analysis tools that identify potential issues.

4. **Review and Assessment Reports**

- **Code Review Documentation:** Records of peer reviews, comments, and subsequent modifications.
- **Corrective Action Records:** Documentation of how identified defects were resolved or mitigated, including references to updated code and retesting outcomes.

12. Remaining Test Tasks

1. **Test Setup and Environment Configuration**
    - Configure test environments (development, staging, production)
    - Set up test databases
    - Configure test user accounts with different roles (student, faculty, admin)

2. **Unit Testing Tasks**

   - Test individual controllers:
   - AdminController
   - AudioController
   - BookController
   - CourseCatalogController
   - EventController
   - StudentController
   - TeacherController
   - And other identified controllers

3. **Integration Testing Tasks**

   - Test interactions between:
   - Different controllers
   - Controllers and database
   - Frontend and backend components
   - Authentication/authorization systems

4. System Testing Tasks

   - End-to-end user flows
   - Performance testing
   - Security testing
   - Cross-browser compatibility testing

13. Environmental Needs

1. **Hardware Requirements**

   - Development servers
   - Test servers
   - CI/CD infrastructure
   - Developer workstations with sufficient processing power and memory

2. **Software Requirements**

   - Spring Boot development environment

- Java Development Kit (JDK)
- IDE (IntelliJ/Eclipse)
- Git version control
- Maven/Gradle build tools
- Testing frameworks (JUnit, Mockito)
- Postman for API testing
- Browser testing tools

3. **Network Requirements**

   - Local development network
   - Test environment network
   - Internet connectivity for external dependencies

4. **Database Requirements**

   - Test database instance
   - Database management tools
   - Test data sets

14. Staffing and Training Needs

1. **Required Roles**

   - Test Manager (1)
   - Test Lead (1)
   - Software Test Engineers (2-3)
   - Performance Test Engineer (1)
   - Security Test Engineer (1)
   - Automation Engineers (2)

2. **Training Requirements**

   - Spring Boot testing methodologies
   - Test automation frameworks
   - Security testing tools and techniques
   - Performance testing tools
   - Test management tools

3. **Technical Skills Required**
    - Java/Spring Boot expertise
    - SQL and database testing
    - API testing
    - Web application testing
    - Security testing
    - Performance testing
    - Test automation

15. Responsibilities

1. **Test Manager**
    - Overall test strategy
    - Resource allocation
    - Test plan approval
    - Stakeholder communication
    - Risk management

2. **Test Lead**
    - Test execution oversight
    - Test case review
    - Defect tracking
    - Progress reporting
    - Team coordination

3. **Test Engineers**
    - Test case development
    - Test execution
    - Defect reporting
    - Test automation
    - Documentation

4. **Development Team**

- Unit test development
- Bug fixes
- Code reviews
- Technical support

5. **Project Stakeholders**

   - Requirements clarification
   - User acceptance testing
   - Feature validation
   - Sign-off on deliverables

16. Schedule

1. **Phase 1: Initial Setup (Weeks 1-2)**

   - Environment setup
   - Test planning
   - Test case development
   - Initial test data creation

2. **Phase 2: Unit Testing (Weeks 3-4)**

   - Controller testing
   - Service layer testing
   - Repository testing
   - Initial bug fixes

3. **Phase 3: Integration Testing (Weeks 5-6)**

   - API integration tests
   - Database integration
   - Service integration
   - Cross-component testing

4. **Phase 4: System Testing (Weeks 7-8)**

- End-to-end testing
- Performance testing
- Security testing
- User interface testing

5. **Phase 5: User Acceptance Testing (Weeks 9-10)**
    - Stakeholder testing
    - Bug fixing
    - Final adjustments
    - Sign-off

17. Planning Risks and Contingencies

1. **Technical Risks**
    - Integration complexity with multiple controllers
    - Performance under heavy user load
    - Data security vulnerabilities
    - Browser compatibility issues

2. **Resource Risks**
    - Team member availability
    - Technical skill gaps
    - Environment availability
    - Tool licensing issues

3. **Schedule Risks**
    - Feature completion delays
    - Testing environment setup delays
    - Bug fix turnaround time
    - Stakeholder availability for UAT

4. **Contingency Plans**
    - Backup testing environments

    - Cross-training team members
    - Buffer time in schedule
    - Automated testing to speed up regression
    - Priority-based testing approach
    - Alternative tool options

18. Approvals

1. **Required Approvals:**
    - Project Manager
    - Test Manager
    - Development Lead
    - Quality Assurance Lead
    - Security Team Lead
    - Business Stakeholder Representative

2. **Sign-off Criteria:**
    - Test plan completeness
    - Resource availability confirmation
    - Schedule feasibility
    - Risk assessment acceptance
    - Budget approval

19. Glossary

- **Key Terms:**
    - UAT: User Acceptance Testing
    - API: Application Programming Interface
    - CI/CD: Continuous Integration/Continuous Deployment
    - Spring Boot: Java-based framework for building web applications
    - JUnit: Java testing framework
    - Mockito: Mocking framework for unit tests
    - REST: Representational State Transfer
    - JWT: JSON Web Token (for authentication)
    - SQL: Structured Query Language
    - Maven/Gradle: Build automation tools
    - Git: Version control system
    - IDE: Integrated Development Environment