CSCI3100: Software Engineering

Test Plan of AIO - Course Planner

April 23, 2025

Table of contents

0.1 Document Revision History

Version	Revised By	Revision Date	Comments
1.0	Anson	23 Apr 2025	Initial draft

1 Test Plan for Pokeman Game

This is a test plan for **AIO - Course Planner**. The most important sections include:

1. Scope and Objectives — This part states what will be (and what not will be) included in the testing.
2. Test Cases and Scenarios — This part explains each test case and scenario, by providing a detailed breakdown of the scope, the testing procedures and pass/fail criteria.
3. Team Roles and Responsibilities — This part shows the assignment of specific duties and accountabilities to each team member.
4. Testing Approach & Timeline and Schedule — This part outlines the approach and schedule of the tests.

1.1 1. Scope and Objectives

This part defines the scope of test cases. By having a clear definition, it avoids multiple test cases overlapping on certain areas.

1.1.1 Scope:

The test plan focuses on ensuring the functionality, performance, and user experience of the **AIO - Course Planner**. The following areas will be tested:

- Core mechanics (e.g., Timetable planning, Study Plan planner, Viewing degree roadmap, Course commenting).
- User interface (UI) and user experience (UX).
- Account system (Login, Logout, Registration).
- Email token verification module.
- Save/load functionality (for Timetable and Study Plan).
- Third-party integration to Google Calendar.

1.1.2 Out of Scope:

- Some hard-to-test aspects of the software (please see Risk assessment and mitigation).

1.1.3 Objectives:

- Verify that the system meets functional and non-functional requirements, and provides a bug-free experience.
- Ensure the system is intuitive, and free of critical bugs.
- Confirm that the system performs well under various conditions (e.g., high user load, low-end devices). (????)
- Validate that the system is ready for release and meets quality standards.

1.2 2. Test Cases and Scenarios

The section outlines comprehensive testing strategies for the **AIO - Course Planner** web application. It has test cases for both functional and non-functional test scenarios designed to ensure the application meets quality standards and user requirements. Each test case includes procedures, expected results, and criteria to guide manual testing efforts into the testing process.

The test cases below use manual testing steps currently, which can serve as the foundation for future test automation efforts.

1.2.1 Test Cases for Functional Requirements:

1. User Authentication:
   - Steps: Register a new account, verify email, and login with credentials.
   - Expected Result: User is registered successfully and can log in without email verification (it is required for course commenting only).
   - Pass/Fail Criteria: Account is not created in database or users cannot log in.
   - Test cases: (TODO: DELETE THIS? NO AUTOMATED TESTING IS AVAILABLE)
     - Test Case ID: TC001
       - Repository Link: GitHub - Test Code for TC001
     - Test Case ID: TC002
       - Repository Link: GitHub - Test Code for TC002
2. Course Search and Selection:
   - Steps: Search for courses by keyword, filter results, and view course details.

- Expected Result: Relevant courses are displayed and details are accessible.
- Pass/Fail Criteria: Course search returns accurate results and course information is complete.

3. Study Plan Creation:
   - Steps: Add courses to study plan, modify sequence, and save changes.
   - Expected Result: Study plan is updated and persisted correctly.
   - Pass/Fail Criteria: Changes to study plan are saved and retrieved accurately on reload.
4. Timetable Management:
   - Steps: Add courses to timetable, detect conflicts, and optimize schedule.
   - Expected Result: Courses appear in timetable at correct times with conflict warnings.
   - Pass/Fail Criteria: Timetable visually displays all courses without UI errors.
5. Comment System:
   - Steps: Add comments to courses, edit/delete own comments, and view others' comments.
   - Expected Result: Comments are saved and displayed correctly for all users.
   - Pass/Fail Criteria: Comment operations complete without errors and update in real-time.
6. Degree Roadmap Visualization:
   - Steps: View degree roadmap, interact with course nodes, and check prerequisites.
   - Expected Result: Roadmap displays correct course sequence with prerequisites highlighted.
   - Pass/Fail Criteria: All course relationships are accurately represented in the visualization.

1.2.2 Test Cases for Non-Functional Requirements

This part focus on test cases that assess the non-functional requirements focus on the quality attributes of the performance, scalability, usability, reliability, and security.

1.2.2.1 1. Performance Testing

1. Page Load Time
   - Objective: Verify that pages load within acceptable time limits.
   - Steps:
     1. Navigate to each main page (index, timetable, study planner, etc.)

    2. Measure the time taken to fully load content.

  - Expected Result: Pages load within 2 seconds on standard connections.
  - Pass/Fail Criteria: Pass if load time meets expectations; fail otherwise.

2. API Response Time

   - Objective: Ensure API endpoints respond quickly to client requests.
   - Steps:
     1. Make requests to key API endpoints (course search, authentication, etc.)
     2. Measure response times under different load conditions.
   - Expected Result: API responses return within 500ms on average.
   - Pass/Fail Criteria: Pass if response times remain within acceptable thresholds.

3. Simultaneous User Testing

   - Objective: Test application behavior with many concurrent users.
   - Steps:
     1. Simulate multiple users accessing the application simultaneously.
     2. Monitor server resource usage and response times.
   - Expected Result: Application remains responsive with up to 100 simultaneous users.
   - Pass/Fail Criteria: Pass if performance degradation is minimal under load.

1.2.2.2 2. Usability Testing

1. Navigation Flow

   - Objective: Verify that users can navigate the application intuitively.
   - Steps:
     1. Ask new users to complete common tasks without instructions.
     2. Observe navigation patterns and points of confusion.
   - Expected Result: Users can complete basic tasks without assistance.
   - Pass/Fail Criteria: Pass if users accomplish tasks efficiently; fail if significant confusion occurs.

2. Mobile Responsiveness

   - Objective: Ensure the application is usable on mobile devices.
   - Steps:
     1. Access application on various mobile devices and screen sizes.
     2. Test all major features on mobile browsers.
   - Expected Result: UI adapts appropriately to screen size with all functionality accessible.
   - Pass/Fail Criteria: Pass if mobile experience is consistent with desktop.

1.2.2.3 3. Reliability Testing

1. Data Persistence
   - Objective: Verify that user data is saved correctly and persistently.
   - Steps:
     1. Create study plans and timetables, then log out and back in.
     2. Check that all user-created data is preserved.
   - Expected Result: All user data is persistent across sessions.
   - Pass/Fail Criteria: Pass if no data loss occurs; fail otherwise.
2. Error Recovery
   - Objective: Test application's ability to handle and recover from errors.
   - Steps:
     1. Simulate network disruptions during operations.
     2. Intentionally submit invalid data to test error handling.
   - Expected Result: Application provides meaningful error messages and recovers gracefully.
   - Pass/Fail Criteria: Pass if app maintains usability after errors.

1.2.2.4 4. Security Testing

1. Authentication Security
   - Objective: Verify secure login and session management.
   - Steps:
     1. Attempt various authentication bypass techniques.
     2. Test password strength requirements and account lockout functionality.
   - Expected Result: Unauthorized access attempts are prevented.
   - Pass/Fail Criteria: Pass if authentication barriers cannot be circumvented.
2. Data Protection
   - Objective: Ensure user data is protected.
   - Steps:
     1. Inspect network traffic for sensitive data transmission.
     2. Test that API endpoints properly validate authentication.
   - Expected Result: Sensitive data is encrypted and access-controlled.
   - Pass/Fail Criteria: Pass if no unauthorized data access is possible.

1.2.2.5 5. Compatibility Testing

1. Browser Compatibility
    - Objective: Ensure functionality across major browsers.
    - Steps:
        1. Test application on Chrome, Firefox, Safari, and Edge.
        2. Verify feature parity and visual consistency.
    - Expected Result: Application works identically across all supported browsers.
    - Pass/Fail Criteria: Pass if no browser-specific issues exist.
2. Device Compatibility
    - Objective: Verify application works on different devices and screen sizes.
    - Steps:
        1. Test on desktop, tablet, and mobile devices.
        2. Check for layout or functionality issues.
    - Expected Result: Application is fully functional across all device types.
    - Pass/Fail Criteria: Pass if the experience is consistent across devices.

1.3 3. Resource Allocation

This section outlines the personnel, tools, environments and other resources required to execute the test plan effectively.

1.3.1 1. Team Roles and Responsibilities

1.3.1.1 Key Team Members: (TODO: Fix this part cuz we just share the work...)

1. Test Lead:
    - Oversees the entire testing process.
    - Creates test plans.
    - Reports progress to stakeholders.
2. Manual Testers (2-3):
    - Execute functional and non-functional test cases.
    - Document results.
    - Report bugs.
3. Frontend Developer:
    - Assists with UI/UX testing.
    - Resolves front-end related issues.

4. Backend Developer:
    - Assists with API testing.
    - Resolves server-side issues.
5. Database Administrator:
    - Manages database configurations.

1.3.1.2 Resource Allocation Table: (TODO: Fix this)

Team Member	Primary Testing Focus	Secondary Focus	Contribution (%)
Group Leader	Test planning, coordination	UI/UX testing	25%
Team Member 1	Authentication & user management	Performance testing	25%
Team Member 2	Course search & management	Browser compatibility	25%
Team Member 3	Timetable & study planner features	Security testing	25%

1.3.2 2. Tools and Software

1.3.2.1 Testing Tools:

- Browser Developer Tools: Chrome DevTools for frontend testing.
- MongoDB Compass: For database inspection and verification.
- Manual Testing: Since no framework is used for webapp building.
- Git: For version controlling test scripts/datasets/files and documents.

1.3.3 3. Testing Environments

1.3.3.1 Environment Types:

- Virtual Machine (Docker): Both developing, testing and production environments are containerized by Docker to avoid endpoint discrepency.
- Developing environment: Consists of test files and source codes, used by developers for developing, testing and debugging.
- Production Environment: Used for final validation during User Acceptance Testing. Mimics the live environment to ensure the system is ready for release.

1.3.3.2 Environment Setup:

- Hardware Requirements:
  - Devices/Browsers/DevTool mimicing to test behaviour of the web-application in different devices.
- Software Requirements:
  - Docker (Compose) Containers: For consistent testing environments.
  - Node.js: The backend of the web-application.
  - MongoDB Database: Stores user data and comments.

1.3.3.3 Environment Allocation Table: (TODO: Fix this)

Environment	Purpose	Configuration	Responsibility
Local	Individual feature testing	Local installations on each member's computer	Individual Team Members
Shared Test	Coordinated testing	Docker compose setup accessible to all members	Group Leader + Team Member with Docker knowledge
Production-like	Final validation	Deployment to free tier hosting (if available)	Group Leader

1.3.4 4. Time Allocation

1.3.4.1 Effort Estimation: (TODO: Fix this later)

- Effort Estimation: 1. Test Planning: 3-4 days 2. Test Case Development: 1 week 3. Functional Testing: 1-2 weeks 4. Non-functional Testing: 1 week 5. Bug Fixing and Verification: Ongoing throughout project

1.3.4.2 Example Time Allocation Table: (TODO: Fix this later)

Testing Phase	Duration	Team Members Involved	Deliverables
Test Planning	3-4 days	All team members, led by Group Leader	Test Plan Document
Test Case Development	1 week	All team members	Test Cases, Test Data

Testing Phase	Duration	Team Members Involved	Deliverables
Authentication Testing	2-3 days	Team Member 1, supported by others	Test Results, Bug Reports
Course Management Testing	2-3 days	Team Member 2, supported by others	Test Results, Bug Reports
Study Plan & Timetable Testing	3-4 days	Team Member 3, supported by others	Test Results, Bug Reports
UI/UX Testing	2-3 days	Group Leader, supported by others	Test Results, Bug Reports
Performance & Security Testing	2-3 days	Shared responsibility	Performance Report
Compatibility Testing	1-2 days	Team Member 2, supported by others	Compatibility Report
Bug Fixing & Verification	Ongoing	All team members	Updated Application

1.3.5 5. Budget Allocation

1.3.5.1 Key Budget Considerations:

- Time Investment: Primary resource is team members' time.
- Free Tools: Utilizing free versions instead of premium versions of software/services.
- Personal Computing Resources: Using team members' own computers.
- Cloud Resources: Minimal use of free tier cloud services if needed (e.g. using `EtherealMail` + `NodeMailer` to mimic email sending).
- No Monetary Budget: Focus on free resources and time allocation.

1.4 4. Testing Approach

1.4.1 Types of Testing:

1.4.2 Methodologies:

1.5 5. Timeline and Schedule

1.5.1 1. Waterfall Model (THIS CHANGE)

1.5.1.1 Dependencies:

1.5.1.2 Key Phases and Timeline:

1.5.1.3 Waterfall Model Timeline Example:

1.5.2 2. Agile Model

1.5.2.1 Key Phases and Timeline:

1.5.2.2 Agile Model Timeline Example for a 2-Week Sprint:

1.5.2.3 Agile Model Key Milestones:

1.6 6. Risk Assessment and Mitigation

1.6.1 Potential Risks:

1.7 7. Success Criteria

1.8 8. Reporting Requirements

1.8.1 Documentation:

1.8.2 Communication: