TEST PLAN

for

Kaiju Academy

Version 0.1

Prepared by
Group A2

YU Ching Hei	**1155193237**	**chyu@link.cuhk.edu.hk**
Lei Hei Tung	**1155194969**	**1155194969@link.cuhk.edu.hk**
Ankhbayar Enkhtaivan	**1155185142**	**1155185142@link.cuhk.edu.hk**
Yum Ho Kan	**1155195234**	**1155195234@link.cuhk.edu.hk**
Leung Chung Wang	**1155194650**	**1155194650@link.cuhk.edu.hk**

The Chinese University of Hong Kong
Department of Computer Science and Engineering

CSCI3100: Software Engineering

May 9, 2025

Contents

Contents

Contents ii

Document Revision History iii

1 Introduction 1
- 1.1 Purpose . . . 1
- 1.2 References and Acknowledgments . . . 1

2 Scope and Objectives 1
- 2.1 Scope . . . 1
- 2.2 Out of Scope . . . 1
- 2.3 Objectives . . . 1

3 Test Cases and Scenarios 1
- 3.1 Functional Test Cases . . . 1
- 3.2 Non-Functional Test Cases . . . 3
  - 3.2.1 Performance Testing . . . 3
  - 3.2.2 Security Testing . . . 3
  - 3.2.3 Usability Testing . . . 3
  - 3.2.4 Reliability Testing . . . 3
  - 3.2.5 Compatibility Testing . . . 3

4 Resource Allocation 3
- 4.1 Team Roles and Responsibilities . . . 3
- 4.2 Tools and Software . . . 4
- 4.3 Testing Environments . . . 4
- 4.4 Time and Budget Estimation . . . 4

5 Testing Approach 4
- 5.1 Types of Testing . . . 4
- 5.2 Methodologies . . . 4

6 Timeline and Schedule 4
- 6.1 Waterfall Model Example . . . 4
- 6.2 Agile/Sprint Model (If Used) . . . 5

7 Risk Assessment and Mitigation 5
- 7.1 Risk Analysis . . . 5

8 Success Criteria 5
- 8.1 Acceptance Criteria . . . 5

9 Reporting Requirements 5
- 9.1 Reporting Methods . . . 5

Document Revision History

Version	Revised By	Revision Date	Comments
0.1	C. W. Leung	2025-05-06	Initial draft for Kaiju Academy Test Plan
0.2	C. W. Leung	2025-05-07	Added typical, edge and exceptional test cases and scenarios

1 Introduction

1.1 Purpose

This document defines the test plan for the Kaiju Academy online learning platform. It describes the scope, objectives, test cases, resources, approach, schedule, risks, and reporting methods for the project.

1.2 References and Acknowledgments

- Kaiju Academy Software Requirements Specification v1.1
- Kaiju Academy Design and Implementation v1.1
- This document was prepared with the assistance of AI tools (e.g., ChatGPT 4.1) for drafting and review.

2 Scope and Objectives

2.1 Scope

Testing will cover the following key areas of Kaiju Academy:
- User registration, login, authentication (including MFA)
- Course browsing, enrollment, and access
- Interactive code assessment and automated grading
- Profile and progress tracking (student and educator)
- Educator course and material management
- Error handling, security, and data integrity
- Responsive UI/UX across devices (desktop, tablet, mobile)
- Licence management: entering and validating a licence key before accessing course content.

2.2 Out of Scope

- Community features (forum, dashboard, notifications, calendar)
- Credit/Payment purchasing and related flows
- Modding/hacking scenarios and hardware-specific edge cases
- Third-party payment gateway/internal payment logic

2.3 Objectives

- Verify that all main user stories and requirements are implemented and work as intended
- Ensure the platform meets performance, usability, and security expectations
- Confirm the system is stable and ready for release
- Validate integration with external modules (SurrealDB, AWS)

3 Test Cases and Scenarios

3.1 Functional Test Cases

To ensure each feature works as expected based on system requirements. These tests validate that the platform behaves correctly under normal and edge-case scenarios.

1. **User Registration and Login**
    - *Typical:* Register with valid email, verify, and login with correct password.
    - *Edge:* Register with an email already in use; login with password at minimum/maximum length; attempt login immediately after registration but before email verification.
    - *Exceptional:* Register with invalid email format; login with SQL injection in username field; brute-force login attempts.
    - Expected Result:

  - Typical: Account created, verification email sent, successful login.
  - Edge: Duplicate email rejected, password constraints enforced, pre-verification login prevented.
  - Exceptional: Invalid format rejected, security input sanitized, rate limits triggered.
- Pass/Fail: Account appears in database, valid JWT issued, protected endpoints accessible after login.

2. **Course Browsing and Enrollment**
   - *Typical:* Browse catalog, enroll in available courses.
   - *Edge:* Attempt to enroll in a course at enrollment capacity; unenroll and re-enroll quickly.
   - *Exceptional:* Enroll in a non-existent course via forged request; enroll without being logged in.
   - Expected Result:
     - Typical: Enrollment successful, course appears in profile.
     - Edge: Enrollment denied if at capacity, rapid actions handled gracefully.
     - Exceptional: Forged/non-authenticated requests rejected.
   - Pass/Fail: Enrolled courses listed, correct access to course content.

3. **Interactive Code Assessment**
   - *Typical:* Submit valid code, receive grade and feedback.
   - *Edge:* Submit code at maximum allowed length; submit code that runs close to time/memory limits.
   - *Exceptional:* Submit malicious code (e.g., infinite loop, file access); submit empty or syntactically invalid code.
   - Expected Result:
     - Typical: Code graded, output and feedback displayed.
     - Edge: Large/slow submissions handled, limits enforced.
     - Exceptional: Malicious/invalid code blocked, user notified.
   - Pass/Fail: Output matches expected, grade visible in profile.

4. **Educator Course Management**
   - *Typical:* Create a new course, upload materials, update course info.
   - *Edge:* Upload a file at the size limit; make rapid, successive updates.
   - *Exceptional:* Attempt to delete a course in use; upload unsupported file types; educator tries unauthorized changes.
   - Expected Result:
     - Typical: Changes visible to students, materials accessible.
     - Edge: Size/operation limits enforced, system remains stable.
     - Exceptional: In-use/unauthorized actions blocked, errors reported.
   - Pass/Fail: CRUD operations function, permissions enforced.

5. **Profile and Progress Tracking**
   - *Typical:* Complete module, progress bar updates.
   - *Edge:* Complete all modules in rapid succession; access progress from multiple devices simultaneously.
   - *Exceptional:* Data corruption or missing progress data; attempt to alter progress via API.
   - Expected Result:
     - Typical: Progress updates in real time.
     - Edge: Sync issues handled, state remains consistent.
     - Exceptional: Data integrity preserved, tampering rejected.
   - Pass/Fail: Progress bar/percentage accurate, completed modules listed.

6. **Licence Management**
   - *Typical:* Enter valid licence key, gain access.
   - *Edge:* Enter licence key at maximum/minimum accepted length; attempt to use a key already in use by another user.
   - *Exceptional:* Input malformed key, SQL/script injection in licence field, repeated invalid attempts.
   - Expected Result:

  - Typical: Valid key grants access.
  - Edge: Key limits enforced, duplicate use prevented.
  - Exceptional: Malformed/injected keys rejected, lockout after repeated failures.
- Pass/Fail: System restricts access appropriately; error messages are clear.

3.2 Non-Functional Test Cases

3.2.1 Performance Testing

To check if the platform can handle high user loads, fast page loads, and real-time responses during tasks. This ensures the system remains responsive and efficient even under stress.
- Home page, login, and course page load within 2 seconds for 95% of users
- Code execution returns result within 5 seconds for 99% of cases

3.2.2 Security Testing

To ensure the platform protects sensitive data and enforces access controls.
- Only authenticated users can access protected endpoints
- Role-based access control enforced (e.g., only educators can create courses)
- No sensitive data stored in plain text; JWT securely signed

3.2.3 Usability Testing

To ensure the platform is user-friendly and supports a smooth learning experience.
- Navigation is clear and intuitive for both students and educators
- UI is responsive on mobile and desktop devices

3.2.4 Reliability Testing

To ensure the platform can recover from failures and provide meaningful error messages. This builds trust that user progress and course content remain stable and accessible.
- System recovers from AWS or DB failure without data loss
- Error messages are meaningful for invalid operations

3.2.5 Compatibility Testing

To verify that the platform works correctly across different devices, operating systems, and browsers. This ensures that all users have a consistent experience regardless of their setup.
- Supported browsers (Chrome, Firefox, Safari, Edge) render all pages correctly
- All main functions work on Windows, macOS, Android, iOS

4 Resource Allocation

4.1 Team Roles and Responsibilities

Role	Name	Responsibilities
Backend Developer	C. H. Yu	Assist with unit/integration test, bug fixing
Cloud Engineer	Ankhbayar	Deploy project to AWS, configure hosting and cloud resources
UI/UX Designer	H. T. Lei	Validate user interface and accessibility
Frontend Developer	H. K. Yum	Develop and maintain user interface; assist with unit/integration testing and bug fixing
Documentation Specialist	C. W. Leung	Maintain documentation, reports
Product Owner	Group A2	Validate requirements and acceptance criteria

4.2 Tools and Software

- Test Management: GitHub issues, test cases in code repository
- Automation: Cypress (UI), Jest (unit), custom scripts
- Bug Tracking: GitHub, Jira
- Performance: Browser dev tools, Lighthouse
- Compatibility: BrowserStack, manual device testing
- CI/CD: GitHub Actions

4.3 Testing Environments

Environment	Purpose	Owner
Development	Unit testing, feature development	Developers
Staging	System/integration testing, regression	QA Team
Production (UAT)	Final acceptance, release candidate	Product Owner

4.4 Time and Budget Estimation

- Test Planning: 10%
- Test Case Development: 15%
- Test Execution: 50%
- Bug Fixing/Retesting: 20%
- Regression: 5%
- Budget: Team time, device access, third-party tools (if any)

5 Testing Approach

5.1 Types of Testing

- **Unit Testing**: Rust/TypeScript unit tests for backend/frontend logic
- **Integration Testing**: End-to-end user flows (registration, course enrollment, code submission)
- **System Testing**: Full workflow from user registration to course completion
- **Regression Testing**: After each major code change
- **User Acceptance Testing**: Final validation by product owner/instructors

5.2 Methodologies

- Manual testing for UI/UX and major user stories
- Automated testing for regression and repetitive flows
- Exploratory testing for edge and negative cases

6 Timeline and Schedule

6.1 Waterfall Model Example

Phase	Duration	Activities
Test Planning & Preparation	Week 1	Test plan, environment setup
Unit Testing	Week 2	Backend/frontend unit tests
Integration Testing	Week 3	End-to-end flow testing
System Testing	Week 4	Full platform testing
UAT & Regression	Week 5	User acceptance, bug fixing, regression

6.2 Agile/Sprint Model (If Used)

- Each sprint: Write user-story based tests, run regression, demo to stakeholders

7 Risk Assessment and Mitigation

7.1 Risk Analysis

- **Delays in Development**: Frequent communication, adjust test schedule
- **High Bug Volume**: Prioritize critical bugs, allocate more QA time
- **Integration Failures**: Early integration testing, CI/CD monitoring
- **Compatibility Issues**: Early device/browser testing, use BrowserStack
- **Untestable Features**: Add debug/logging where possible, clarify requirements

8 Success Criteria

8.1 Acceptance Criteria

- All high/critical bugs are resolved
- All functional and non-functional requirements pass
- Positive feedback from UAT/instructors
- System is ready for deployment

9 Reporting Requirements

9.1 Reporting Methods

- Test results and bug status tracked via GitHub issues
- Test execution summary and coverage report before release
- Weekly status meetings with stakeholders
- Final test summary and sign-off document