SOFTWARE REQUIREMENTS SPECIFICATION

for

Kaiju Academy

Version 0.6

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
<name>	<student id>	<email>
<name>	<student id>	<email>

The Chinese University of Hong Kong
Department of Computer Science and Engineering
CSCI3100: Software Engineering

February 10, 2025

Contents

Document Revision History

Version	Revised By	Revision Date	Comments
0.1	C. H. Yu	2025-02-08	Updated: –Initial document structure –Basic template setup
0.2	C. H. Yu	2025-02-08	Updated: –Formatting
0.3	C. H. Yu	2025-02-08	Updated: –Titlepage formatting –Page numbering –Chapters title formatting –Chapter and sections arrangement
0.4	H. T. Lei	2025-02-09	Added: –Specific requirements
0.4.1	C. H. Yu	2025-02-10	Updated: –Specific requirements: fixed compilation error –Titlepage formatting: fixed alignment issue
0.5	C. H. Yu	2025-02-10	Added: –Acronyms and Abbreviations table –Product Overview –Product Functionality –User Classes and Characteristics –Design and Implementation Constraints –Assumptions and Dependencies
0.6	A. Enkhtaivan	2025-02-10	Updated: –System Features –Specific requirements Use Case Model

1. Introduction

1.1 Document Purpose

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.> TODO: Write 1-2 paragraphs describing the purpose of this document as explained above

1.2 Project Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. TODO: 1-2 paragraphs describing the scope of the product. Make sure to describe the benefits associated with the product. >

1.3 Intended Audience and Document Overview

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers (In your case it would probably be the "client" and the professor). Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

1.4 Definitions, Acronyms and Abbreviations

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS. TODO: Please provide a list of all abbreviations and acronyms used in this document sorted in alphabetical order >

Abbreviation	Definition
AI	Artificial Intelligence
API	Application Programming Interface
CCPA	California Consumer Privacy Act
CD	Continuous Deployment
CDN	Content Delivery Networks
CI	Continuous Integration
CSS	Cascading Style Sheets
FAQ	Frequently Asked Questions
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
JS	JavaScript
LMS	Learning Management System
MC	Multiple Choice
OS	Operating System
REQ	Requirement
REST	Representational State Transfer
SRS	Software Requirements Specification
SSL	Secure Sockets Layer
TBD	To Be Determined
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
2FA	Two-factor Authentication

1.5 Document Conventions

<In general this document follows the IEEE formatting requirements. Use Arial font size 11, or 12 throughout the document for text. Use italics for comments. Document text should be single spaced and maintain the 1" margins found in this template. For Section and Subsection titles please follow the template.
TODO: Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. Sometimes, it is useful to divide this section to several sections, e.g., Formatting Conventions, Naming Conventions, etc >

1.6 References and Acknowledgments

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

2. Overall Description

2.1 Product Overview

Kaiju Academy is a web-based e-learning platform that provides interactive programming education through modern LMS capabilities and specialized coding features, facilitating the learning of programming anytime and anywhere as long as you are connected to the internet.

The platform serves as a comprehensive learning environment where students can engage with programming concepts through a structured curriculum. Users access the platform through web browsers, eliminating the need for software installation reducing the barrier to entry for users.

The platform employs a module-based learning approach, where content is organized into discrete units that build upon each other. Each module typically contains:

- Video lectures with synchronized transcripts

- Interactive coding exercises

- Supplementary reading materials

- Practice problems and assignments

- Progress assessments

Students can track their progress through a personalized dashboard, which displays completed modules, upcoming assignments, and achievement metrics. The integrated code editor allows students to practice coding directly within the browser, with immediate feedback and automated grading capabilities.

Educators can utilize the platform to create courses, manage course materials and provide targeted assistance where needed. They could also answer questions from students in the discussion forum. The system's analytics tools help identify areas where students may be struggling, enabling educators to provide additional teaching resources. The platform features automated grading and feedback for assignments, allowing educators to have more flexible arrangement.

2.2 Product Functionality

Kaiju Academy provides the following major functionalities:

- User Management: Account creation, authentication, role-based access, and profile settings

- Course Management: Creation and organization of courses, content management, and enrollment tracking

- Learning Content: Video lectures, documents, quizzes, coding assignments, and progress tracking

- Coding Environment: Browser-based code editor with testing and feedback

- Discussion Features: Course forums, announcements, and community interaction

- Assessment Tools: Automated grading, progress tracking, and achievement system

- Administration: User management, content moderation, and system monitoring

2.3 User Classes and Characteristics

Kaiju Academy serves several distinct user classes, each with specific characteristics and needs:

Administrators: Manage platform operations and user access
- High technical expertise with system administration
- High privilage access for maintenance and monitoring
- Need complete system control and monitoring tools
- Require access to all platform features and settings

Students: Primary users of the learning platform
- From various technical expertise levels (beginner to advanced)
- Regular access for course participation and completion
- Need intuitive interface and clear learning paths
- Need interactive learning environment
- Require progress tracking and performance feedback

Educators: Create and manage course content
- Technical expertise in their teaching domains
- Regular access for content updates and student monitoring
- Need comprehensive content management tools
- Require analytics and student progress tracking capabilities

Content Moderators: Monitor and manage forum discussions
- Moderate technical expertise required
- Regular access for content moderation
- Need moderation tools and user management features
- Require access to communication and reporting tools

2.4 Design and Implementation Constraints

The implementation and ongoing development of Kaiju Academy shall be governed by the following technical and operational constraints:

Hardware Constraints
- TODO: Add hardware constraints

Security Constraints
- Must implement OAuth2 for authentication
- All communications must be encrypted using SSL/TLS
- User passwords must be hashed using industry-standard algorithms
- Code execution must be isolated in secure containers
- Regular security audits must be conducted
- Must comply with data protection regulations

Performance Constraints
- Page load time must not exceed 3 seconds

- System must support at least 1000 concurrent users
- Video streaming must adapt to user bandwidth
- Code execution response time must be under 5 seconds
- Database queries must complete within 1 second

Development Constraints

- Must follow Git version control practices
- Code must pass automated testing before deployment
- Must implement CI/CD pipeline
- Must follow RESTful API design principles
- Must maintain comprehensive API documentation

Operational Constraints

- System must achieve 99.9% uptime
- Must implement automated backup systems
- Must support horizontal scaling
- Must implement monitoring and logging
- Must provide disaster recovery procedures

2.5 Assumptions and Dependencies

2.5.1 Assumptions

- **Internet access:** Since the application relies on a web interface, users must have stable internet connectivity with minimum 5 Mbps bandwidth or access through institution's local network
- **Minimum System Requirements:** All client devices must feature:
  - Computer or mobile device with minimum 4GB RAM for smooth performance
  - Web browser (Chrome, Firefox, Safari, Edge - latest 2 versions)
  - HTML5 and JavaScript enabled

2.5.2 Dependencies

- **Third-Party Libraries and Services:**
  - Authentication protocols (OAuth2)
  - Database management systems (PostgreSQL, MongoDB, Redis)
  - Cloud infrastructure services (AWS/Google Cloud/DigitalOcean)
  - Content delivery networks (CDN)
  - Video hosting platforms
  - Code editor components (CodeMirror/Monaco)

- **Development Methodologies:**
  - UML Modeling: System architecture documentation and design specifications
  - COMET Methodology: Concurrent object modeling for workflow integration
  - Container orchestration with Docker and Kubernetes
  - CI/CD pipeline implementation

2.5.3 Critical Risks

- **External System Failures:**
    - Third-party service disruptions (authentication, video hosting)
    - Cloud infrastructure outages
    - CDN performance issues
    - Database system failures

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>
Kaiju Academy will have a graphical user interface (GUI) designed for ease of use and accessibility across different roles, including students, educators, and administrators. The UI components include:

· **Dashboard:** Upon login, users will be presented with a dashboard displaying their enrolled courses, upcoming deadlines, progress tracking, and notifications.

· **Navigation:** A global navigation bar providing quick access to Courses, Assessments, Calendar, Discussion Forum, and Profile settings.

· **Course Pages:** Each course contains a structured layout displaying modules, videos, PDFs, quizzes, and assessments with a progress tracker.

· **Assessment Interface:** Interactive assessment screens allowing multiple-choice (MC) questions with self-checking, short-answer autograded questions, and long-answer questions submitted for manual grading.

· **Progress Tracking:** A visual roadmap for students to track completed and pending modules and overall course progress.

· **Discussion Forum:** A interactive discussion forum with search, filter, and sort options.

· **Calendar:** An integrated calendar highlighting course schedules, assignment deadlines, and upcoming events.

· **Notifications:** A notification center alerting users about new content, deadlines, and updates.

· **Coding Environment** An embedded coding environment for practice exercises and coding competitions, similar to LeetCode.

· **Error Handling** Consistent error messages displayed in case of invalid input, failed login attempts, or system errors.

· **Keyboard Shortcuts** Common keyboard shortcuts for navigation and execution of commands, enhancing efficiency.

· **Mobile Compatibility:** A responsive design ensuring accessibility on desktops, tablets, and mobile devices.

3.1.2 Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>
The platform is designed to support various hardware components. Key considerations include:

· **User Devices:** The platform will be compatible with desktops, laptops, tablets, and smartphones supporting modern web browsers.

· **Server Infrastructure:** Hosted on cloud-based servers (AWS, Google Cloud, or Azure) with auto-scaling to accommodate user load.

· **Peripheral Support:** Users can interact using keyboards, mice, touchscreens, and audio devices for accessibility.

· **Network Requirements:** Requires a stable internet connection with minimum bandwidth to support video streaming and coding environment interaction.

3.1.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

The system will integrate with various software components to ensure smooth operation and functionality. These include:

· **Operating Systems:** Compatible with Windows, macOS, Linux, Android, and iOS.

· **Web Browsers:** Supports the latest versions of Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge.

· **Database Management System:** Uses PostgreSQL or MySQL to store user data, course materials, progress tracking, and assessment records.

· **Authentication Services:** Integration with OAuth2 for third-party authentication (Google, GitHub, etc.) and two-factor authentication (2FA).

· **APIs:** RESTful APIs to connect frontend and backend services, including: User authentication and management, Course content retrieval and management, Assessment grading and tracking, Notification and messaging services, Code execution API for online coding exercises

· **Content Delivery Networks (CDN):** Utilized for efficient media delivery and reduced latency.

· **Notification Services:** Integration with email and push notification services for alerts and reminders.

· **Logging and Monitoring:** Implementation of centralized logging and monitoring tools for system health tracking.

3.2 Functional Requirements

<Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. This section is the direct continuation of section 2.2 where you have specified the general functional requirements. Here, you should list in detail the different product functions. >

Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks, or functions the system is required to perform. Below is a detailed list of product functions:

· **User Management:**   1. Users can sign up, log in, and update their profiles.

    2. Role-based access control (Student, Educator, Admin).

    3. Password recovery and security settings.

· **Course Management:**   1. Educators can create, update, and delete courses.

    2. Upload and manage course materials (videos, PDFs, quizzes).

    3. Students can enroll, drop, and track their progress.

· **Quiz and Assessment:** 1. MC questions with self-checking and automatic grading.

2. Short-answer questions (some autograded, some educator-reviewed).

3. Long-answer questions assigned to educators for grading.

4. Popup MC questions during the course for engagement.

· **Progress Tracking:** 1. Roadmap displaying completed and pending modules.

2. Percentage-based completion tracker.

3. Assessment performance tracking.

· **Notifications:** 1. Automated alerts for deadlines, new content, and assessments.

2. Customizable notification preferences.

· **Search and Filter:** Users can search and filter courses, discussions, and assessments.

· **Calendar Integration:** Displays course schedules, assignment deadlines, and learning reminders.

· **Discussion Forum:** Users can ask questions, respond, and interact with educators.

· **Daily Assessment & Learning Reminders:** Personalized daily learning tasks and notifications.

· **Learning Path Review & Recommendations:** System-generated personalized course suggestions based on progress.

· **Online Coding Judge:** Users can solve coding problems with real-time execution and have coding competitions with leaderboard tracking.

3.3 Use Case Model

3.3.1 Use Case 1: View Course Progress

Description

This use case describes how a student can view their progress in any course

Actors and Preconditions

- **Actors**: Student

- **Preconditions**: The student must be logged in and enrolled in the course.

Steps

1. The student navigates to the "Courses" section.

2. The student selects a course.

3. The system displays the course progress.

3.3.2 Use Case 2: Course Modification

Description

This use case describes the process of an educator modifies a course on the Kaiju Academy platform.

Actors and Preconditions

- **Actors**: Educator

- **Preconditions**: The educator must be logged in and have the necessary permissions to modify courses.

Steps

1. The educator navigates to the "Course Management" section.

2. The educator enters course details (e.g., title, description, modules, and materials).

3. The system validates the course details.

4. The system adds the course to the platform and notifies enrolled students (if applicable).

Alternative Flow

- **Step 4a**: If the educator provides invalid details (e.g., missing title), the system displays an error and prompts corrections.

- **Step 5a**: If the educator cancels, the system discards the changes and returns to the "Course Management" section.

4. System Features

This sections serves to describe the functional requirements of the product by its major features. The features are described in detail, including their development priority, user interactions, and specific functions.

4.1 User Dashboard

4.1.1 Description and Priority

The User Dashboard is the High Priority feature as the hub is where users can access their courses, track their progress, view upcoming deadlines, and receive notifications.

4.1.2 Stimulus/Response Sequences

- **User Action**: Logs into the system.
  System Response: Displays the dashboard with course progress, deadlines, and notifications.

- **User Action**: Clicks on a course.
  System Response: Redirects to the course page, displaying modules, videos, and assessments.

- **User Action**: Marks a module as completed.
  System Response: Updates the progress tracker and highlights the next module.

4.1.3 Functional Requirements

Functional Requirements

- **REQ-1**: The dashboard must display enrolled courses, progress, and deadlines upon login.

- **REQ-2**: Users should be able to navigate to course pages directly from the dashboard.

- **REQ-3**: Progress tracking must update in real-time when a module is marked as completed.

4.1.4 Course Management

Description and Priority

The **Course Management** feature is the High Priority feature as the feature enables educators to create, update, and delete courses, as well as manage course materials such as videos, PDFs, and quizzes. It ensures that educators can efficiently organize and maintain course content, providing a structured learning experience.

Stimulus/Response Sequences

- **Educator Action**: Creates a new course.
  System Response: Displays section to the educator to upload course materials and set deadlines.

- **Educator Action**: Updates a course module.
  System Response: Saves changes and notifies enrolled students of updates.

- **Educator Action**: Deletes a course.
  System Response: Removes the course and all associated materials from the system.

Functional Requirements

- **REQ-1**: Educators must be able to create, update, and delete courses.

- **REQ-2**: Course materials (videos, PDFs, quizzes) must be uploaded and organized by modules.

- **REQ-3**: Students should receive notifications when course materials are updated.

4.1.5 Assessment and Grading

Description and Priority

The **Assessment and Grading** feature allows educators to create quizzes and assessments, while students can complete them and receive grades. This feature is of **Medium Priority** as it supports the learning process but is secondary to content delivery. It provides a mechanism for evaluating student understanding and progress.

Stimulus/Response Sequences

- **Educator Action**: Creates a quiz with multiple-choice (MC) and short-answer questions.
  System Response: Saves the quiz and makes it available to students.

- **Student Action**: Completes a quiz.
  System Response: Automatically grades MC questions and displays results.

- **Educator Action**: Reviews and grades short-answer questions.
  System Response: Updates the student's grade and provides feedback.

Functional Requirements

- **REQ-1**: Educators must be able to create quizzes with MC and short-answer questions along with coding assessments with the test cases.

- **REQ-2**: MC, and coding assessment questions must be automatically graded, and results displayed immediately.

- **REQ-3**: Educators should be able to manually and use AI prompt grade short-answer questions and provide feedback.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product's design or use. Define any safety certifications that must be satisfied.>

5.3 Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.5 Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

6. Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>