SOFTWARE REQUIREMENTS SPECIFICATION

for

Kaiju Academy

Version 0.2

Prepared by
Group A2

YU Ching Hei	**1155193237**	**chyu@link.cuhk.edu.hk**
Lei Hei Tung	**1155194969**	**1155194969@link.cuhk.edu.hk**
Leung Chung Wang	**1155194650**	**1155194650@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

1 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	C. W. Leung	2025-02-10	Updated: –Introduction without abbreviations

2 Introduction

2.1 Purpose

This document outlines the software requirements for Kaiju Academy, an online self-learning platform for coding education, Version 0.2 and prepared by Group A2. The document encompasses all essential functionalities, including course delivery, user progress tracking, community interaction, and assessment tools, providing both high-level and specific requirements. This document shall form the basis for all stakeholders and developers to understand intended features and constraints of the platform.

2.2 Document Conventions

This Project Requirements Specification adheres to the IEEE Std 830-1998 standard.It utilizes Times New Roman font in size 11 for consistency and readability. Important terms are highlighted in bold, while supplementary notes are presented in italics. Each requirement is uniquely identified, with higher-level requirements inheriting their priority unless otherwise specified. Sections and requirements are sequentially numbered to facilitate easy navigation. Additionally, technical terms and acronyms are clearly defined in the Glossary.

2.3 Intended Audience and Reading Suggestions

This document is intended for a diverse audience, including software developers, project managers, quality assurance testers, and stakeholders involved in the Kaiju Academy project, organized into sections that detail the overall description of the product, specific requirements, system features, and non-functional requirements. Readers are encouraged to start with the overall description to understand the context and then proceed to the specific requirements relevant to their roles.

2.4 Project Scope

Kaiju Academy is an online self-learning platform aiming to innovate code education. The aspire of Kaiju is to make users' learning of programming easier and more interactive in course-and-assessment-driven environment with community support, enhancing accessibility as users can learn at their own pace, while interactive and game-like experiences drive engagement and completion. Key objectives include a wide variety of coding courses, progress tracking, and instant feedback through assessments.

Kaiju Academy aligns with corporate goals of expanding digital education and enhancing user engagement through technology-driven solutions, supporting the development of a skilled workforce ready for the digital age.

2.5 References

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

3 Overall Description

3.1 Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

3.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>

3.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

3.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

3.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

3.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

3.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

4 External Interface Requirements

4.1 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.

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.

4.2 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.

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.

4.3 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.

4.4 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.

4.5 Use Case Model

<This section is the direct continuation of section 2.3 where you have specified the general use cases. Here, you should list in detail the different use cases. >

4.5.1 Use Case #1

<Describe the use case in detail.>

4.5.2 Use Case #2

<Describe the use case in detail.>

5 System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

5.1 System Feature 1

<Don't really say "System Feature 1." State the feature name in just a few words.>

5.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

5.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

5.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use "TBD" as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1: REQ-2:

5.2 System Feature 2 (and so on)

6 Other Nonfunctional Requirements

6.1 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.>

6.2 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.>

6.3 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.>

6.4 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.>

6.5 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.>

7 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.>

7.1 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.>

7.2 Appendix B: Analysis Models

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

7.3 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.>