

---

# SOFTWARE REQUIREMENTS SPECIFICATION

for

**Kaiju Academy**

Version 0.5

Prepared by  
Group A2

YU Ching Hei	1155193237	chyu@link.cuhk.edu.hk
Lei Hei Tung	1155194969	1155194969@link.cuhk.edu.hk
<name>	<student id>	<email>
<name>	<student id>	<email>
<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

<b>Contents</b>	<b>ii</b>
<b>Document Revision History</b>	<b>iii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Document Purpose . . . . .	1
1.2 Project Scope . . . . .	1
1.3 Intended Audience and Document Overview . . . . .	1
1.4 Definitions, Acronyms and Abbreviations . . . . .	1
1.5 Document Conventions . . . . .	1
1.6 References and Acknowledgments . . . . .	1
<b>2 Overall Description</b>	<b>2</b>
2.1 Product Perspective . . . . .	2
2.2 Product Functionality . . . . .	2
2.3 User Classes and Characteristics . . . . .	3
2.4 Operating Environment . . . . .	4
2.5 Design and Implementation Constraints . . . . .	5
2.6 Assumptions and Dependencies . . . . .	6
2.7 User Documentation . . . . .	6
<b>3 Specific Requirements</b>	<b>8</b>
3.1 External Interface Requirements . . . . .	8
3.1.1 User Interfaces . . . . .	8
3.1.2 Hardware Interfaces . . . . .	8
3.1.3 Software Interfaces . . . . .	9
3.2 Functional Requirements . . . . .	9
3.3 Use Case Model . . . . .	10
3.3.1 Use Case #1 . . . . .	10
3.3.2 Use Case #2 . . . . .	10
<b>4 System Features</b>	<b>11</b>
4.1 System Feature 1 . . . . .	11
4.1.1 Description and Priority . . . . .	11
4.1.2 Stimulus/Response Sequences . . . . .	11
4.1.3 Functional Requirements . . . . .	11
4.2 System Feature 2 (and so on) . . . . .	11
<b>5 Other Nonfunctional Requirements</b>	<b>12</b>
5.1 Performance Requirements . . . . .	12
5.2 Safety Requirements . . . . .	12
5.3 Security Requirements . . . . .	12
5.4 Software Quality Attributes . . . . .	12
5.5 Business Rules . . . . .	12
<b>6 Other Requirements</b>	<b>13</b>
<b>Appendix A: Glossary</b>	<b>14</b>
<b>Appendix B: Analysis Models</b>	<b>15</b>
<b>Appendix C: To Be Determined List</b>	<b>16</b>

## 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
0.5	H. T. Lei	2025-02-09	Added: –Other requirements

# 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 >

## 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 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.>

Kaiju Academy is a new, self-contained e-learning platform designed to provide an interactive and comprehensive coding education experience. The system is built as a web-based application that combines modern learning management capabilities with specialized features for programming education.

The platform consists of several interconnected components:

- **Frontend Application:** Built using modern web frameworks (React/Angular/Vue.js) to provide a responsive and intuitive user interface.
- **Backend Services:** RESTful APIs built with Node.js/Django/Ruby on Rails handling business logic and data management.
- **Database System:** Utilizes both relational (PostgreSQL/MySQL) and NoSQL databases for efficient data storage and retrieval.
- **Code Execution Engine:** Containerized environment (Docker) for secure code execution and testing.
- **Video Content Delivery:** Integration with third-party video hosting services (YouTube/Vimeo) for efficient content streaming.
- **Cloud Infrastructure:** Hosted on cloud platforms (AWS/Google Cloud/DigitalOcean) for scalability and reliability.

The system integrates with various external services:

- **Authentication Services:** OAuth providers for user authentication
- **Video Hosting Platforms:** For course content delivery
- **Code Editor Components:** Integration with CodeMirror or Monaco editor
- **Cloud Services:** For hosting and scaling

### 2.2 Product Functionality

<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.>

Kaiju Academy provides the following major functionalities:

- User Management
  - User registration and authentication
  - Role-based access control (Student, Educator, Admin)
  - Profile management and settings

- Security features including password recovery
- Course Management
  - Course creation and organization
  - Content management (videos, PDFs, quizzes)
  - Course enrollment and progress tracking
  - Module-based learning paths
- Learning Content Delivery
  - Video streaming with playback controls
  - Document viewing and download
  - Interactive quizzes and assessments
  - Progress tracking and roadmaps
- Interactive Coding Environment
  - Integrated code editor
  - Automated code testing
  - Real-time error feedback
  - Secure code execution
- Discussion and Community
  - Course-specific discussion forums
  - Post and comment functionality
  - Pinned announcements
  - User interaction tracking
- Assessment and Progress
  - Multiple assessment types
  - Automated grading
  - Progress visualization
  - Achievement tracking and badges
- Administrative Tools
  - User management dashboard
  - Content management interface
  - System monitoring tools
  - Analytics and reporting

## 2.3 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.>

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

**Students**     • Primary users of the learning platform

- Various technical expertise levels (beginner to advanced)
- Regular access for course participation and completion
- Need intuitive interface and clear learning paths
- 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

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

- 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 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.>

Kaiju Academy is a web-based platform that operates in the following environment:

- Client-Side Requirements**
- Modern web browsers (Chrome, Firefox, Safari, Edge - latest 2 versions)
  - Minimum 4GB RAM for smooth operation
  - Stable internet connection (minimum 5 Mbps)
  - HTML5 and JavaScript enabled
  - Support for WebSocket connections

- Server-Side Infrastructure**
- Cloud-based hosting (AWS/Google Cloud/DigitalOcean)
  - Linux-based operating system
  - Container orchestration with Docker and Kubernetes
  - Load balancing for high availability
  - SSL/TLS encryption for secure communications

- Database Environment**
- PostgreSQL/MySQL for relational data
  - MongoDB for NoSQL requirements
  - Redis for caching and session management
  - Regular backup and recovery systems

- Integration Requirements**
- OAuth2 compatible authentication services

- RESTful API endpoints
- WebSocket support for real-time features
- CDN integration for content delivery
- Third-party video hosting platform compatibility

## 2.5 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).>

The development and implementation of Kaiju Academy must adhere to the following constraints:

**Technical Constraints**      • Must use modern web frameworks (React/Angular/Vue.js) for frontend development

- Backend must be implemented using Node.js/Django/Ruby on Rails
- Database implementation must use PostgreSQL/MySQL for relational data
- Code execution environment must be containerized using Docker
- Must support WebSocket for real-time features
- Video content must be hosted on third-party platforms (YouTube/Vimeo)

**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



## 2.6 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).>

The development of Kaiju Academy is based on the following assumptions and dependencies:

### Technical Assumptions

- Users have access to modern web browsers

- Third-party services (video hosting, OAuth) remain available
- Cloud service providers maintain their service levels
- Required development tools remain available and supported
- Open-source libraries maintain compatibility

### External Dependencies

- Third-party video hosting platforms

- OAuth authentication providers
- Cloud infrastructure services
- Code editor components (CodeMirror/Monaco)
- Content delivery networks

### Business Assumptions

- User base will grow gradually

- Internet connectivity is generally available to users
- Users have basic computer literacy
- Educational content will be provided by educators
- System will be maintained by technical staff

## 2.7 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.>

Kaiju Academy will provide comprehensive documentation for all user classes:

### Student Documentation

- Getting Started Guide

- Platform Navigation Tutorial
- Course Enrollment Guide
- Code Editor Usage Manual
- Assessment Guidelines
- FAQ Section

### Educator Documentation

- Course Creation Guide

- Content Management Manual
- Assessment Creation Tutorial
- Student Progress Tracking Guide
- Best Practices for Course Design
- Video Content Guidelines

**Administrator Documentation**      • System Administration Guide

- User Management Manual
- Security Guidelines
- Backup and Recovery Procedures
- Performance Monitoring Guide
- Troubleshooting Manual

**Documentation Formats**      • Interactive Web-based Documentation

- Downloadable PDF Manuals
- Video Tutorials
- Context-sensitive Help
- API Documentation
- Regular Documentation Updates

All documentation will be:

- Accessible through the platform's help center
- Searchable with comprehensive indexing
- Available in multiple languages
- Regularly updated to reflect system changes
- Version controlled and archived
- Mobile-responsive for various devices

## 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

<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. >

#### 3.3.1 Use Case #1

<Describe the use case in detail.>

#### 3.3.2 Use Case #2

<Describe the use case in detail.>

## 4. 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.>

### 4.1 System Feature 1

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

#### 4.1.1 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).>

#### 4.1.2 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.>

#### 4.1.3 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:

### 4.2 System Feature 2 (and so on)

## 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.>

This section defines additional requirements that are not covered elsewhere in the document.

- **Database Requirements:** The system should support relational databases with ACID compliance.
- **Internationalization:** The platform should support multiple languages, including English, Spanish, and Chinese.
- **Legal Compliance:** The platform must adhere to data protection regulations (e.g., GDPR, CCPA) and accessibility laws.
- **Reuse Objectives:** Modular architecture should be adopted to allow for code reuse across projects and facilitate easier maintenance.
- **Scalability Requirements:** The system should be designed to handle high concurrent users efficiently.
- **Backup and Recovery:** Automated daily backups must be implemented, with a recovery plan in place for data loss scenarios.
- **Customization and Extensibility:** The system should allow customization for branding and the addition of new features without major rework.



## 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.>