# CSCI3100: Software Engineering

**Project Requirements Specification**

September 25, 2025

## Table of contents

# 1 Document Revision History

| Version | Revised By | Revision Date | Comments |
|---|---|---|---|
| 1.1 | CSCI3100 instructors | 21 Nov 2025 | Updated the deadlines; Updated section **Source Code** |
| 1.0 | CSCI3100 instructors | 15 Sep 2025 | Initial draft |

# 2 Introduction

The course project aims to apply concepts from CSCI3100 Software Engineering by specifying, designing, implementing, testing, and documenting a typical software project for facilitating software engineering. It enhances understanding of software engineering and develops essential skills in teamwork and project management.

The project includes two components: (1) documentation, detailing the design and testing processes, and (2) software production, covering specification, design, coding, and demonstration. Documentation and implementation should be managed via a GitHub repository. A demo day at the end of the semester will allow for product presentation, with complete code submitted afterward.

# 3 Scope

## 3.1 Definitions, Acronyms, and Abbreviations

(nil)

## 3.2 Project Overview

In a realm where demon slayers fought valiantly against the forces of darkness, the boss of the Demon Slayer Corps struggled with managing their software projects, leading to chaotic battles and the tragic loss of many brave warriors. Recognizing the dire need for better organization, Kei, the CSCI3100 lecturer, and his eager students decided to step in, determined to create a robust software project management tool tailored specifically for the Demon Slayers. With their combined skills in coding and project management, they envisioned a system that would streamline communication, track progress, and allocate resources effectively, ultimately empowering the slayers to focus on their true mission: vanquishing demons and protecting humanity…

Sample project ideas include:

- An automatic code-event-issue connection system — An automatic code-event-issue connection system for software engineering seamlessly links code changes, event triggers, and related issues, enabling developers to track and manage the impact of code modifications on project workflows and bug resolution efficiently.
- A Kanban in command-line — A command-line version of Kanban that allows users to manage tasks and workflows through a text-based interface, enabling configuration and customization via simple text files for streamlined project organization and tracking.
- Any software that enables collaborative software development

### 3.3 Document Overview

This Project Requirements Specification document defines the complete requirements for the CSCI3100 course project. It outlines the scope and requirements of the project, serving as a guide for the students to understand what needs to be done and as a look-alike reference for writing the software requirements specification.

## 4 Document Conventions

## 5 Referenced Documents

| Reference No. | Reference | Date | Published By | Source |
|---|---|---|---|---|
| Ref1 | ISO/IEC/IEEE 29148:2018(E2) Systems and software engineering — Life cycle processes — Requirements engineering | 2018 | IEEE | IEEE website |

## 6 Assumptions and Dependencies

| Assumption No. | Title | Description |
|---|---|---|
| A1 | Project scale | Undergraduate project level |
| A2 | Student Attitude | All students are good and eager to learn |

## 7 Requirements

### 7.1 Completing the Software Development Process

It is required for every team to complete the software development process in its entirety. Proof of work must include:

1. Documentation (including the documented code).
2. The software system.
3. A journaled, auditable software development process.

Each team can adopt their own software development methodology, whether it be waterfall or any one Agile process. The following phases exist regardless of the chosen methodology.

### 7.1.1 Requirements Specification

The requirements specification document must include all necessary information identified, such as requirements, features, assumptions, use cases, and a high-level overview of the system architecture at the time of submission. This information should be agreed upon by both users and developers. The document should be written in the recommended Software Requirements Specification format.

### 7.1.2 Design and Implementation

The software system should be designed before coding. All design and implementation considerations must be documented in code and/or in a standalone developer-oriented document where appropriate. While UML diagrams are not strictly required, they are extremely useful for clearer explanations and will enhance the reader's understanding. The design and implementation document should include the key components of the software system.

### 7.1.3 Testing

Testing is crucial. Regardless of whether test-driven development or a regular code-test cycle is adopted, testing must not be skipped. A document describing the testing process and strategies should include the following parts:

1. A test plan created during the development process for current and future developers.
2. Representative test cases.

The test plan should list components that were covered and those that were not in the testing process. Additionally, the document should detail some representative test cases designed to evaluate the key functionalities of the system, including the rationale behind their design and the testing approach used.

### 7.1.4 Delivery

At this phase, coding should be frozen for a software release. Decisions regarding the inclusion or exclusion of certain features must be based on factors such as the completeness of those features and the extent of testing conducted. These decisions will be reflected in the release notes. User manuals, which should be developed concurrently with coding, should be finalized at this stage. Consistency between the user manuals and the product's visible behaviours must be ensured.

Additionally, the source control system must maintain a record of which versions of source files have been released.

## 7.2 Software Requirements

### 7.2.1 Global Database

The system must employ either a third-party SQL database (e.g., MySQL, SQLite) or NoSQL database (e.g., MongoDB, Redis) for data storage.

### 7.2.2 User Interface

The system should have a clear graphical or text user interface design. The UI must be consistent and easy to understand, allowing users to operate the system without assistance from developers.

### 7.2.3 User Management

The system should have basic user management functionalities. Specifically, it must allow users to sign up, log in or log out.

### 7.2.4 Licence Management

The system should have basic licence management functionalities. It must allow users to use the software only if they can provide a valid licence key or licence key-file.

A licence key is a string, formatted like AAAA-BBBB-CCCC-DDDD, that users enter into the system to confirm their right to use the software or access specific features. A licence key file contains one or more licence keys, allowing the user to specify the file instead of typing all the keys. Possible licensing schemes include a single licence key for the entire system, a separate licence key for individual features, or a licence key that is valid for a specific subscription period (such as those used for services like Nxxflxx).

This means that having a user account alone is not sufficient for the user. Licence keys must be provided in order to fully enjoy using the system or accessing specific features.

### 7.2.5 Application-Specific Functionalities Stated in the Software Requirements Specification

The system must incorporate the functionalities outlined in the Software Requirements Specification to be submitted and perform as described in the documentation. For a group of $n$ members, $n-1$ externally observable features are expected. These $n-1$ features are implementations of the intended functionalities of the system in addition to the basic functions of Global Database, User Interface, User Management, and Licence Management.

### 7.2.6 Operating System and Environment

The system must be able to run on Linux, Windows, or Android released within 5 years. For web-based applications, it must support major decent browsers, including Firefox and Chrome. The users must be able to install and set up the system without much effort.

### 7.2.7 Code

Code should be well documented, organized, and versioned.

### 7.2.8 Hardware

For web-based or desktop applications, the system must run on hardware with a quad-core x86_64 CPU at 2GHz and 8GB of RAM. For Android applications, the requirements are a quad-core ARM CPU at 2GHz and 2GB of RAM. No advanced or dedicated GPU is required.

## 7.3 Documents

### 7.3.1 Font, Font Size, and Styles

The document must use Times New Roman, size 11 throughout.

### 7.3.2 Cover Page

The cover page must contain the following information:

- Name of the document
- Project title (you may freely name your own project)
- Document version number and revision history
- Printing date
- Group ID
- Names and SIDs of group members

### 7.3.3 Requirements Specification

The main body of the design document should not exceed 10 pages. The filename should be "Group**_Requirements_Specification", where ** is the group ID. Marks will be deducted if the format requirements are not met.

### 7.3.4 Design and Implementation

This document serves as a developer guide. The main body of the design document should not exceed 20 pages. The filename should be "Group**_Design_Implementation", where ** is the group ID. Marks will be deducted if the format requirements are not met.

### 7.3.5 Testing

The main body of the document should not exceed 15 pages. The filename should be "Group**_Testing", where ** is the group ID.

### 7.3.6 Release Notes and User Manual

The main body of the document should not exceed 5 pages. The filename should be "Group**_Release_Notes_User_Manual", where ** is the group ID.

## 7.4 Source control

All project materials (including source code, images, database files, configuration files, documentation, etc.) must be maintained using GitHub. The development history on GitHub is crucial for evaluating the project's coding phases and development process. Version control systems should be utilized to support both the development and documentation of the software system. The project must be submitted via GitHub, and submissions through any other methods will not be accepted. Course instructors will review the version control logs when assessing the coding efforts.

## 7.5 Grouping

Each project group consists of three to five members for the duration of the project. All team members must collaborate closely on the same project based on the requirements outlined in this document. Please inform the instructors if you are unable to form a group.

## 7.6 Other requirements

### 7.6.1 No Plagiarism

When designing the system, the most important feature to keep in mind is that the software product developed should require a reasonable programming effort. Joint work on any technical aspects of the project between groups or teams is not allowed. Any issues regarding project requirements should be directed to the instructors via email, in-person discussions, or tutorial sessions. This policy is in place to enforce team separation for proper credit.

This project should be considered as if there is only one team—namely, your team—accountable for the entire development. Plagiarism in any aspect of the project is strictly prohibited. While reusing existing designs or code (such as from open-source projects) is allowed, clear attribution is required for any reused code. This means that any code not authored must be attributed at the beginning of each submitted file. Professional tools will be employed to verify the percentage of existing code in the project.

It is important to note that excessive reuse of existing code may result in a significant deduction of the project mark.

### 7.6.2 No Free-rider

Although the project grade is generally assigned to the entire team rather than individuals, each team member must understand that a significant portion of their final project grade depends on teamwork. Failures to cooperate with other team members and to contribute an

equitable amount of effort can lead to undesirable outcomes, particularly if other team members raise complaints about non-participating members. Cases of free-riding will be investigated on project demo day, where group members can report free-riders. The course instructors will verify the validity of the complaints with all team members.

## 7.7 List of Requirements

| Requirement No. | Title | Description |
|---|---|---|
| R1 | Completing the Software Development Process | |
| R1.1 | Requirements Specification | A process that produces the requirements and a software requirements specification document |
| R1.2 | Software Design and Implementation | Design, implementation, and documentation |
| R1.3 | Testing | Practices of testing during development, along with a document describing such practices |
| R1.4 | Delivery | User manual and release notes |
| R2 | Functional Software System | A working system whose functionalities are consistent with the listed requirements and specifications |
| R3 | Software Requirements | |
| R4 | Documentation | |
| R5 | Auditable Software Development Process | Meeting minutes, edits history, and issue tracking system logs that can serve as evidence of the development process |

| Requirement No. | Title | Description |
|---|---|---|
| R6 | Source control | Versioned code and documentation in GitHub |
| R7 | Grouping | 4-5 people |
| R8 | No Plagiarism | |
| R9 | No Free-rider | |

(Some requirements are not expanded)

# 8 Schedule

| Schedule No. | Phase Deliverables | Weighting | Due Date (23:59) |
|---|---|---|---|
| SC1 | Team Formation | | 22 Sep 2025 |
| SC2 | Requirements Specification | 10% | 22 Dec 2025 |
| | Design And Implementation | 20% | 22 Dec 2025 |
| | Documentation | | |
| SC3 | Software System (Demo) | 55% | 24 Dec 2025 (demo video) |
| | Testing Document | 10% | 22 Dec 2025 |
| | Release Notes & User Manual | 5% | 22 Dec 2025 |
| | Total | 100% | |

# 9 Grading Criteria

## 9.1 Documentation

Documentation will be graded based on the clarity. "Clarity" indicates whether the software can be recreated by following the documentation. Marks will be deducted if the format requirements are not met. Exceeding the page limit is acceptable if it is justifiable.

## 9.2 Software System

| Grading Criteria No. | Functionality | Requirements | Points |
|---|---|---|---|
| GS1 | Architecture, Design and Implementation | | |
| GS1.1 | Software Architecture, Design and Implementation | Scalable | 8 |
| GS1.2 | Consistency | Consistent with the Requirements Specification | 3 |
| GS1.3 | Usability | Easy to use, features are useful | 6 |
| GS2 | Database | | |
| GS2.1 | Database integration | A database is integrated for certain purposes such as user management, licence management, and system configuration | 4 |
| GS3 | User Interface | | |
| GS3.1 | Graphical or Text UI | An user interface for users to interact with the system | 4 |
| GS4 | User Management | | |
| GS4.1 | Signup | Able to create new user profiles | 2 |
| GS4.2 | Login and logout | Users can use the core functions only after login via login credentials | 2 |
| GS5 | Licence Management | | |
| GS5.1 | Authorization by key or key-file | Allow the system to be used only when valid key or key-file is provided | 4 |

| Grading Criteria No. | Functionality | Requirements | Points |
|---|---|---|---|
| GS6 | Code Documentation | Useful to other developers | 4 |
| GS7 | Application-specific Requirements | | |
| GS7.1 | Every one of the $n-1$ application-specific features | Make the system useful and complete | $\frac{12}{n-1}$ |
| GS8 | Difficulty | Intrinsic difficulty of the problem and/or implementation | 6 |

### 9.3 Auditable Software Development Process

Marks for the documentation and the software system will be deducted if the software development process is not properly recorded.

## 10 Submission Policies

### 10.1 Documentation

Project documents must be submitted alongside signed VeriGuide receipts, following the same submission process as homework. The policy for late submissions, including the VeriGuide receipt, is consistent with the homework policy. Detailed information regarding this policy will be available on the course website.

Furthermore, while the use of AI tools is permitted for this project, it is essential to explicitly cite or acknowledge these tools in the submissions to ensure transparency and maintain academic integrity.

### 10.2 Source Code

Please add the link to your GitHub repository in the Design and Implementation Documentation. Ensure it is accessible to us. The repository should contain all code and/or documentations.