

Software Requirements Specification

Campus Wellness Portal with Medical System and Fitness Center Integration

Version: 1.0

Date: 05/05/2025

Release By: TT6L - Group 3

**Table of Content**

[**1. Introduction**](#_j56nxs9857ny) **3**

[1.1 Purpose](#_75unuusrlw9s) 3

[1.2 Scope](#_c8eecaohfd4r) 3

[1.3 Product Overview](#_qe1l2gw8w8or) 3

[1.3.1 Product Perspective](#_niw6ypvcteih) 3

[1.3.2 Product Functions](#_tgduldkvba2j) 4

[1.3.3 User Characteristics](#_3dvdno6w3l47) 4

[1.3.4 Limitations](#_seply36radau) 4

[1.4 Definitions](#_rwjbm381oy52) 4

[**2. References**](#_2bopy57ido95) **5**

[**3. Requirements**](#_l9kkxtfvpdw6) **5**

[3.1 Functions](#_9sd4k3jmxvho) 5

[3.2 Performance Requirements](#_ooak04jl1s4h) 6

[3.3 Usability Requirements](#_u4sokbljjjs0) 6

[3.4 Interface Requirements](#_chxynaauml5s) 6

[3.4.1 System Interfaces](#_g2v17drnge7q) 6

[3.4.2 User Interfaces](#_yulgj58k8o3z) 6

[3.4.3 Hardware Interfaces](#_y0ktqljwf03t) 6

[3.4.4 Software Interfaces](#_iapgsn18l6ob) 7

[3.4.5 Communications Interfaces](#_gb4h8guttzfy) 7

[3.5 Logical Database Requirements](#_uf7n4ed8gnko) 7

[3.6 Design Constraints](#_dezq6ys120ah) 7

[3.7 Software System Attributes](#_txxuwz4s49x0) 7

[3.8 Supporting Information](#_dks3tyy3dua) 8

[**4. Verification**](#_c4lq8go4u9ze) **8**

[4.1 Verification Approach](#_vci46d1vmth) 8

[4.2 Verification Criteria](#_f0zvozls3dvb) 9

[**5. Appendices**](#_z0q6r66a5kr0) **9**

[5.1 Assumptions and Dependencies](#_nwo8g3vykxu) 9

[5.2 Acronyms and Abbreviations](#_ujh9zc5b1lve) 9

[5.3 Glossary (Optional Section)](#_bm2fx37n9c1z) 10

# **1. Introduction**

## **1.1 Purpose**

(Mapped to 9.6.2 Purpose)

State the purpose of the software being developed, addressing the problem it solves or the goals it meets.

Example:

The purpose of the "University Research Grant Management System" is to automate the submission, review, and approval process for research grant applications within the university.

## **1.2 Scope**

(Mapped to 9.6.3 Scope)

Define the boundaries of the software product, including its capabilities and coverage. Example:

The system will manage the submission, review, approval, and notification of research grant applications.

## **1.3 Product Overview**

Provide an overview of the software product and how it relates to other systems.

Example:

The Research Grant Management System is integrated with the university's Student Information System and Faculty Portal, allowing students to submit applications and faculty members to review and approve them.

### **1.3.1 Product Perspective**

(Mapped to 9.6.4 Product Perspective)

Describe how the software fits into the larger system. Include a context diagram showing how your system interacts with users and other systems.

Example:

If your system is a module within a university portal, show how it connects with the authentication system, finance system, etc.

### **1.3.2 Product Functions**

List the primary functions of the software. Example:

Submit applications

### **1.3.3 User Characteristics**

(Mapped to 9.6.6 User Characteristics)

Describe the intended user groups and any relevant characteristics that could influence usability, such as educational level or technical expertise. Example:

Students, Faculty members, and Administrators are expected to have basic computer skills, with students having no technical expertise required, and faculty having moderate technical familiarity.

### **1.3.4 Limitations**

(Mapped to 9.6.7 Limitations)

Describe any limitations that may affect the functionality or performance of the software. Example:

The system is limited to processing research grant applications within specific academic departments.

## **1.4 Definitions**

Provide clear definitions for all key terms used in the Software Requirements Specification (SRS).

This section ensures that all stakeholders have a shared understanding of the terminology used throughout the document. Definitions can include terms specific to the software system being developed, technical jargon, or concepts that require further clarification.

Example:

• **Application**: A set of software programs designed to perform a specific function for the user. In this context, it refers to the Research Grant Management System that helps manage grant submissions and approvals.

# **2. References**

References list all the sources you’ve cited or consulted while preparing the SRS. These may include standards (like ISO/IEC/IEEE 29148:2018), textbooks, research articles, technical documentation, or software manuals.

Note*:* Use APA 7th edition format for consistency and credibility. This is especially helpful if your SRS will be reviewed in academic settings or by non-technical stakeholders.

Example:

IEEE. (2018). *ISO/IEC/IEEE 29148:2018 Systems and software engineering—Life cycle processes—*

*Requirements engineering*. https://www.iso.org/standard/72089.html

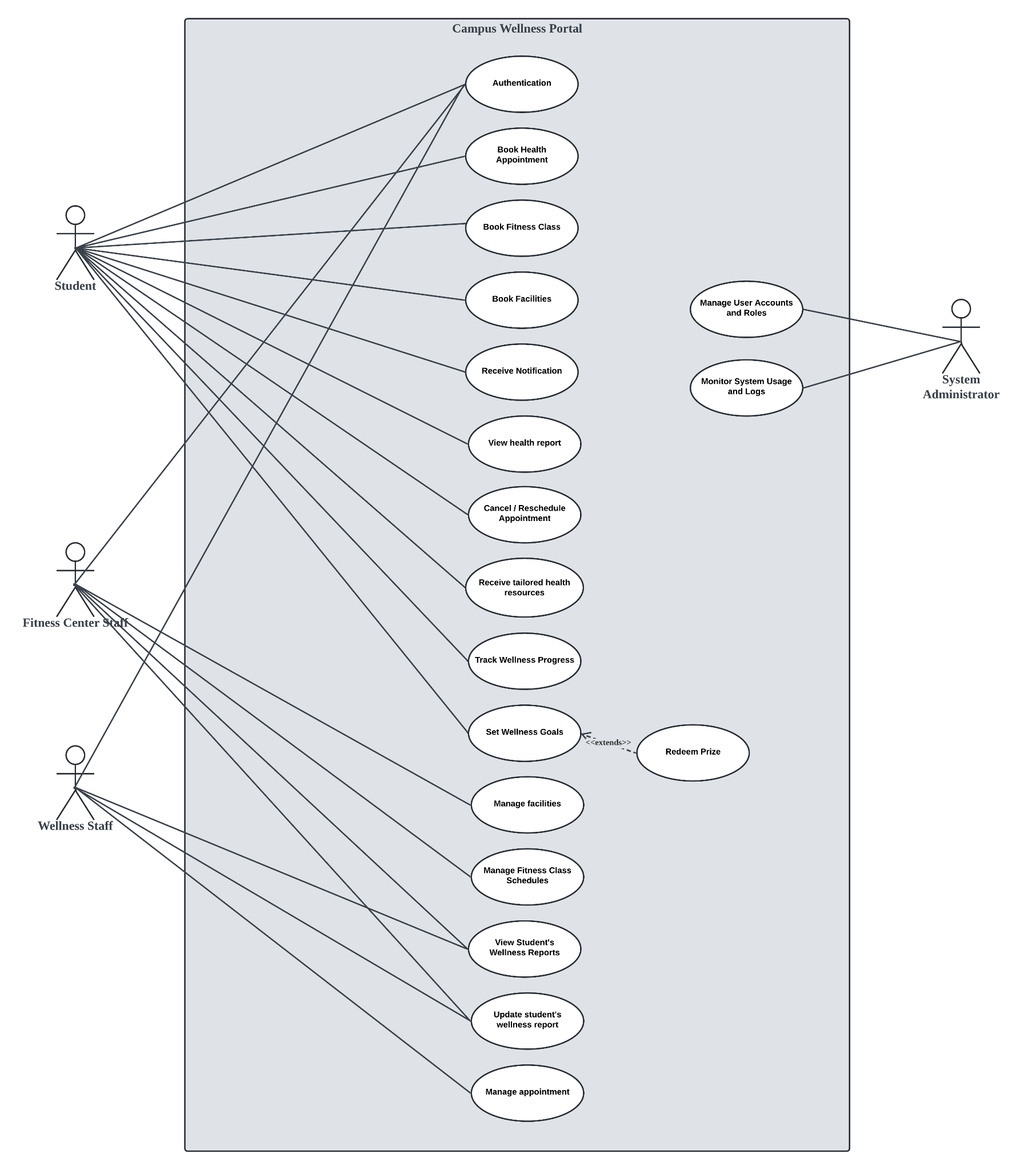
Pohl, K. (2010). *Requirements engineering: Fundamentals, principles, and techniques*. Springer.

# **3. Requirements**

## **3.1 Functions**

(Mapped to 9.6.5 Product Functions)

List the functions of the software and organize them by user (actor). You should provide a use case diagram to represent the system and its functions, as use cases. You can place the use case specifications here and relate each use case to the guidelines in 9.6.10. The process for each function must be clearly specified. For each function, you can illustrate by drawing the sequence/activity diagram.



**Example**:

*Student*: Submit, view, and edit research grant applications.

## **3.2 Performance Requirements**

(Mapped to 9.6.14 Performance Requirements)

Specify performance requirements, both static and dynamic, including response times, throughput, and scalability. These should be measurable with clear, quantitative targets. Example:

The system shall respond to user queries within 2 seconds under a normal load.

## **3.3 Usability Requirements**

(Mapped to 9.6.13 Usability Requirements)

Specify the usability objectives, including ease of use, learnability, efficiency, and user satisfaction. These should be quantifiable and aligned with user needs. Example:

The interface shall allow users to perform primary tasks within 3 clicks.

## **3.4 Interface Requirements**

(Mapped to 9.6.11 External Interfaces and 9.6.4 System Interfaces, User Interfaces, Hardware Interfaces, Software Interfaces, Communications Interfaces)

Specify all system interfaces, including external systems, user interfaces, hardware, and communications.

### **3.4.1 System Interfaces**

: Interfaces with external systems or hardware.

Example: The system will integrate with the university’s authentication system (LDAP).

### **3.4.2 User Interfaces**

: Describe the layout and interaction elements, e.g., navigation, buttons, data entry fields.

Example: The web interface will use a responsive layout with a fixed top navigation bar for easy access to key features.

### **3.4.3 Hardware Interfaces**

: Specify hardware connections, devices, and communication protocols.

Example: The system shall support USB-connected fingerprint readers for user authentication.

### **3.4.4 Software Interfaces**

: Describe interactions with other software or APIs. Example: The system will interact with a third-party cloud service for file storage (e.g., Amazon S3).

### **3.4.5 Communications Interfaces**

: Specify protocols, message formats, and network requirements.

Example: The system will use HTTPS for secure communication between client and server.

## **3.5 Logical Database Requirements**

(Mapped to 9.6.15 Logical Database Requirements)

Describe key data entities, relationships, and constraints. This could include an EntityRelationship (ER) diagram or class diagram. Example:

The “Application” entity has attributes such as applicationID, title, and submissionDate, and it is related to the “Reviewer” entity.

## **3.6 Design Constraints**

(Mapped to 9.6.16 Design Constraints)

List any restrictions or limitations imposed on the design of the software, whether they are from external standards, regulations, or technical limitations. Examples:

The user interface must comply with the university’s branding guidelines.

## **3.7 Software System Attributes**

(Mapped to 9.6.18 Software System Attributes)

Specify the required attributes of the software product, which affect its quality and performance:

• **Reliability**: The system should be able to recover from a crash within 1 minute.

• **Availability**: The system should be available 99.9% of the time during working hours (Monday through Friday, 8 AM to 6 PM).

• **Security**: The system should use role-based access control (RBAC) and encryption for all sensitive user data.

• **Maintainability**: The system should follow best coding practices and be modular to facilitate updates.

• **Portability**: The software should be able to run on both Linux and Windows servers without additional configuration.

## **3.8 Supporting Information**

(Mapped to 9.6.20 Supporting Information)

Any additional supporting information, including:

a) sample input/output formats, descriptions of cost analysis studies or results of questionnaires or any other elicitation techniques;

b) supporting or background information that can help the readers of the SRS;

c) a description of the problems to be solved by the software; and

d) special packaging instructions for the code and the media to meet security, export, initial loading or other requirements.

The SRS should explicitly state whether or not these information items are to be considered part of the requirements.

Example:

Sample input/output formats for key system functions (e.g., CSV format for data export).

# **4. Verification**

## **4.1 Verification Approach**

(Mapped to 9.6.19 Verification)

Specify how the system will be verified, including methods, responsible parties, timing, and locations. Example:

• **How**: Functional testing, unit testing, and system integration testing will be used to verify system performance.

• **Who**: Verification will be conducted by the product team and quality assurance (QA) department.

• **When**: Verification will occur at key milestones in the development cycle (e.g., after each sprint).

• **Where**: Verification activities will take place in the QA testing environment.

## **4.2 Verification Criteria**

Define the criteria against which the software will be verified. These should align with the functional and quality requirements. Example:

The response time for a search query should be less than 3 seconds under normal load.

# **5. Appendices**

## **5.1 Assumptions and Dependencies**

(Mapped to 9.6.8 Assumptions and Dependencies)

List any assumptions and dependencies that impact the software development process or its requirements. Example:

The system depends on the availability of the university's student database for user authentication.

## **5.2 Acronyms and Abbreviations**

(Mapped to 9.6.4 Definitions)

Include a list of acronyms and abbreviations used in the document. Example:

**SaaS**: Software as a Service

## **5.3 Glossary (Optional Section)**

**Explain the purpose**:

Include a glossary if your project involves many domain-specific or technical terms. This section is especially useful when your system is used by non-technical users, stakeholders, or clients. It complements Section 1.4 (Definitions), but allows for a broader, more explanatory list of terms.