

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 3**](#_j56nxs9857ny)

[1.1 Purpose 3](#_75unuusrlw9s)

[1.2 Scope 3](#_c8eecaohfd4r)

[1.3 Product Overview 3](#_qe1l2gw8w8or)

[1.3.1 Product Perspective 4](#_niw6ypvcteih)

[1.3.1.1 System Interfaces 4](#_aa1tnkrpndcj)

[1.3.1.2 User Interfaces 5](#_75nmofwptx58)

[1.3.1.3 Hardware Interfaces 5](#_x1qs31k33ob7)

[1.3.1.4 Software Interfaces 5](#_bn2i2e7vlx9f)

[1.3.1.5 Communications Interfaces 5](#_gfhbbcdx91az)

[1.3.1.6 Memory Constraints 5](#_f880x5n2hq3b)

[1.3.1.7 Operations 5](#_vy33ihovwxgr)

[1.3.1.8 Site Adaptation Requirements 5](#_cjl8f87cjbzf)

[1.3.1.9 Interfaces with Services 5](#_v66ide574chl)

[1.3.2 Product Functions 6](#_tgduldkvba2j)

[1.3.3 User Characteristics 6](#_3dvdno6w3l47)

[1.3.4 Limitations 7](#_seply36radau)

[1.3.4.1 Regulatory Requirements and Policies 7](#_opqx2mhw68ew)

[1.3.4.2 Hardware Limitations 7](#_pjp0gizf6tk9)

[1.3.4.3 Interfaces to Other Applications 7](#_wpnzn9cberm3)

[1.3.4.4 Parallel Operation 8](#_gs2q7a3wercm)

[1.3.4.5 Audit Functions 8](#_58r4r3ni7at6)

[1.3.4.6 Control Functions 8](#_wxle84mi5j9x)

[1.3.4.7 Higher-Order Language Requirements 8](#_serxpn4a9ox4)

[1.3.4.8 Signal Handshake Protocols 8](#_lxz6pjm9dtjd)

[1.3.4.9 Quality Requirements 8](#_uiavwlfd9wol)

[1.3.4.10 Criticality of the Application 9](#_wyu9os6ee0x1)

[1.3.4.11 Safety and Security Considerations 9](#_ky9gotxdshpm)

[1.3.4.12 Physical/Mental Considerations 9](#_emx1fw9106xa)

[1.3.4.13 Real-Time Requirements 9](#_efevourvu9gy)

[1.4 Definitions 9](#_rwjbm381oy52)

[**2. References 10**](#_2bopy57ido95)

[**3. Requirements 10**](#_l9kkxtfvpdw6)

[3.1 Functions 10](#_9sd4k3jmxvho)

[3.2 Performance Requirements 13](#_ooak04jl1s4h)

[3.3 Usability Requirements 14](#_u4sokbljjjs0)

[3.4 Interface Requirements 14](#_chxynaauml5s)

[3.4.1 System Interfaces 14](#_g2v17drnge7q)

[3.4.2 User Interfaces 14](#_yulgj58k8o3z)

[3.4.3 Hardware Interfaces 14](#_y0ktqljwf03t)

[3.4.4 Software Interfaces 14](#_iapgsn18l6ob)

[3.4.5 Communications Interfaces 14](#_gb4h8guttzfy)

[3.5 Logical Database Requirements 15](#_uf7n4ed8gnko)

[3.6 Design Constraints 15](#_dezq6ys120ah)

[3.7 Software System Attributes 15](#_txxuwz4s49x0)

[3.8 Supporting Information 16](#_dks3tyy3dua)

[**4. Verification 16**](#_c4lq8go4u9ze)

[4.1 Verification Approach 16](#_vci46d1vmth)

[4.2 Verification Criteria 17](#_f0zvozls3dvb)

[**5. Appendices 17**](#_z0q6r66a5kr0)

[5.1 Assumptions and Dependencies 17](#_nwo8g3vykxu)

[5.2 Acronyms and Abbreviations 17](#_ujh9zc5b1lve)

[5.3 Glossary (Optional Section) 17](#_bm2fx37n9c1z)

# **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.*

The purpose of the “Campus Wellness Portal with Medical System and Fitness Center Integration(CWS)” is to empower students to take charge of their holistic well-being by providing a seamless digital platform that facilitates access to medical services, fitness programs and personalized wellness tracking tools.

This document is intended for system developer, project manager, configuration manager and client.

## **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.

The system is called “Campus Wellness Portal with Medical System and Fitness Center Integration(CWS)”. The system will manage the scheduling, tracking and coordination of campus health and fitness services for students. It provides the booking of health center appointments, fitness class registration, personalized wellness goal tracking, user role management and permission and notification module.

## **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.

The campus wellness portal is a system that provides an integrated digital platform for students to manage their health condition. The system allows users to:

1. Direct user to their particular dashboard based on their role
2. Book, cancel and reschedule appointments with the campus health center
3. Register and manage fitness classes and gym sessions
4. Set and track personalized wellness goals
5. Access notifications
6. Redeem prizes by using points earned from achieving wellness goals
7. Enable staff to manage schedules and view reports

### **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.

The Campus Wellness Portal is a subsystem within the University Campus Management System ecosystem which interacts with University Health Center Appointment System and Campus Recreation Facility Management Software. It acts as an integrator, providing a user-facing interface while leveraging backend data and functionalities from the integrated system.

#### **1.3.1.1 System Interfaces**

* Medical system: Interfaces with university medical system using REST APIs for appointment management and health report.
* Fitness center system: Interfaces with university fitness center system using REST APIs for class scheduling and facility usage.
* Notification system: Trigger-based reminders and updates.
* Authentication system: Single sign-on via university’s LDAP system.

#### **1.3.1.2 User Interfaces**

* Responsive web-based interface specified for different user roles
* Accessible to students, health center staff, fitness staff and administrators
* Must comply with WCAG 2.1 accessibility standards.

#### **1.3.1.3 Hardware Interfaces**

* Hosted on university-managed servers.
* Client’s hardware needs to support browsers.

#### **1.3.1.4 Software Interfaces**

* Use REST APIs to communicate with external systems (University’s health center system and fitness center system).
* Use the LDAP system which is implemented by the university for authentication.
* Data exchange using JSON and XML.

#### **1.3.1.5 Communications Interfaces**

* HTTPS-based encrypted communication.
* Push notifications managed via Firebase.

#### **1.3.1.6 Memory Constraints**

* Cloud-scalable memory management.
* Use asynchronous data loading and caching for memory optimization.
* Use lightweight data storage for goal tracking, class history and user data.

#### **1.3.1.7 Operations**

* System operates 24/7 with downtime scheduling via the admin interface.
* Logs user activities for compliance and monitoring.
* Includes admin tools for monitoring system usage and managing user accounts.

#### **1.3.1.8 Site Adaptation Requirements**

* Admin panel allows configuration of:
  + Facility information
  + Appointment types
  + Notification templates
* Support institutional branding and theme switching

#### **1.3.1.9 Interfaces with Services**

* Health and wellness content services for tailored resources.
* University wide notification services for alerts.

### **1.3.2 Product Functions**

List the primary functions of the software. Example:

Submit applications

* Login
* Book health appointment
* Book fitness class
* Book facility
* Receive notification
* View health report
* Cancel / Reschedule appointment
* Receive tailored health resource
* Track wellness progress
* Set wellness goal
* Manage facility
* Manage fitness class schedule
* View student’s wellness report
* Manage appointment

### **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.

| User | Technical Expertise | Usability Considerations |
| --- | --- | --- |
| Student | Low | Interface must be visually clear, mobile-responsive and intuitive. The guidance about how to use the system should be implemented. Users no need to be familiar with the system. |
| Fitness Center staff | Low | Interface should be easy to use, simple and efficient. |
| Wellness staff | Moderate | Users are expected to be familiar with the system. The interface should support quick access and minimize the complexity. |
| System administrator | High | Require full access to system settings, user management panels and monitoring dashboards. |

### **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.

This section will outline the constraints that may affect the design, development, deployment, functionality and performance of the CWP.

#### **1.3.4.1 Regulatory Requirements and Policies**

* University policy mandates integration with the campus SSO and use of official SMTP servers for notifications.
* The system must comply with following data privacy regulations:
  + HIPAA (for health related data)
  + APDP (for personal data)

#### **1.3.4.2 Hardware Limitations**

* The CWP will depends on the hosting infrastructure of the university which ensures:
  + Minimum 99.5% uptime
  + Adequate IOPs for appointment and facility search queries
* Since the system is developed as a web-interface application using a browser, it is limited to mobile phones and PCs with web browsers only, wearable devices are not able to run this system fluently.

#### **1.3.4.3 Interfaces to Other Applications**

* The system relies on external APIs from the Medical System and Fitness Center System of the university.
  + Any changes or downtime in those APIs will directly affect related functionalities such as class booking and appointment scheduling.
* API rate limits and schema changes from third-party providers such as notifications and fitness tracking services may impact performance and availability.

#### **1.3.4.4 Parallel Operation**

* The system must operate in parallel with the Medical System and Fitness Center System of the university.
* The portal must degrade gracefully and notify users in case of temporary unavailability of either external system.

#### **1.3.4.5 Audit Functions**

* The system must maintain audit logs of:
  + Logins, role changes and administrative actions
  + Appointment modifications and sensitive data access
* These logs must be retained securely and comply with audit trail retention policies for 5 years.

#### **1.3.4.6 Control Functions**

* The system has limited control over the external systems it depends on.
* The system cannot enforce business rules or user permissions on the Medical System and Fitness Center System.
* The portal can only control user roles and feature access within its own scope.

#### **1.3.4.7 Higher-Order Language Requirements**

* The portal is being developed in Python (Flask backend) and JavaScript (React frontend)
* The system will not support legacy languages or platforms.
* Integration SDKs for third-party devices must support JavaScript.

#### **1.3.4.8 Signal Handshake Protocols**

* API-level acknowledgements are essential for synchronous booking and cancellation features.

#### **1.3.4.9 Quality Requirements**

* Achieve a reliability target of 99.5% uptime during active semester weeks.
* Fail gracefully and inform users in case of backend issues.
* Provide accurate synchronization with external system data.

#### **1.3.4.10 Criticality of the Application**

* Moderately critical.

#### **1.3.4.11 Safety and Security Considerations**

* All data must be encrypted in transit (HTTPS) and at rest (AES-256)
* Role based control.
* The system must be protected against cross-site scripting (XSS), SQL injection and session hijacking.
* Two-factor authentication, 2FA may be mandated for administrative access.

#### **1.3.4.12 Physical/Mental Considerations**

* The system must meet accessibility standards of:
  + WCAG 2.1 Level AA compliance
  + Keyboard navigation
  + Screen-reader support
* Avoid triggering mental health issues such as no negative comments around BMI/fitness goals.

#### **1.3.4.13 Real-Time Requirements**

* The system’s responsiveness depends on:
  + Real-time updates from the Medical System
  + Live class capacity updates from Fitness Center System
* Data may be outdated if the external systems do not push real-time updates or expose polling endpoints.

## **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.

*Web Content Accessibility Guidelines (WCAG) 2.1*. (2025, May 6).

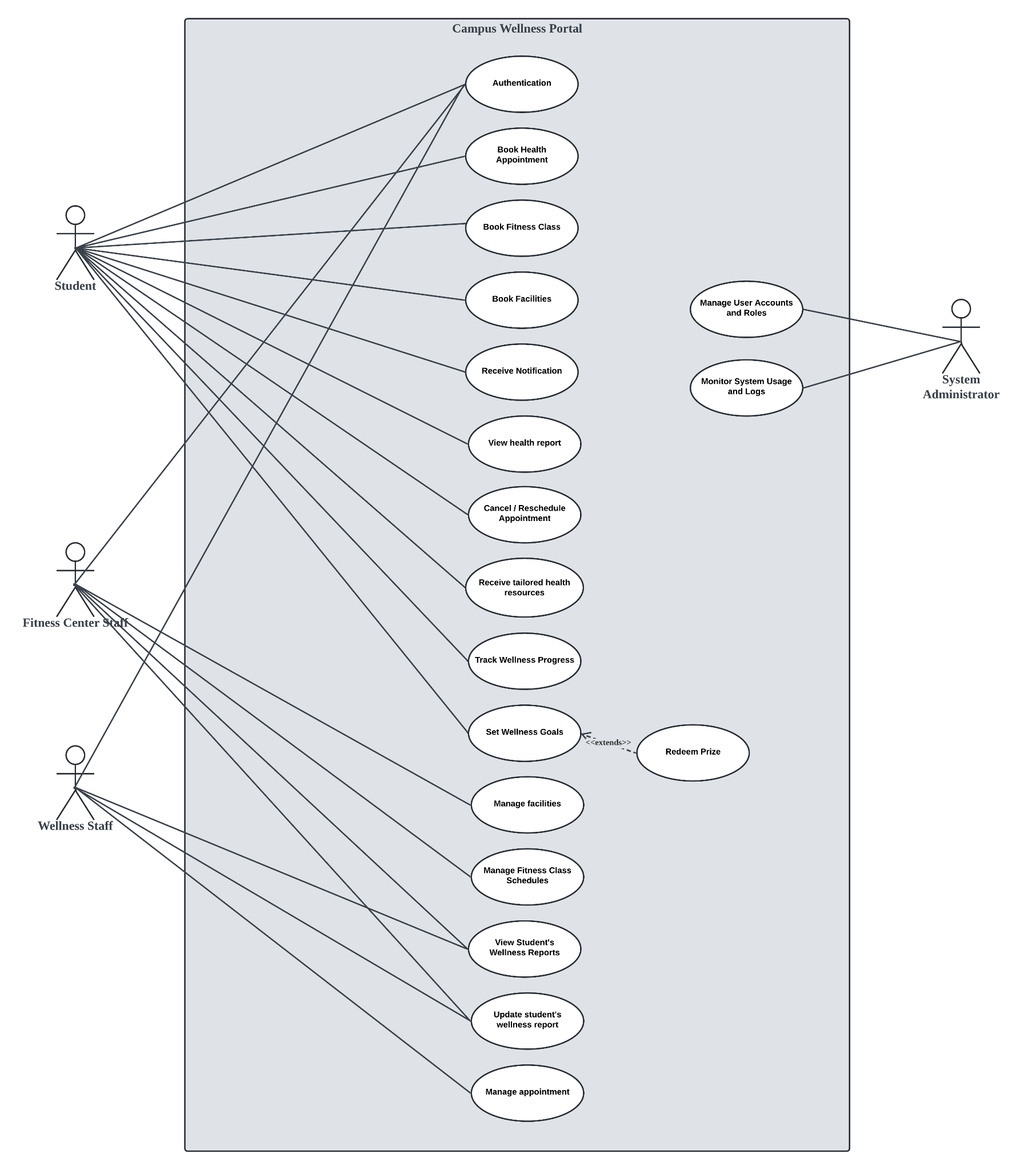
https://www.w3.org/TR/WCAG21/

# **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.

| Section | 3.1.1 |
| --- | --- |
| Name | Manage facility |
| Author | Yeoh Han Yi |
| Source | Fitness Center Staff (Stakeholder) |
| Short Description | Fitness Center Staff can manage campus fitness facilities, which include updating facility status and adding new facilities. |
| Goal | To ensure that the information about the available facilities are always accurate and updated in the system. |
| Actor | Fitness Center Staff |
| Pre-condition | 1. User is logged in 2. User’s role is Fitness Center Staff |
| Post-condition |  |
| Relationship to other use cases | Related to ‘Book facility’ use case |

| Section | 3.1.2 |
| --- | --- |
| Name | Yeoh Han Yi |
| Author | Manage fitness class schedule |
| Source | Fitness Center Staff (Stakeholder) |
| Short Description | Fitness Center Staff can manage the schedule of the fitness classes for students to view and book. |
| Goal | To provide a flexible and accurate schedule for fitness classes so that students could view and book effectively. |
| Actor | Fitness Center Staff |
| Pre-condition | 1. User is logged in 2. User’s role is Fitness Center Staff |
| Post-condition |  |
| Relationship to other use cases | Related to ‘Book fitness class’ use case |

| Section | 3.1.3 |
| --- | --- |
| Name |  |
| Author |  |
| Source |  |
| Short Description |  |
| Goal |  |
| Actor |  |
| Pre-condition |  |
| Post-condition |  |
| Relationship to other use cases |  |

## **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.