Software Requirements Specification (SRS) Guidance based on ISO/IEC/IEEE 29148:2018

This document is just an extra guidance/suggestion on how to write the Software Requirements Specification (SRS) following the outline in Figure 8 and the detailed description in 9.6 of ISO/IEC/IEEE 29148:2018. Each section and subsection includes practical descriptions and examples to help students apply the standard effectively.

1. **Introduction**
   1. **Purpose**

The aim of the Campus Accessibility Navigation System with Facilities and Event Integration is to have a robust navigation system that enables users to chart accessible routes across campus. By utilizing real-time information from the facilities management database and events calendar, the system addresses problems such as construction advisories, elevator closures, and temporary accommodations for events, thus improving campus accessibility and user safety. For example, this software system caters to campus mobility problems by facilitating students, staff, and visitors in adaptive route planning that responds to changing campus conditions and aiding special needs users.

* 1. **Scope**

By combining data from several internal sources, the system is intended to make campus navigation simple for users. Offering accessible routes, providing information regarding maintenance or construction projects, and modifying routes in response to forthcoming campus activities are some of its primary advantages. It only includes the areas of campus where we have information on events and facilities. It may, for instance, create a completely accessible route from beginning to end, alert users to potential impediments, and quickly modify the recommended route in the event that campus facilities or event schedules change.

* 1. **Product Overview**

The Campus Accessibility Navigation System with Facilities and Event Integration is a web-based application designed to help students, staff, and visitors, especially those with mobility disabilities, who navigate the university campus efficiently. It offers accessible routes and interfaces with campus systems to display essential facilities such as ramps, lifts, toilets, and event venues. It also enables users to view scheduled campus events and give corresponding location-based notifications. The system enhances campus convenience and inclusiveness by integrating accessibility and facility information into one system.

* + 1. **Product Perspective**

Campus Accessibility Navigation System is a sub-system of the general university digital infrastructure. It is an online service module implemented in the university campus portal. The system communicates with the key external systems like the facilities management database, campus events calendar, and the university digital ID authentication system. These communications enable the application to provide role-based, context-aware navigation based on user roles and dynamic campus conditions. A top-level context diagram (see Appendix A) illustrates the system, user group, and data source interaction, thus framing the stage for efficient data exchange and system interoperability.

* + 1. **Product Functions**

The major functions of the Campus Accessibility Navigation System are:

* Accessible Route Planning: Offers step-by-step real-time navigational routes that highlight accessibility features such as ramps, elevators, and tactile pathways.
* Real-Time Warnings: Offers alerts to users regarding real-time updates such as facility shutdown, temporary closure, or changes in routes due to real-time events or maintenance.
* User Feedback Integration: Offers a user interface for users to report navigation issues, suggest amendments, or flag-of-limit routes, which will be considered for future changes.
* Facility and Event Mapping: Displays locations of significant facilities (e.g., restrooms, prayer areas, parking areas) and maps to the university event system to display accessible event locations.
* User Profile Adaptation: Will adapt navigation suggestions according to user profiles (e.g., wheelchair users, visually impaired users) to enhance usability and customization.
  + 1. **User Characteristics**

This system is designed to serve a multidevice user population with the following:

* Students who rely on accessible paths to attend lectures, labs, and other activities across campus.
* Students utilizing easily accessible pathways to reach class, lab, and other campus activities.
* Faculty and staff members who may be able to use accessible pathway or must report facility issues related to access.
* Prospective students, parents, or event attendees that require easy and accessible entrance to areas of the campus.

All the users ought to possess basic digital literacy in order to work with the web interface. However, the system is implemented based on accessibility-first principles to serve users with various disabilities and has assistive design features such as voice guidance, support for screen readers, and high-contrast display options.

* + 1. **Limitations**

Notwithstanding the efforts of the system to give accurate and helpful navigation, the following limitations apply:

* The effectiveness of the system depends on data timeliness and accuracy retrieved from external databases such as the events and facilities databases.
* It is geospatially constrained to the campus of the university and excludes off-campus addresses or external navigation aids such as Google Maps.
* Physical constraints like low-quality GPS signal indoors can impact the accuracy of location, particularly within multi-story buildings.
* Performance can briefly reduce somewhat during times of intense campus use (e.g., during semester beginnings, large events) due to increased system resource utilization.
  1. **Definitions**

This part provides definitions to the key terms used throughout the Software Requirements Specification (SRS), to achieve common understanding among all stakeholders, including developers, testers, users, and maintainers. Definitions eliminate ambiguity and assist in consistent interpretation of system features and requirements.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| **Accessible Route** | One designed for use by individuals with disabilities, including ramps, elevators, and no steps. |
| **Real-Time Data** | Data delivered immediately or with minimal delay, such as live event info or facility status. |
| **Digital ID Verification** | Authenticates a user's identity through university-issued digital credentials. |
| **Facility Availability** | The current operational status of campus facilities like elevators or accessible restrooms. |
| **Event Integration** | Links campus activities to the navigation system for routing to activity locations. |
| **Navigation System** | Guides users around campus using accessible paths and Points of Interest (POIs). |
| **Point of Interest (POI)** | Key campus locations like lecture halls, restrooms, and entrances. |
| **Accessibility Feedback Module** | Allows users to report inaccessibility or suggest improvements. |
| **User Role** | User type (e.g., Student, Staff, Visitor) that determines access to system features. |

1. **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](http://www.iso.org/standard/72089.html)

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

1. **Requirements**
   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.



## List of Use Case Specifications

**1. View Accessible Routes**

Actor: User

Description:

The user views accessible routes across the campus.

Preconditions:

User is logged in.

Postconditions:

User sees the list of accessible routes.

Basic Flow:

1. User requests to view accessible routes.
2. System fetches available accessible routes from the database.
3. System displays accessible routes to the user.

Alternative Flows:

None.

**2. Plan Route to Destination**

Actor: User

Description:

The user plans an accessible route to their desired destination.

Preconditions:

User is logged in and has entered a valid destination.

Postconditions:

User receives an accessible route plan.

Basic Flow:

1. User selects a destination on the campus map.
2. System calculates the best accessible route, considering real-time data.
3. System displays the route and estimated travel time.

Alternative Flows:

If no accessible route is found, the system notifies the user and suggests alternative paths or assistance.

**3. Receive Real-time Alerts**

Actor: User

Description:

The system sends real-time alerts (e.g., construction, elevator outages) to the user.

Preconditions:

User is logged in and has enabled notifications.

Postconditions:

User is notified about any real-time changes.

Basic Flow:

1. System detects real-time issues (e.g., construction).
2. System pushes notifications to users.
3. User receives alerts and can view affected routes.

Alternative Flows:

User disables alerts in their profile settings.

**4. Access Event-based Route Changes**

Actor: User

Description:

User can view temporary route changes due to events.

Preconditions:

User is logged in.

Postconditions:

User sees updated routes for events.

Basic Flow:

1. User requests information about event-based route changes.
2. System retrieves event data and adjusts routes accordingly.
3. System displays updated route suggestions.

Alternative Flows:

None.

**5. Update Facility Information**

Actor: Admin

Description:

Admin updates facility data such as elevator status, accessible entrances, etc.

Preconditions:

Admin is logged in with proper authorization.

Postconditions:

Facility information is updated in the database.

Basic Flow:

1. Admin selects the facility to update.
2. Admin enters the new facility information.
3. System validates and saves the updates.
4. System notifies users if updates impact route accessibility.

Alternative Flows:

Admin input errors (e.g., missing data), system prompts for correction.

**6. Update Event Information**

Actor: Admin

Description:

Admin updates event details that affect accessibility.

Preconditions:

Admin is logged in and authorized.

Postconditions:

Event information is updated in the database.

Basic Flow:

1. Admin selects an event to update.
2. Admin enters updated event information (e.g., event location, time, temporary closures).
3. System saves the updated event data.
4. System updates related route data for users.

Alternative Flows:

None.

**7. Manage Accessibility Data**

Actor: Admin

Description:

Admin manages accessibility information, including accommodations and route features.

Preconditions:

Admin is logged in and has access rights.

Postconditions:

Accessibility data is updated.

Basic Flow:

1. Admin accesses accessibility management interface.
2. Admin updates data like accessible entrances, ramps, and temporary accommodations.
3. System saves updates and reflects them in route planning.

Alternative Flows:

None.

**8. Generate Temporary Accommodations**

Actor: Admin

Description:

Admin generates temporary accommodations for events or construction (e.g., temporary ramps).

Preconditions:

Admin is logged in.

Postconditions:

Temporary accommodations are saved and integrated into route planning.

Basic Flow:

1. Admin identifies areas needing temporary accommodations.
2. Admin enters details of accommodations (location, duration, type).
3. System updates the accessibility database.
4. Users see updated routes that consider temporary accommodations.

Alternative Flows:

None.

* 1. **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.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Metric | Validation Method |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. **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.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requirement | Metric | Validation Method |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

* 1. **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.

* 1. **Logical Database Requirements**

**(Mapped to 9.6.15 Logical Database Requirements)**

Describe key data entities, relationships, and constraints. This could include an Entity- Relationship (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.

* 1. **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.

* 1. **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.
  1. **Supporting Information**

**(Mapped to 9.6.20 Supporting Information)**

Any additional supporting information, including:

1. sample input/output formats, descriptions of cost analysis studies or results of questionnaires or any other elicitation techniques;
2. supporting or background information that can help the readers of the SRS;
3. a description of the problems to be solved by the software; and
4. 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).

1. **Verification**
   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.
  1. **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.

1. **Appendices**
   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.

* 1. **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

* 1. **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.