Software Requirements Specification: Realm Software Engineering

Team #13, ARC
Avanish Ahluwalia
Russell Davidson
Rafey Malik
Abdul Zulfiqar

Contents

Revision History

Date	Version	Notes
yyyy-mm-dd	1.0	Notes
yyyy-mm-dd	1.1	Notes

1 Introduction

[This section should provide an overview of the entire document —SS]

1.1 Document Purpose

[Describe the purpose of the SRS and its intended audience. —SS]

1.2 Product Scope

[Identify the product whose software requirements are specified in this document, including the revision or release number. Explain what the product that is covered by this SRS will do, particularly if this SRS describes only part of the system or a single subsystem. Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here. —SS]

1.3 Definitions, Acronyms and Abbreviations

1.4 References

[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. —SS]

1.5 Document Overview

[Describe what the rest of the document contains and how it is organized. —SS]

2 Product Overview

[This section should describe the general factors that affect the product and its requirements. This section does not state specific requirements. Instead, it provides a background for those requirements, which are defined in detail in Section 3, and makes them easier to understand. —SS]

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. —SS]

2.2 Product Functions

[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. —SS]

2.3 Product Constraints

[This subsection should provide a general description of any other items that will limit the developer's options. These may include:

- Interfaces to users, other applications or hardware.
- Quality of service constraints.
- Standards compliance.
- Constraints around design or implementation.

—SS]

2.4 User 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. —SS]

2.5 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). —SS]

2.6 Apportioning of Requirements

[Apportion the software requirements to software elements. For requirements that will require implementation over multiple software elements, or when allocation to a software element is initially undefined, this should be so stated. A cross reference table by function and software element should be used to summarize the apportioning.
—SS]

[Identify requirements that may be delayed until future versions of the system (e.g., blocks and/or increments). —SS]

3 Requirements

[This section specifies the software product's requirements. Specify all of the software requirements to a level of detail sufficient to enable designers to design a software system to satisfy those requirements, and to enable testers to test that the software system satisfies those requirements. The specific requirements should:

- Be uniquely identifiable.
- State the subject of the requirement (e.g., system, software, etc.) and what shall be done.
- Optionally state the conditions and constraints, if any.
- Describe every input (stimulus) into the software system, every output (response) from the software system, and all functions performed by the software system in response to an input or in support of an output.
- Be verifiable (e.g., the requirement realization can be proven to the customer's satisfaction)
- Conform to agreed upon syntax, keywords, and terms.

—SS]

3.1 External Interfaces

[This subsection defines all the inputs into and outputs requirements of the software system. Each interface defined may include the following content:

- Name of item
- Source of input or destination of output
- Valid range, accuracy, and/or tolerance
- Units of measure
- Timing
- Relationships to other inputs/outputs
- Screen formats/organization
- Window formats/organization
- Data formats
- Command formats
- End messages

—SS]

3.1.1 User interfaces

[Define the software components for which a user interface is needed. 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. Details of the user interface design should be documented in a separate user interface specification. —SS]

[Could be further divided into Usability and Convenience requirements. —SS]

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. —SS]

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. —SS]

3.2 Functional

[This section specifies the requirements of functional effects that the software-to-be is to have on its environment. —SS]

3.3 Quality of Service

[This section states additional, quality-related property requirements that the functional effects of the software should present. —SS]

3.3.1 Performance

[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. —SS]

3.3.2 Security

[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. —SS]

3.3.3 Reliability

[Specify the factors required to establish the required reliability of the software system at time of delivery. —SS]

3.3.4 Availability

[Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. —SS]

3.4 Compliance

[Specify the requirements derived from existing standards or regulations, including:

- Report format
- Data naming
- Accounting procedures
- Audit tracing

—SS]

[For example, this could specify the requirement for software to trace processing activity. Such traces are needed for some applications to meet minimum regulatory or financial standards. An audit trace requirement may, for example, state that all changes to a payroll database shall be recorded in a trace file with before and after values. —SS]

3.5 Design and Implementation

3.5.1 Installation

[Constraints to ensure that the software-to-be will run smoothly on the target implementation platform. —SS]

3.5.2 Distribution

[Constraints on software components to fit the geographically distributed structure of the host organization, the distribution of data to be processed, or the distribution of devices to be controlled. —SS]

3.5.3 Maintainability

[Specify attributes of software that relate to the ease of maintenance of the software itself. These may include requirements for certain modularity, interfaces, or complexity limitation. Requirements should not be placed here just because they are thought to be good design practices. —SS]

3.5.4 Reusability

3.5.5 Portability

[Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. —SS]

3.5.6 Cost

[Specify monetary cost of the software product. —SS]

3.5.7 Deadline

[Specify schedule for delivery of the software product. —SS]

3.5.8 Proof of Concept

4 Verification

[This section provides the verification approaches and methods planned to qualify the software. The information items for verification are recommended to be given in a parallel manner with the requirement items in Section 3. The purpose of the verification process is to provide objective evidence that a system or system element fulfills its specified requirements and characteristics. —SS]

5 Appendixes

5.1 Reflection

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning. Please answer the following questions:

1. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

2. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?