ÇANKAYA UNIVERSITY

Software Engineering Department

SENG 491 – 492 Graduation Project

Software Requirements

Specification

<<Project Name>>

<<Project Members>>

Version <<1.0>>

Table of Contents

1   Introduction

1.1   Purpose

This subsection should
  a)   Delineate the purpose of the SRS;
  b)   Specify the intended audience for the SRS.

1.2   Scope

This subsection should
  a)   Identify the software product(s) to be produced by name (e.g., Host DBMS, Report Generator, etc.);
  b)   Explain what the software product(s) will, and, if necessary, will not do;
  c)   Describe the application of the software being specified, including relevant benefits, objectives, and goals;
  d)   Be consistent with similar statements in higher-level specifications (e.g., the system requirements specification), if they exist.

1.3   Definitions, acronyms, and abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

1.4   References

This subsection should
  a)   Provide a complete list of all documents referenced elsewhere in the SRS;
  b)   Identify each document by title, report number (if applicable), date, and publishing organization;
  c)   Specify the sources from which the references can be obtained.

If the list of references is too long, it should be provided at the end of the document as a separate section.

1.5   Overview

This subsection should
  a)   Describe what the rest of the SRS contains;
  b)   Explain how the SRS is organized.

1.6   Version History

Provide a table of table of changes to this document

Version No	Description of change	Date
1.0	Initial Release	15.10.2024

2   Overall description

2.1   Product perspective

This subsection of the SRS should put the product into perspective with other related products. If the product is independent and totally self-contained, it should be so stated here. If the SRS defines a product that is a component of a larger system, as frequently occurs, then this subsection should relate the requirements of that larger system to functionality of the software and should identify interfaces between that system and the software.

A block diagram showing the major components of the larger system, interconnections, and external inter- faces can be helpful.

This subsection should also describe how the software operates inside various constraints. For example, these constraints could include

a)   System interfaces;
b)   User interfaces;
c)   Hardware interfaces;
d)   Software interfaces;
e)   Communications interfaces;
f)   Memory;
g)   Operations;
h)   Site adaptation requirements.

2.1.1   System interfaces

This subsection should list each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system.

2.1.2   User interfaces

This should specify the following:
a)   The logical characteristics of each interface between the software product and its users. This includes those configuration characteristics (e.g., required screen formats, page or window layouts, content of any reports or menus, or availability of programmable function keys) necessary to accomplish the software requirements.
b)   All the aspects of optimizing the interface with the person who must use the system. This may simply comprise a list of do's and don'ts on how the system will appear to the user. One example may be a requirement for the option of long or short error messages. Like all others, these requirements should be verifiable, e.g., "a clerk typist grade 4 can do function X in Z min after 1 h of training" rather than "a typist can do function X." (This may also be specified in the Software System Attributes under a section titled Ease of Use.)

2.1.3   Hardware interfaces

This should specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics (number of ports, instruction sets, etc.). It also covers such matters as what devices are to be supported, how they are to be supported, and protocols. For example, terminal support may specify full-screen support as opposed to line-by-line support.

2.1.4   Software interfaces

This should specify the use of other required software products (e.g., a data management system, an operating system, or a mathematical package), and interfaces with other application systems (e.g., the linkage between an accounts receivable system and a general ledger system). For each required software product, the following should be provided:

- Name;
- Mnemonic;
- Specification number;
- Version number;

For each interface, the following should be provided:

- Discussion of the purpose of the interfacing software as related to this software product.
- Definition of the interface in terms of message content and format. It is not necessary to detail any well-documented interface, but a reference to the document defining the interface is required.

2.1.5   Communications interfaces

This should specify the various interfaces to communications such as local network protocols, etc.

2.1.6   Memory constraints

This should specify any applicable characteristics and limits on primary and secondary memory.

2.1.7   Operations

This should specify the normal and special operations required by the user such as

a) The various modes of operations in the user organization (e.g., user-initiated operations);
b) Periods of interactive operations and periods of unattended operations;
c) Data processing support functions;
d) Backup and recovery operations.

NOTE-This is sometimes specified as part of the User Interfaces section.

2.1.8 Site adaptation requirements

This should

a) Define the requirements for any data or initialization sequences that are specific to a given site, mission, or operational mode (e.g., grid values, safety limits, etc.);
b) Specify the site or mission-related features that should be modified to adapt the software to a particular installation.

2.2 Product functions

This subsection of the SRS should provide a summary of the major functions that the software will perform. For example, an SRS for an accounting program may use this part to address customer account maintenance, customer statement, and invoice preparation without mentioning the vast amount of detail that each of those functions requires.

Sometimes the function summary that is necessary for this part can be taken directly from the section of the higher-level specification (if one exists) that allocates particular functions to the software product. Note that for the sake of clarity

a) The functions should be organized in a way that makes the list of functions understandable to the customer or to anyone else reading the document for the first time.
b) Textual or graphical methods can be used to show the different functions and their relationships. Such a diagram is not intended to show a design of a product, but simply shows the logical relation- ships among variables.

2.3 User characteristics

This subsection of the SRS should describe those general characteristics of the intended users of the product including educational level, experience, and technical expertise. It should not be used to state specific requirements, but rather should provide the reasons why certain specific requirements are later specified in Section 3 of the SRS.

2.4 Constraints

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

a) Regulatory policies;
b) Hardware limitations (e.g., signal timing requirements);
c) Interfaces to other applications;
d) Parallel operation;

e) Audit functions;
f) Control functions;
g) Higher-order language requirements;
h) Signal handshake protocols (e.g., XON-XOFF, ACK-NACK);
i) Reliability requirements;
j) Criticality of the application;
k) Safety and security considerations.

2.5 Assumptions and dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption may be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

2.6 Apportioning of requirements

This subsection of the SRS should identify requirements that may be delayed until future versions of the system.

3 Specific Requirements

This section of the SRS should contain all of the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements. Throughout this section, every stated requirement should be externally perceivable by users, operators, or other external systems. These requirements should include at a minimum a description of every input (stimulus) into the system, every output (response) from the system, and all functions performed by the system in response to an input or in support of an output. As this is often the largest and most important part of the SRS, the following principles apply:

a) Specific requirements should be stated in conformance with all the characteristics described in 4.3 of the standards.
b) Specific requirements should be cross-referenced to earlier documents that relate.
c) All requirements should be uniquely identifiable.
d) Careful attention should be given to organizing the requirements to maximize readability.

3.1 External Interface Requirements

This should be a detailed description of all inputs into and outputs from the software system. It should complement the interface descriptions in section 2 and should not repeat information there.

It should include both content and format as follows:

a) Name of item;
b) Description of purpose;

c)   Source of input or destination of output;
d)   Valid range, accuracy, and/or tolerance;
e)   Units of measure;
f)   Timing;
g)   Relationships to other inputs/outputs;
h)   Screen formats/organization;
i)   Window formats/organization;
j)   Data formats;
k)   Command formats;
l)   End messages.

3.1.1   User interfaces

3.1.2   Hardware interfaces

3.1.3   Software interfaces

3.1.4   Communications interfaces

3.2   Functional Requirements

Functional requirements should define the fundamental actions that must take place in the software in accepting and processing the inputs and in processing and generating the outputs. These are generally listed as "shall" statements starting with "The system shall."

These include

a)   Validity checks on the inputs
b)   Exact sequence of operations
c)   Responses to abnormal situations, including
   1)   Overflow
   2)   Communication facilities
   3)   Error handling and recovery
d)   Effect of parameters
e)   Relationship of outputs to inputs, including
   1)   Input/output sequences
   2)   Formulas for input to output conversion

It may be appropriate to partition the functional requirements into subfunctions or subprocesses. This does not imply that the software design will also be partitioned that way.

3.2.1 User class 1

3.2.1.1 Feature 1.1

3.2.1.1.1 Introduction/Purpose of feature

3.2.1.1.2 Stimulus/Response sequence

3.2.1.1.3 Associated functional requirements

3.2.1.2 Feature 1.2

3.2.1.2.1 Introduction/Purpose of feature

3.2.1.2.2 Stimulus/Response sequence

3.2.1.2.3 Associated functional requirements

.
.
.

3.2.2 User class 2

.
.
.

3.3 Performance requirements

This subsection should specify both the static and the dynamic numerical requirements placed on the software or on human interaction with the software as a whole. Static numerical requirements may include the following:

a) The number of terminals to be supported;
b) The number of simultaneous users to be supported;
c) Amount and type of information to be handled.

Static numerical requirements are sometimes identified under a separate section entitled Capacity.

Dynamic numerical requirements may include, for example, the numbers of transactions and tasks and the amount of data to be processed within certain time periods for both normal and peak workload conditions.

All of these requirements should be stated in measurable terms. For example,

95% of the transactions shall be processed in less than 1 *s.*

rather than,

An operator shall not have to wait for the transaction to be completed.

NOTE-Numerical limits applied to one specific function are normally specified as part of the processing subparagraph description of that function.

3.4   Logical database requirements

This should specify the logical requirements for any information that is to be placed into a database. This may include the following:

a)   Types of information used by various functions;
b)   Frequency of use;
c)   Accessing capabilities;
d)   Data entities and their relationships;
e)   Integrity constraints;
f)   Data retention requirements.

3.5   Design constraints

This should specify design constraints that can be imposed by other standards, hardware limitations, etc.

3.5.1   Standards compliance

This subsection should specify the requirements derived from existing standards or regulations. They may include the following:

a)   Report format;
b)   Data naming;
c)   Accounting procedures;
d)   Audit tracing.

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 must be recorded in a trace file with before and after values.

3.6   Software system attributes

There are a number of attributes of software that can serve as requirements. It is important that required attributes be specified so that their achievement can be objectively verified.

3.6.1   Reliability

This should specify the factors required to establish the required reliability of the software system at time of delivery.

3.6.2   Availability

This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

3.6.3   Security

This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to

a)   Utilize certain cryptographical techniques;
b)   Keep specific log or history data sets;
c)   Assign certain functions to different modules;
d)   Restrict communications between some areas of the program;
e)   Check data integrity for critical variables.

3.6.4   Maintainability

This should specify attributes of software that relate to the ease of maintenance of the software itself. There may be some requirement for certain modularity, interfaces, complexity, etc. Requirements should not be placed here just because they are thought to be good design practices.

3.6.5   Portability

This should specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems. This may include the following:

a)   Percentage of components with host-dependent code;
b)   Percentage of code that is host dependent;
c)   Use of a proven portable language;
d)   Use of a particular compiler or language subset;
e)   Use of a particular operating system.

4   References

The resources used to produce this document should be listed here and referenced in the relevant text of this document

[1] Software Engineeering Department, "Graduation Projects," SENG, [Online]. Available: https://seng.cankaya.edu.tr/graduation-projects/. [Accessed 28 June 2024].