****

**Software Requirements Specification**

**Project Name**

**Team Number**

**Name Surname, Student ID**

**Name Surname, Student ID**

**Name Surname, Student ID**

**Name Surname, Student ID**

**Name Surname, Student ID**

Bilkent University

Department of Information Systems and Technologies

06.11.2024

Change History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **File Name** | **Document Type** | **Deliverable** | **Version** | **Submission Date** |
| Deliverable\_2.docx | MS Word 2013 | 2 | 0.1 | - |
| Deliverable\_2\_2.docx | MS Word 2013 | 2 | 1 | - |
| TeamX\_SRS\_FR1 | MS Word 2013 | 2 | 2 | 06.11.2024 |
|  |  |  |  |  |

Project Team

|  |  |  |
| --- | --- | --- |
| **Name, Surname** | **Student Id Number** | **e-mail** |
| Umut Hüseyinoğlu | 12345678 | uhus@bilkent.edu.tr |
|  |  | Use the email that you regularly check! |
|  |  |  |
|  |  |  |

Project Details

|  |  |
| --- | --- |
| **Project Name** | **Project Name** |
| **Software Name** | **Software (Product) Name, if the same with the project name, repeat it** |
| **Company Name** | **(if there is a sponsor company and it is applicable, otherwise state “No sponsor company”)** |
| **Academic Advisor** | **Instructor Name, with full title** |
| **Github URL** | **Github URL** |
| **WEB page** | **(if applicable, if not leave it empty)** |

Executive Summary

Text, exactly 1 page (minimum and maximum 1 page), no images.

The SRS Executive Summary should consist of only one (1) paragraph.

This section provides a brief overview of your SRS and your findings, major requirements and most important things that your Executive should notice / read.

Table of Contents

**Page Number**

The table of contents should be automatically created by MS Word.

List of Tables

**Page Number**

The list of tables (if any) should be automatically created by MS Word.

If no Tables exist on your document, delete this page.

List of Figures

**Page Number**

The list of figures (if any) should be automatically created by MS Word.

If no Figures exist on your document, delete this page.

Abbreviations

|  |  |
| --- | --- |
| SRS | Software Requirements Specification |
| *Abbreviations need to be alphabetically ordered* |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Scope

Your whole document’s paragraph text should be with font face Cambria, font size 12pt, line spacing of 1.5, paragraphs justified.

Briefly describe the scope of the software under consideration by

a) Identifying the software product(s) to be produced by name (e.g., Host DBMS, Report Generator, etc.);

b) Explaining briefly what the software product(s) will do;

c) Describing the application of the software being specified, including relevant benefits, objectives, and goals;

d) Being consistent with similar statements in higher-level specifications (e.g., the requirements specification in the Initial Plan).

Make appropriate references to your Initial Plan, as required.

# Product Perspective

Define the system's relationship to other related products.

If the product is an element of a larger system, then relate the requirements of that larger system to the functionality of the product covered by the SRS.

If the product is an element of a larger system, then identify the interfaces between the product covered by the SRS and the larger system of which the product is an element.

A block diagram showing the major elements of the larger system, interconnections, and external interfaces can be helpful.

Describe how the software operates within the following constraints. However, not all of the following constraints may be applicable for your product, if this is the case just state “Not Applicable” and a small explanation text why it is not applicable.

1. **System interfaces;**

List each system interface and identify the functionality of the software to accomplish the system requirement and the interface description to match the system.

1. **User interfaces;**

Specify the following:

i) 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.

ii) All the aspects of optimizing the interface with the person who uses, maintains, or provides other support to 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.

1. **Hardware interfaces;**

Specify the logical characteristics of each interface between the software product and the hardware elements 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.

1. **Software interfaces;**

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, specify (if applicable):

a) Name;

b) Mnemonic;

c) Specification number;

d) Version number;

e) Source.

For each interface specify:

a) Discussion of the purpose of the interfacing software as related to this software product.

b) 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.

1. **Communications interfaces;**

Specify the various interfaces to communications such as local network protocols.

1. **Memory constraints;**

Specify any applicable characteristics and limits on primary and secondary memory.

# User characteristics

Describe those general characteristics of the intended groups of users of the product including characteristics that may influence usability, such as educational level, experience, disabilities, and technical expertise. Remember that your users and their characteristics will be a major factor for shaping your requirements.



Figure 1 This is a sample Figure. Delete it. Make sure that the Figure captions need to be below the figure

Table 1 Sample Table - Delete and use this table format where appropriate – Make sure that the table caption is auto-inserted and at the top of the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Info** | **Project A** | **Project B** | **Project C** | **Project D** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Assumptions and dependencies

List each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but any changes to these factors 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.

# Functional requirements

This section should consist at least two subsections, 1) a list of functional requirements in a textual format, 2) use case diagram of the functional requirements, 3) mapping table of functional requirements to the use case diagram.

In subsection 1), specify all of the functional software requirements to a level of detail sufficient to enable designers to design a software system to satisfy those requirements. Be careful to use the terminology and language format specified in the ISO/IEC/IEEE 29148:2018(E) (and our Lecture Notes regarding User Requirements).

Specify all of the software requirements to a level of detail sufficient to enable testers to test that the software system satisfies those requirements.

At a minimum, 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.

In subsection 2), it is mandatory for you to use Use Case diagram(s) with the appropriate UML template. There is no need for you to provide Use Case descriptions, just the Use Case diagram(s) will be enough.

In subsection 3) you will link your Functional Requirements in subsection 1 with the use cases in subsection 2 and the high level requirements in Initial Plan. Provide a table with two columns, in which one column will be Functional Requirement ID and the second column will be the Use Case ID. The third column will be a reference to the Functional Requirement in the Initial Plan. An example table is given below. Remember that a Functional Requirement may be related to multiple Use Cases and a Use Case may be related to multiple Functional Requirements.

Table 2 Cross Referencing Functional Requirements to Use Cases

|  |  |  |
| --- | --- | --- |
| Functional Requirement ID | Use Case ID | Initial Plan Functional Requirement ID [1] |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

The specific requirements should:

a) Be stated in conformance with all the characteristics described in subclause 5.2 of the ISO/IEC/IEEE 29148:2018(E).

b) Be cross-referenced to Initial Plan.

c) Be uniquely identifiable with the use of an appropriate numbering system.

The specific use cases should be identifiable with the use of an appropriate numbering system.

# System Model

For the Functional Requirements / Use Cases that you developed in Section 5, identify two complex Use Cases, and draw the following logical and object-oriented diagrams. Please note that this diagrams should be at the Analysis Level and not the Design Level.

--- Activity Diagrams (with partitions / swim-lanes) or State Diagrams

--- Analysis Level System Sequence Diagrams or Communication Diagrams

--- Analysis Level Class Diagram

# Requirements Prototypes

In this section you are required to select two Use Cases and model them with coding or with the use of prototyping tools (such as: <https://www.trustradius.com/prototyping#products> ). Please ask your Project Supervisor / Sponsor Company which Use Cases that they would like to see the prototype.

Unless there is a specific reason, you SHOULD prototype the two (most) complex Use Cases that you have used to develop the UML System Models (Section 6). If you have not used these two to prototype the two Use Cases in this section, explain in detail your reason.

Your prototype should consist of screenshots / visuals of your software, accompanied with detailed textual description and explanation. Any prototype section which does not explain adequately the visuals given, will automatically receive 0 points.

The aim of the Requirements Prototype is that it will make it easier for me and your project supervisor (and any interested stakeholder, e.g. sponsor company) to envision the finished product.

# Non-functional Requirements

This section describes the most important non-functional requirements that are applicable for your project. It is important to remember than NOT ALL non-functional requirements given in subsection 8.3 are going to be of consideration for you. Therefore, if a specific non-functional requirement subsection below is not appropriate / applicable for your product, please state that and explain briefly why. For the rest subsections, follow the explanations given below.

Do not forget, filling more subsections in 8.3 will not get you extra points. You will receive full points if your non-functional requirement subsections in 8.3 are appropriately selected and filled.

## Usability requirements

Define usability (quality in use) requirements. Usability requirements and objectives for the software system include measurable effectiveness, efficiency, and satisfaction criteria in specific contexts of use.

## Performance requirements

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.

The performance requirements should be stated in measurable terms.

For example, “95 % of the transactions shall be processed in less than 1 second.” rather than, “an operator shall not have to wait for the transaction to complete”.

## Software system attributes

Specify the required attributes of the software product. The following is a partial list of examples:

a) Reliability - Specify the factors required to establish the required reliability of the software system at time of delivery.

b) Availability - Specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart.

c) Security - Specify the requirements to protect the software from accidental or malicious access, use modification, destruction, or disclosure. Specific requirements in this area could include the need to:

1) Utilize certain cryptographic techniques;

2) Keep specific log or history data sets;

3) Assign certain functions to different modules;

4) Restrict communications between some areas of the program;

5) Check data integrity for critical variables;

6) Assure data privacy.

d) 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.

e) Portability - Specify attributes of software that relate to the ease of porting the software to other host machines and/or operating systems, including:

1) Percentage of elements with host-dependent code;

2) Percentage of code that is host dependent;

3) Use of a proven portable language;

4) Use of a particular compiler or language subset;

5) Use of a particular operating system.

## Constraints

Specify constraints on the software to be developed imposed by external standards, regulatory requirements, or project limitations. A constraint on an application is a requirement that limits the available options for developing it. Design or implementation constraints describe limits or conditions on how the application is to be designed or implemented. These (nonfunctional) requirements are NOT intended to replace the design process, they merely specify conditions imposed upon the project by the customer, the environment, or other circumstances.

## Error Handling Requirements.

Error-handling requirements specify how the application responds to different types of errors. Options for error-handling requirements are:

* Ignore
* Warn user
* Allow unlimited retries
* Log and proceed anyway
* Substitute default values
* Shut down

There are two major types of errors that you may consider discussing:

1. the errors that actors (entities interacting with the application such as a useror other system) make. In these group you need to explain how the application must respond to anomalies in the environment For example: what should the application do if it receives a message from another application that is not in an agreed-upon format?
2. the errors that may result by bug or a malfunction of the system (developer based). These errors refer to actions that the application should take if it finds itself having committed an error-that is, because of a defect in its construction. This kind of error requirement is applied very selectively, because the teams aim is to produce defect-free applications in the first place rather than covering their mistakes with a large set of error-handing requirements. For example, when a function is called with improper parameters, we program a continuation of the application only if such an erroneous continuation is preferable to the actual termination of the application

## Other Non-Functional Requirements

If there are any non functional requirements that you could not place on the above categories, please specify these requirements here, with appropriate subheadings.

# Logical Database Requirements

Specify the logical requirements for any information that is to be placed into a database, including:

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.

Provide a detailed Enhanced Entity Relationship (EER) diagram or any other database design diagram to better display the overall database design you envision. If you are going to provide an EER, it needs to be based on Chen’s notation (check the Lecture Notes section on moodle for the notation)

If you are going to use a non-relational database, support your design with an appropriate diagram.

# Verification

Software verification is a discipline of software engineering whose goal is to assure that software fully satisfies all the expected requirements. In this section briefly provide the verification approaches and methods planned to qualify the software. In other words how are you going to make sure that the software has satisfied all functional and non-functional requirements that you have specified on the above sections.

# Discussions

This is the part where you discuss the problems you have experienced while preparing the SRS.

In this document you will discuss the following subheadings within the perspective of SRS and while you were preparing it and the activities you conducted for its preparation. If some of these subheadings are not applicable, just specify why they are not applicable.

## Limitations and Constraints

List any limitations and constraints you experienced while preparing the SRS.

## Health and Safety Issues

List any health and safety issues you experienced while preparing the SRS.

## Legal Issues

List any legal issues you experienced while preparing the SRS.

## Economic Issues and Constraints

List any economic issues and constraints you experienced while preparing the SRS.

## Sustainability

List any sustainability related activities you undertake while preparing the SRS.

## Ethical Issues

List any ethical issues you experienced while preparing the SRS.

## Multidisciplinary Collaboration

List any multidisciplinary collaboration you realized while preparing the SRS.

# References

|  |  |
| --- | --- |
| [1] | Chouseinoglou, Sevgi, Genç, Uçar and Yıldırım, "Initial Plan of Team 24," 2024. |
| [2] | A Guide to the Project Management Body of Knowledge, Project Management Institute, 2021. |
| [3] | J. Pinto, Project Management: Achieving Competitive Advantage, 5 / Global ed., Pearson, 2019. |
| [4] | "ISO/IEC/IEEE 29148:2018 - Systems and software engineering - Life cycle processes - Requirements engineering," 2018. |