****

**COMSATS University Islamabad (CUI)**

Software Requirement Specification  
(SRS DOCUMENT)

for

**<Project Name>**  
Version 1.0

***By***

**Student Name 1 CIIT/SP18-BCS-xxx/ISB**

**Student Name 2 CIIT/SP18-BCS-xxx/ISB**

***Supervisor*Supervisor Name**

***Co-Supervisor (if any)*C0-Supervisor Name**

*Bachelor of Science in Computer Science (20xx-20xx)*

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Application Evaluation History

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Supervised by

<Supervisor’s Name>

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Table of Contents**

Revision History i

Application Evaluation History ii

1. Introduction i

1.1 Purpose i

1.2 Scope i

2. Overall Description i

2.1 Product Perspective i

2.2 Operating Environment i

3. Design and Implementation Constraints i

4. Requirement Identifying Technique ii

4.1 Use Case Diagram ii

4.2 Use Case Description ii

5. Functional Requirements iv

5.1 Functional Requirement X v

6. Non-Functional Requirements v

6.1 Reliability v

6.2 Usability vi

6.3 Performance vi

6.4 Security vi

7. External Interface Requirements vi

7.1 User Interfaces vi

7.2 Software Interfaces vii

7.3 Hardware Interfaces vii

7.4 Communication Interfaces vii

8. Project Gantt Chart viii

9. Conclusion viii

10. Appendices viii

11. References viii

12. Plagiarism Report ix

# 

# Introduction

The introduction presents an overview to understand how the SRS is organized and how to use it.

## Purpose

Identify the product or application whose requirements are specified in this document.

## Scope and Modules

Provide a short description of the software being specified and its purpose. You might provide a high-level summary of the major features the software contains or the significant functions that it performs.

## Modules

Briefly describe the modules

# Overall Description

This section presents the overall description of the targeted project.

## Product Perspective

Describe the product’s context and origin. Is it the next member of a growing product line, the next version of a mature system, a replacement for an existing application, or an entirely new product?

## Operating Environment

Describe the environment in which the software will operate, which might include the hardware platform; operating systems and versions; geographical locations of users, servers, and databases; and organizations that host the related databases, servers, and websites.

Example:

***OE-1:****The System shall operate correctly with the following web browsers: Windows Internet Explorer versions 7, 8, and 9; Firefox versions 12 through 26; Google Chrome (all versions); and Apple Safari versions 4.0 through 8.0.*

## Design and Implementation Constraints

There are times when a certain programming language must be used, a code library that has already had time invested to develop it needs to be used, and so forth. Describe any factors that will restrict the options available to the developers and the rationale for each constraint. Constraints are described further in Chapter 14[[1]](#footnote-1), “Beyond functionality.”

Example:

*CO-1: The system shall use the current corporate standard Oracle database engine*

# Requirement Identifying Technique

This section describes the requirements identifying technique(s) which further help to derive functional requirements specification. The selection of the technique(s) will depend on the type of project. For instance,

* **Use case** is an effective technique for interactive end-user applications
* **Event- response tables** is for real-time system and
* **Storyboarding** for graphically intensive applications.

In addition to the above, the projects involving data warehouses, batch processes, hardware devices with embedded control software, and computationally intensive applications required to follow other suitable techniques. Such techniques are described further in Chapter 12, “A picture is worth 1024 words.” For documenting this section let consider identifying requirements through use case as an example.

## Use Case Diagram

Create a use case diagram using **MS Visio** for your system. For detail guideline to develop use case diagram, follow any of latest **UML book**]

## Use Case Description

The table below indicates a comprehensive use case template filled in with an example drawn from the Cafeteria ordering system (COS). (Appendix C) shows more sample use cases written according to this template. As with all templates, you don’t complete this from top to bottom, and you don’t necessarily need all the template information for every use case. The template is simply a structure in which to store the information you encounter during a use case discussion in an organized and consistent fashion. The template reminds you of all the information you should contemplate regarding each use case. For more detail see Chapter 8, “Understanding user requirements”

Table 1: Textual Description of <use case name>.

|  |  |
| --- | --- |
| **Use Case ID:** | Enter a unique numeric identifier for the Use Case. e.g. UC-1 |
| **Use Case Name:** | Enter a short name for the Use Case using an active verb phrase. e.g.  Order a Meal |
| **Actors:** | [An actor is a person or other entity external to the software system being specified who interacts with the system and performs use cases to accomplish tasks.] e.g.   |  |  |  |  | | --- | --- | --- | --- | | Primary Actor: | Patron | Secondary Actors: | Cafeteria Inventory System | |
| **Description:** | [Provide a brief description of the reason for and outcome of this use case.] e.g.  A Patron accesses the Cafeteria Ordering System from either the corporate intranet or external Internet, views the menu for a specific date, selects food items, and places an order for a meal to be picked up in the cafeteria or delivered to a specified location within a specified 15-minute time window. |
| **Trigger:** | [Identify the event that initiates the use case.]e.g.  A Patron indicates that he wants to order a meal. |
| **Level:** | Enter Use Case Goal Level here: Hight/Medium/Low |
| **Preconditions:** | [List any activities that must take place, or any conditions that must be true, before the use case can be started.  PRE-1. Patron is logged into COS.  PRE-2. Patron is registered for meal payments by payroll deduction. |
| **Includes:** | [List any other use cases that are included (“called”) by this use case. Common functionality that appears in multiple use cases can be split out into a separate use case that is included by the ones that need that common functionality. e.g. steps 1-4 in the normal flow would be required for all types of ATM transactions- a Use Case could be written for these steps and “included” in all ATM Use Cases.] |
| **Normal Flow:** | [Provide a detailed description of the user actions and system responses that will take place during execution of the use case under normal, expected conditions.  1.0 Order a Single Meal  1. A patron asks to view the menu for a specific date. (see 1.0. E1, 1.0.E2)  2. COS displays the menu of available food items and the daily special.  3. Patron selects one or more food items from the menu. (see 1.1)  4. Patron indicates that the meal order is complete. (see 1.2)  5. COS displays ordered menu items, individual prices, and total price, including taxes and delivery charges.  6. Patron either confirms meal order (continue normal flow) or requests to modify meal order (return to step 2).  7. COS displays available delivery times for the delivery date.  8. Patron selects a delivery time and specifies the delivery location.  9. Patron specifies the payment method.  10. COS confirms acceptance of the order.  11. COS sends Patron an email message confirming order details, price, and delivery instructions.  12. COS stores order sends food item information to Cafeteria Inventory System, and updates available delivery times. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | [Document legitimate branches from the main flow to handle special conditions (also known as extensions). For each alternative flow reference the branching step number of the normal flow and the condition that must be true for this extension to be executed. e.g.  1.1 Order multiple identical meals  1. Patron requests a specified number of identical meals. (see 1.1. E1)  2. Return to step 4 of normal flow.  1.2 Order multiple meals  1. A patron asks to order another meal.  2. Return to step 1 of normal flow.  Note: Insert a new row for each distinctive alternative flow. ] |
| **Exceptions:** | 1.0. E1 Requested date is today and the current time is after today’s order cutoff time  1. COS informs Patron that it’s too late to place an order for today.  2a. If Patron cancels the meal ordering process, then COS terminates use case.  2b. Else if Patron requests another date, then COS restarts use case.  1.0. E2 No delivery times left  1. COS informs Patron that no delivery times are available for the meal date.  2a. If Patron cancels the meal ordering process, then COS terminates use case.  2b. Else if Patron requests to pick the order up at the cafeteria, then continue with normal flow, but skip steps 7 and 8.  1.1. E1 Insufficient inventory to fulfill multiple meal order  1. COS informs Patron of the maximum number of identical meals he can order, based on current available inventory.  2a. If Patron modifies a number of meals ordered, then return to step 4 of normal flow.  2b. Else if Patron cancels the meal ordering process, then COS terminates use case. |
| **Postconditions:** | [Describe the state of the system at the conclusion of the use case execution.  POST-1. Meal order is stored in COS with a status of “Accepted.”  POST-2. Inventory of available food items is updated to reflect items in this order.  POST-3. The remaining delivery capacity for the requested time window is updated. |
| **Business Rules** | Use cases and business rules are intertwined. Some business rules constrain which roles can perform all or parts of a use case. Perhaps only users who have certain privilege levels can perform specific alternative flows. That is, the rule might impose preconditions that the system must test before letting the user proceed. Business rules can influence specific steps in the normal flow by defining valid input values or dictating how computations are to be performed e.g.  BR-1 Delivery time windows are 15 minutes, beginning at each quarter-hour.  BR-2 Deliveries must be completed between 11:00 A.M. and 2:00 P.M. local time, inclusive.  Note: If you are maintaining the business rule in a separate table in SRS then only mention here their IDs. |
| **Assumptions:** | [List any assumptions.e.g. Assume that 15 percent of Patrons will order the daily special (Source: previous 6 months of cafeteria data). |

# 

# Functional Requirements

This section describes the functional requirements of the system expressed in the natural language style. This section is typically organized by feature as a system feature name and specific functional requirements associated with this feature. It is just one possible way to arrange them. Other organizational options include arranging functional requirements by use case, process flow, mode of operation, user class, stimulus, and response depend on what kind of technique has been used to understand functional requirements. Hierarchical combinations of these elements are also possible, such as use cases within user classes. For further detail see Chapter 10 “Documenting the requirements”. Let consider the feature scheme as an example.

## Functional Requirement X

Itemize the specific functional requirements associated with each feature. These are the software capabilities that must be implemented for the user to carry out the feature’s services or to perform a use case. Describe how the product should respond to anticipated error conditions and to invalid inputs and actions. Uniquely label each functional requirement, as described earlier. You can create multiple attributes for each functional requirement, such as rationale, source, dependencies, etc. The following template is required to write functional requirements. For further detail see Chapter 11” Writing excellent requirements”.

Table 1: Description of FR-1

|  |  |
| --- | --- |
| **Identifier** | FR-1 |
| **Title** | Title of requirement |
| **Requirement** | Description of requirement which may be written either from the user or system perspective e.g.  If written in a **user perspective**  The [user class or actor name] shall be able to [do something] [to some object] [qualifying conditions, response time, or quality statement].  If written in a **system perspective**  [optional precondition] [optional trigger event] the system shall [expected system response] |
| **Source** | Where this requirement comes from (who originate it) |
| **Rationale** | The motivation behind the requirement |
| **Business Rule (if required)** | Any restriction, policy, the rule that the particular requirement must be fulfilled through its functional behavior |
| **Dependencies** | Requirements ID that is dependent on this requirement |
| **Priority** | High/Medium/Low |

# Non-Functional Requirements

This section specifies nonfunctional requirements other than constraints, which are recorded in section 2.3, and external interface requirements, which will appear in section 7. These quality requirements should be specific, quantitative, and verifiable. Chapter 14 “beyond functionality” presents more information about these quality attribute requirements and many examples. The following are some examples of documenting guidelines.

## Reliability

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Also, don't confuse reliability with availability which is quite a different kind of requirement.  Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

## Usability

Usability requirements deal with ease of learning, ease of use, error avoidance and recovery, the efficiency of interactions, and accessibility. The usability requirements specified here will help the user interface designer create the optimum user experience.

Example:

*USE-1: The COS shall allow a user to retrieve the previous meal ordered with a single interaction.*

## 

## Performance

State specific performance requirements for various system operations. If different functional requirements or features have different performance requirements, it’s appropriate to specify those performance goals right with the corresponding functional requirements, rather than collecting them in this section.

Example:

*PER-1: 95% of webpages generated by the COS shall download completely within 4 seconds from the time the user requests the page over a 20 Mbps or faster Internet connection.*

## Security

One or more requirements about protection of your system and its data. The measurement can be expressed in a variety of ways (effort, skill level, time, ...) to break into the system.  Do not discuss solutions (e.g. passwords) in a requirements document.

# External Interface Requirements

This section provides information to ensure that the system will communicate properly with external components. If different portions of the product have different external interfaces, incorporate an instance of this section within the detailed requirements for each such portion.  
  
Reaching agreement on external and internal system interfaces has been identified as a software industry best practice. Place detailed descriptions of the data and control components of the interfaces in the data dictionary. A complex system with multiple subcomponents should use a separate interface specification or system architecture specification. The interface documentation could incorporate material from other documents by reference. For instance, it could point to a separate application programming interface (API) specification or to a hardware device manual that lists the error codes that the device could send to the software.

## User Interfaces

Describe the logical characteristics of each user interface that the system needs. Some possible items to include are

\* References to GUI standards or product family style guides that are to be followed.

\* Standards for fonts, icons, button labels, images, color schemes, field tabbing sequences, commonly used controls, and the like.

\* Screen layout or resolution constraints.

\* Standard buttons, functions, or navigation links that will appear on every screen, such as a help button.

\* Shortcut keys.

\* Message display conventions.

\* Layout standards to facilitate software localization.

\* Accommodations for visually impaired users.

Document the user interface design details, such as specific dialog box layouts, in a separate user interface specification, not in the SRS. Including screen mock-ups in the SRS to communicate another view of the requirements is helpful, but make it clear that the mock-ups are not the committed screen designs. If the SRS is specifying an enhancement to an existing system, it sometimes makes sense to include screen displays exactly as they are to be implemented. The developers are already constrained by the current reality of the existing system, so it's possible to know up front just what the modified, and perhaps the new, screens should look like.

## Software Interfaces

Describe the connections between this product and other software components (identified by name and version), including databases, operating systems, tools, libraries, and integrated commercial components. State the purpose of the messages, data, and control items exchanged between the software components. Describe the services needed by external software components and the nature of the intercomponent communications. Identify data that will be shared across software components. If the data-sharing mechanism must be implemented in a specific way, such as a global data area, specify this as a constraint.

## Hardware Interfaces

Describe the characteristics of each interface between the software and hardware components of the system. This description might include the supported device types, the data and control interactions between the software and the hardware, and the communication protocols to be used.

## Communication Interfaces

State the requirements for any communication functions the product will use, including e-mail, Web browser, network communications protocols, and electronic forms. Define any pertinent message formatting. Specify communication security or encryption issues, data transfer rates, and synchronization mechanisms.

# Project Gantt Chart

Create the Gantt Chart and provide estimated start and end dates of all proposed modules/tasks for each team member. Also, identify the dependencies (which tasks cannot be started/completed, until the dependent task is completed). Gantt chart can be created using MS Project. Semester calendar can be useful to develop the Gantt Chart.



Figure 1: Gant Chart of the Targeted Project.

# Conclusion

Conclude this document.

(Usually 4-5 sentences)

# Appendices

Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.

# References

List any documents or other resources to which this SRS refers, if any. These might include user interface style guides, standards, system requirements specifications, interface specifications, or the SRS for a related product. The following are a few examples of different resources i.e. journal article, book, and website.

* 1. Lyda M.S. Lau, Jayne Curson, Richard Drew, Peter Dew and Christine Leigh, (1999), Use Of VSP Resource Rooms to Support Group Work in a Learning Environment, ACM 99, pp-2. **(Journal paper example)**
  2. Hideyuki Nakanishi, Chikara Yoshida, Toshikazu Nishmora and TuruIshada, (1996), FreeWalk: Supporting Casual Meetings in a Network, pp 308-314 **(paper on web)** http://www.acm.org/pubs/articles/proceedings/cscw/240080/p308-nakanishi.pdf
  3. Ali Behforooz& Frederick J.Hudson, (1996), Software Engineering Fundamentals, Oxford University Press. Chapter 8, pp255-235. **(book reference example)**
  4. Page Author, Page Title, http://www.bt.com/bttj/archive.htm, Last date accessed**. (web site)**

# Plagiarism Report

Attach the Plagiarism report of the targeted project scope document from the library staff of Turnitin tool (<http://turnitin.com>).

1. Karl Wiegers and Joy Beatty, Software Requirements 3rd edition.

   Note: All the referenced chapters are selected from the above book [↑](#footnote-ref-1)