**COMSATS University Islamabad,**

**Abbottabad Campus**

**SOFTWARE REQUIREMENTS SPECIFICATION   
(SRS DOCUMENT)**

**for**

**<PROJECT NAME>**  
Version 1.0

***By***

**Student Name 1 CIIT/SP09-BCS-xxx/ATD**

**Student Name 2 CIIT/SP09-BCS-xxx/ATD**

***Supervisor*Supervisor Name**

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

**Table of Contents**

**Revision History 3**

**1.** **Introduction 5**

1.1 Purpose 5

1.2 Scope 5

**2.** **Overall description 5**

2.1 Product perspective 5

2.2 Operating environment 5

2.3 Design and implementation constraints 5

**3.** **Requirement identifying technique 6**

3.1 Use Case Diagram 6

3.2 Use Case Description 6

**4.** **Specific Requirements 8**

4.1 System feature X 8

**5.** **Quality attributes** Error! Bookmark not defined.

5.1 Usability 9

5.2 Performance 9

**6.** **External interface requirements 35**

6.1 User interfaces 35

6.2 Software interfaces 35

6.3 Hardware interfaces 35

6.4 Communications interfaces 35

**7.** **Project Gantt chart 35**

**8.** **References 10**

**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\_\_\_\_\_\_\_\_\_\_\_

# 1. Introduction

The **Software Requirements Specification (SRS)** document provides an organized framework for understanding the requirements and design of our liver tumor segmentation system. This system automates tumor detection in 3D CT scans using advanced nnFormer architecture. The project leverages deep learning techniques, specifically a hybrid approach of convolutional layers and attention mechanisms, to achieve accurate and efficient segmentation.

This document aims to facilitate communication among stakeholders, including developers, researchers, and potential users, ensuring that the system meets the specified requirements.

## 1.1 Purpose

The purpose of this project is to develop a liver tumor segmentation system that significantly improves diagnostic processes by automating tumor detection. By using the nnFormer architecture, the system captures both local and global features, offering a robust solution for liver tumor analysis. It is intended to aid healthcare professionals by providing precise segmentation and statistical analysis of tumours.

## 1.2 Scope

The system will process 3D CT scans by converting them into 2D patches to manage computational complexity. The architecture includes two primary models: one for liver segmentation and another for tumor segmentation. The frontend will allow doctors to upload images, view segmented outputs, and analyze tumor-related statistics.

**Overall description**

**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
* **Story boarding** for graphically intensive applications.

In addition to 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 indicate 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 Show the detail use case template**

|  |  |
| --- | --- |
| **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. |
| **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. |
| **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. Remaining delivery capacity for the requested time window is updated. |
| **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. Patron asks to view menu for a specific date. (see 1.0. E1, 1.0.E2)  2. COS displays menu of available food items and the daily special.  3. Patron selects one or more food items from menu. (see 1.1)  4. Patron indicates that meal order is complete. (see 1.2)  5. COS displays ordered menu items, individual prices, and total price, including taxes and delivery charge.  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 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 which 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. 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 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 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. |
| **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 on 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.   1. 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 natural language style. This section is typically organized by feature as 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 what kind of technique which 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 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 2 Show the functional requirement template**

|  |  |
| --- | --- |
| **Identifier** | Requirement ID |
| **Title** | Title of requirement |
| **Requirement** | Description of requirement which may be written either from user or system perspective e.g.  If written in **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 **system perspective**  [optional precondition] [optional trigger event] the system shall [expected system response] |
| **Source** | Where this requirement is come from (who originate it) |
| **Rationale** | Motivation behind the requirement |
| **Business Rule (if required)** | Any restriction, policy, rule that the particular requirement must be fulfilled through its functional behavior |
| **Dependencies** | Requirements ID that are 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. Following are some example for documenting guideline.

**Usability**

Usability requirements deal with ease of learning, ease of use, error avoidance and recovery, 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.*

**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.

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

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