**COMSATS University Islamabad,**

**Abbottabad Campus**

**SOFTWARE REQUIREMENTS SPECIFICATION   
(SRS DOCUMENT)**

**for**

**AI-Assisted Learning Management System**  
Version 0.1

***By***

**Hanzla Nouman CIIT/FA21-BSE-015/ATD**

**Laiba Binta Tahir CIIT/FA21-BSE-019/ATD**

**Arfah Ali CIIT/FA21-BSE-080/ATD**

***Supervisor*Mr. Mukhtiar Zamin**

***Bachelor of Science in Computer Science (2021-2025)***

**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**

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\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Introduction**

This document presents the Software Requirements Specification (SRS) for the AI-Assisted Learning Management System (LMS). It provides an organized and structured overview of the system, its purpose, scope, and features. The document serves as a foundation for all stakeholders, including students, teachers, administrators, and developers, to understand the system requirements and design goals.

The SRS will guide the development, implementation, and validation processes to ensure the system meets user needs and expectations.

**Purpose**

The purpose of this document is to outline the functional and non-functional requirements for the AI-Assisted LMS. This software will be developed to assist educational institutions in managing teaching, learning, and evaluation processes efficiently. The primary users of this system include students, teachers, and administrators.

By leveraging AI tools, the system aims to enhance content delivery, automate administrative tasks, personalize learning experiences, and streamline evaluations. This document ensures that the requirements are clear, measurable, and feasible for implementation.

**Scope**

The AI-Assisted LMS is an advanced learning platform designed to improve the educational process by integrating artificial intelligence into content generation, evaluation, and administrative functions. It supports the following key objectives:

* Personalized Learning: Providing AI-driven content recommendations and interactive materials tailored to individual students’ needs.
* Automated Evaluations: Enabling self-evaluation tools and standard assessment processes with instant feedback.
* Performance Analytics: Generating performance insights for teachers and administrators to make data-driven decisions.
* Attendance Management: Automating attendance tracking through AI-powered recognition tools.
* Content Generation: Assisting teachers with AI-based tools to create high-quality learning materials efficiently.

The software will be accessible via web browsers and mobile devices, ensuring flexibility for users. It connects with external services such as authentication servers and AI API servers for facial recognition and secure user management. The AI-Assisted LMS will serve educational institutions ranging from small schools to large universities, offering an efficient and intelligent solution for managing learning and teaching processes.

**Overall description**

**Product perspective**

The AI-Assisted Learning Management System (LMS) is a product designed to address the evolving needs of educational institutions by integrating artificial intelligence into traditional LMS platforms. Unlike existing systems, this product incorporates advanced AI-driven capabilities such as content generation, personalized learning recommendations, automated evaluations, and performance analytics.

The system is developed as a standalone solution but can also integrate with existing educational tools, databases, and infrastructure. It serves as a replacement for outdated or manual systems and enhances existing LMS solutions by offering innovative AI-powered features.

**Key Context**

**Origin:** The system was conceptualized to modernize educational processes, providing a scalable, intelligent, and efficient platform that addresses challenges faced by educators and learners.

**New Capabilities:** The AI-Assisted LMS offers functionalities such as:

* AI-powered content generation and recommendations.
* Automated evaluations and real-time feedback.
* AI-driven attendance management through facial recognition.
* Comprehensive performance analytics for decision-making.

**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 outlines the quality requirements that ensure the system meets performance, usability, and reliability standards.

**Usability**

*USABILITY-1: The system shall allow users to access the Content Recommendation module and receive recommendations within 2 clicks of the main dashboard.*

*USABILITY-2: The Standard Evaluation module shall provide clear, step-by-step instructions for completing evaluations to minimize user confusion.*

*USABILITY-3: The system interface shall support accessibility features compliant with WCAG 2.1 Level AA standards to accommodate users with disabilities.*

*USABILITY-4: The mobile version of the LMS shall provide consistent and intuitive user experience compared to the web version.*

*USABILITY-5: Self-Evaluation and Attendance modules shall include tooltips to guide users on their functionality.*

**Performance**

*PERFORMANCE-1: 95% of webpages generated by the LMS shall load within 3 seconds over a 20 Mbps or faster internet connection.*

*PERFORMANCE-2: The Content Generation module shall process and generate learning content within 10 seconds for standard text inputs.*

*PERFORMANCE-3: The AI Face Recognition Service (AI API Server) for attendance tracking shall process and verify a student’s face within 2 seconds of submission.*

*PERFORMANCE-4: The system shall support a minimum of 500 concurrent users accessing modules like Analytics and Content Recommendation without performance degradation.*

*PERFORMANCE-5: Database queries in the Analytics module shall return results within 5 seconds for a dataset of up to 100,000 records.*

**Reliability**

*RELIABILITY-1: The system uptime shall be 99.9% over a 24/7 operational period.*

*RELIABILITY-2: The Database Server shall recover automatically from failures within 30 seconds.*

*RELIABILITY-3: The system shall support data backup every 12 hours to prevent data loss.*

**Security**

*SECURITY-1: The system require multi-factor authentication (MFA) for users accessing sensitive modules like Analytics and Attendance.*

*SECURITY-2: Data transferred between Client Devices and servers shall be encrypted using HTTPS/TLS 1.3 protocols.*

*SECURITY-3: All user data stored in the Database Server shall be encrypted using AES-256 encryption standards.*

*SECURITY-4: The system shall log all user activities, including logins, access, and modifications, for auditing purposes.*

**Scalability**

*SCALABILITY-1: The system shall be able to scale horizontally to support up to 10,000 concurrent users by adding additional Application Servers.*

*SCALABILITY-2: The AI Face Recognition Service shall be optimized to scale up to process 5,000 attendance records per hour.*

*SCALABILITY-3: The system shall ensure seamless performance even as the database size grows up to 1 million records.*

**Maintainability**

*MAINTAINABILITY-1: The system shall allow new AI components to be integrated into the Application Server with minimal changes to existing modules.*

*MAINTAINABILITY-2: Code shall follow industry-standard design patterns and include detailed documentation to ensure maintainability.*

*MAINTAINABILITY-3: System logs and error messages shall be stored to facilitate troubleshooting and future system updates.*

**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)