A Software Subsystem of a Learning Management System Using Java

Software Engineering Team Project – CS 3321

Spring 2020

Team: Magdalena Hernandez, Rick Jenkins, Jose Ramos, Mario Rodriguez

May 2, 2020

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Updated By** | **Update Comments** |
| 0.1 | 2/6/2020 | Project Team | Initial Document Release. |
| 0.2 | 04/17/20 |  | Adding Use Cases |
| 0.3 | 5/3/20 |  | Adding CRC cards |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 

# Introduction

# 

# As access to high speed internet improves and access to internet becomes more accessible, remote learning opportunities are becoming ubiquitous. This has made learning management systems (LMS) software important to institutions that educate or train remotely. LMS are software products that deliver educational courses, training programs, and/ or learning and development programs. LMS key features are designed to allow users to manage the administration, documentation, tracking, reporting, and delivery of courses.

# The objective of this project is to deliver a subsystem of an LMS that will manage the administration of course. This subsystem of an LMS will store and retrieve student’s courses, id numbers, exam scores, and semester grade point average (GPA) and include tracking and reporting features.

# This project includes going through the core phases of the software development cycle from requirement gathering to implementation, resulting in a functional subsystem of an LMS.

* 1. **Project Deliverables**

This is a table of the project deliverables that we will have for this project.

|  |  |
| --- | --- |
| **Deliverables** | **Date** |
| Software Project Management Plan | 5/03/2020 |
| UML Diagrams | 5/03/2020 |
| Executable Software | 5/03/2020 |
| Software Source Code | 5/03/2020 |
| Version Control Documentation | 5/03/2020 |
| Software Test Cases | 5/03/2020 |
| Data Storage Files | 5/03/2020 |

# Project Organization

* 1. Process Model

For this project we decided to use the build-it – and fix it approach since the scope of this project was relatively small. We used this approach for all phases of the software development cycle. This means that we would “build” our project deliverables for a specific phase, and then fix it until it meets our standards.

* 1. Organizational Structure

For our organizational structure we decided to use the democratic team approach. We decided to use this method, since we were all skilled at coding, and because we all had vested interests in this project. The development team consist of Magdalena Hernandez, Rick Jenkins, Jose Ramos, Mario Rodriguez (software engineers).

* 1. Organizational Boundaries and Interfaces

All the work on this project will be divided equally among the development team. We will have meetings on weekly basis to ensure that team is making sufficient progress on the project. Any major changes will have to be approved by the entire team vote and must be documented.

* 1. Project Responsibilities

Each member is responsible for the quality of the module he or she codes. The development team will oversee the class definitions and modules, as well as module integration and overall quality of the product.

# Design Details.

* 1. Design Objective and Priorities

This project will be developed using JAVA. Java is an object-oriented programming language (OOP) that is useful for the building of large applications. MySQL server is used in database creation for the LMS. The database stores the structure of the data is form of tables and holds all information for the LMS.

* 1. Use Cases for Project LMS

This use case diagram depicts the overall design of the LMS with all three users.

A close up of a map

Description automatically generated

* 1. Use Cases for Administrator

The uses cases for the Administrator are displayed below in the various use cases diagrams, these diagrams depict the various options available to the administrator.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

3.3 Use Cases for Admin

A screenshot of text

Description automatically generated

3.3 Use Cases for Admin

A screenshot of text

Description automatically generated

* 1. Faculty Use Cases. The uses cases for the Faculty are displayed below in the various use cases diagrams, these diagrams depict the various options available to the administrator.

A close up of text on a white background

Description automatically generated

* 1. Student Use Cases. The uses cases for the Student are displayed below in the various use cases diagrams, these diagrams depict the various options available to the Student.

A screenshot of text

Description automatically generated

* 1. User Use Cases: The following use case depicts the login, and manage my account modules for users.

A close up of text on a white background

Description automatically generated

**4.1 UML Diagram for LMS**. The UML below represents the tables implemented in MySQL for the LMS database.

A close up of text on a white background

Description automatically generated

* 1. **Sequence Diagrams:** The following sequence diagram depicts the login process.

A screenshot of a social media post

Description automatically generated

5.2 Sequence Diagrams. The following sequence diagram depicts the student accessing course details.

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated5.3. Sequence Diagrams. The following sequence diagram depicts the student registering for a course.

**6. Class Responsibility Cards (CRC).**The following are the CRC cards that we identified as part of the requirements elicitation phase.

|  |
| --- |
| CLASS  **Admin Class** |
| RESPONSIBILITY   1. Send message to **Classes Class** to add student to course 2. Send message to **Student Class** to add student record 3. Send message to **Student Class** to edit student record 4. Send message to **Student Class** to delete student record 5. Send message to **Faculty Class** to add faculty record 6. Send message to **Faculty Class** to edit faculty record 7. Send message to **Faculty Class** to delete faculty record 8. Send message to **Classes Class** to add a course record 9. Send message to **Classes Class** to edit a course record 10. Send message to **Classes Class** to delete a course record |
| COLLABORATION   1. **Classes Class** 2. **Student Class** 3. **Faculty Class** |

|  |
| --- |
| CLASS  **Student Class** |
| RESPONSIBILITY   1. Send message to **Classes Class** to register to course 2. Send message to **Classes Class** to drop course 3. Send message to **Classes Class** to display student record 4. Add student record 5. Edit student record 6. Delete student record |
| COLLABORATION   1. **Classes Class** |

|  |
| --- |
| CLASS  **Faculty Class** |
| RESPONSIBILITY   1. Send message to **Classes Class** to display assigned courses 2. Send message to **Classes Class** to display students in courses 3. Send message to **Classes Class** to display student record 4. Send message to **Classes Class** to modify student grade 5. Add student record 6. Edit student record 7. Delete student record |
| COLLABORATION   1. **Classes Class** |

**7. Conclusion**

In all developing an LMS, helped us identify the various requirements for each software development phase. We also learned about the various ways to model information such as classes and use cases. These requirements including developing a back-end database, that collected all the required information.

**References**

Schach, S. R. (2002). Classical and object-oriented software engineering: with Uml and C . Dubuque, IA: McGraw-Hill.