Learning Management System

**Software Project Management Plan**

Faisal Charolia

Jeffrey Liu

Shahsalah Uddin

Mason Short

Haris Siddiqui

1. **Introduction**
   1. **Project Overview**

This project is a software subsystem of a learning management system (LMS). An LMS typically deals with the various aspects of a student enrolled in an academic course. Blackboard is an example of a well-known LMS used by students at many Universities. At the most basic level an LMS includes the ability to view the student’s profile, courses, and grades from the perspective of the user or student and to modify the student’s profile, courses, and grades from the perspective of the administrator or professor.

The major objective of this project is to build a basic student information management system per the client’s requirements. This includes storing and retrieving students’ partial information in the current semester and other basic information such as student’s name, student’s ID, registered courses in the current semester, each exam’s score in one course, and GPA calculation in the current semester. The 5 core workflows of the Unified Process were utilized iteratively and incrementally to build the LMS.

**1.2 Project Deliverables**

Several deliverables need to be produced and submitted, particularly those that satisfy the 5 core workflows of the Unified Process. This includes but is not limited to a use case diagram, a class diagram, the Software Project Management Plan (SPMP), the source code and files related to the project, an entity relationship (ER) diagram, and a link to the GitHub account used for version control. All of the necessary artifacts will be made available in a compressed file.

**1.3 Evolution of the SPMP**

This document is subject to change. The assumptions, dependencies, and constraints for the project, as well as the detailed time and resource planning for each phase, can change during the life time of the project as needed. Changes in this information will lead to a new SPMP.

**1.4 References**

Schach, Stephen R. Object-Oriented and Classical Software Engineering. 8th ed., McGraw-Hill, 2011.

**1.5 Definitions and Acronyms**

LMS: Learning Management System

OOP: Object-Oriented Programming

RDBMS: Relational Database Management System

SPMP: Software project management plan

UML: Unified Modeling Language

1. **Project Organization**

**2.1 Process Model**

While the spiral life-cycle model would have been an appropriate choice for this project, considering the fact that the project is not large-scale and only a subset of a an LMS, risk analysis and risk resolution would not be as crucial to its development. Therefore, the perfect choice for developing the LMS would be the rapid-prototyping life cycle model. A rapid prototype will be developed and select team members will function as the client and users to interact with the software. Once the “client” is content with the prototype, a software requirements specification (SRS) document can be written.

**2.2 Organizational Structure**

The Democratic team approach was taken in development of the project as it allows for team members to locate bugs in other team members’ code without compromise of professionalism. However, throughout the different phases of the project, the team began to reorganize itself into a more traditional team approach with delineated specializations, such as a variant of the classical chief programmer team approach.

**2.3 Organizational interfaces**

As mentioned previously, a democratic team approach was taken at the onset of the project and specific tasks were handled in a democratic fashion. For this reason, and as development of the project matured, positions were not clearly defined but could fit the criteria of product manager and project manager. In this fashion, the product manager served as the interface between the project and the client. The project manager served as the interface between the project and the other team members.

**2.4 Project responsibilities**

The below responsibilities were not assigned to any specific member but naturally formed as various tasks required completion:

Product manager:

* responsible for researching and setting the product vision
* relays product vision to client
* creates initial requirements
* creates and maintains product roadmap

Project manager:

* breaks down requirements into tasks
* creates project timelines
* monitors progress of tasks
* relays progress to client

Designer & Developer:

* Designer decomposes the product into modules to be used by the developer
* Code artifacts are implemented by the programmer into code

Requirements Engineer:

* Draws up UML diagrams and related artifacts along with use cases for the client

Software Quality Assurance/Testing:

* Perform unit testing, conducting code reviews, and perform product testing

1. **Managerial Process**

**3.1 Management Objectives and Priorities**

As this is a democratic team, each team member is responsible for making sure that the project schedule is on track and that deliverables and related artifacts are being completed as needed. Team members are responsible for sharing all tasks equally, to the extent of their individual skill levels. Since the submission date is fixed and cost is not a factor, functionality or scope will be flexible to accommodate for the schedule.

**3.2 Assumptions, dependencies, and constraints**

Assumptions for this project include that team members have an understanding of object-oriented programming (OOP) languages to develop the LMS. A constraint is the duration of time to complete the project, which is fixed by the submission date.

**3.3 Risk Management**

|  |  |  |
| --- | --- | --- |
| No. | Risk | Method of management |
| 1 | Unable to create deliverables and meet the milestones | Create milestones in advance of submission date to make sure that all deliverables are available. |
| 2 | Loss of data or important files | Use GitHub as version control tool. |
| 3 | Team member unavailability | Task reassignment based on artifact and knowledge of replacement team member. |

**3.4 Monitoring and controlling mechanisms**

To prevent loss of or unintended changes of software GitHub will be used for version control. All members will upload code and other deliverables onto GitHub.

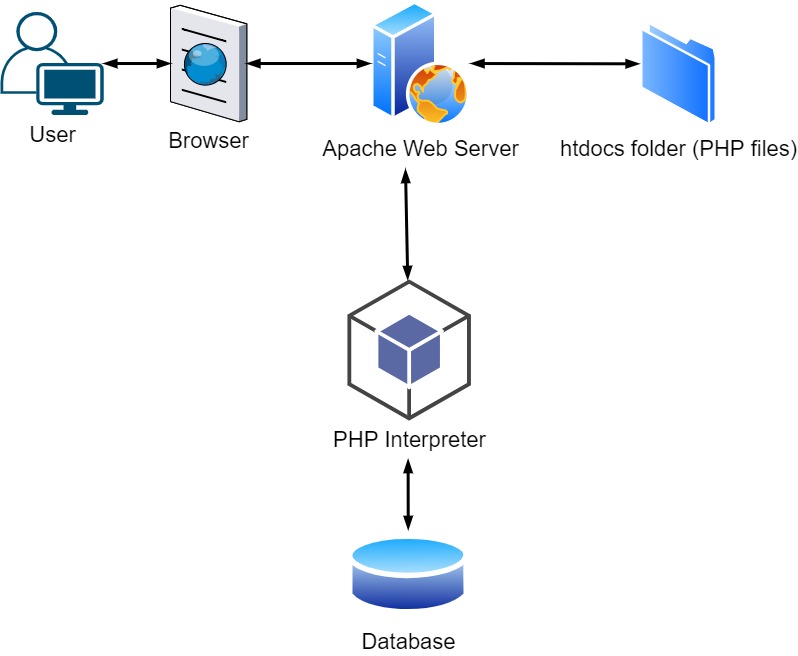
1. **Technical Process**

**4.1 Methods, tools, and techniques**

XAMPP, a software stack consisting of Apache web server, OpenSSL for SSL support, MariaDB (formerly known as MySQL), PHP programming language, and other components, is the platform utilized to develop the LMS. PHP is the primary programming language and MySQL is used as the relational database management system (RDBMS). phpMyAdmin is the front-end, GUI to be used to administer and manage the database.

UML diagrams were created using the website draw.io.

The below architecture diagram displays how the different software components within the LMS connect and exchange data:



**4.2 Software documentation**

Software documentation can be viewed on the GitHub LMS repository.