Software Project Management Plan:

For A Software Subsystem Of Learning Management System

By:

Hilario Lawrence

Long Tong

Puspa Giri

Tooba Hashmi

William Smithson

November 26th 2019

Table of Contents:

pg 1

pg 1

pg 2

pg 2

pg 2

pg 2

pg 3

pg 3

pg 3

pg 3

pg 3

pg 3

pg 4

pg 4

pg 4

pg 4

pg 5

pg 7

pg 7

pg 8

pg 9

1. Introduction

1.1 Project Overview

1.2 Project Deliverables

2. Project Organization

2.1 Process Model

2.2 Organizational Structure

2.4 Project Responsibilities

3. Managerial Process

3.1 Management Objectives and Priorities

3.2 Assumptions, Dependencies, and Constraints

3.5 Staffing Approach.

4. Technical Process

4.1 Methods, Tools, and Techniques

4.2 Software Documentation

4.2.1 Software Requirements Specification (SRS)

4.2.2 Software Design Description (SDD)

4.2.3 Software Test Plan

4.3 User Documentation

5. Schedule

5.1 Detailed Schedule

1. Introduction

This document serves as the project plan for the Learning Management System (LMS) software development effort. The goal of this LMS to help a university IT department improve their activities and improve their services, and for the management to track student’s basic information. This project was done by a team of 5 undergraduate students during an academic semester.

* 1. Project Overview

The Learning Management System is a software program that helps university admins and faculty to create, manage, customize, and conglomerate student information on a large scale.

This software product consists of two separate parts: a database component and a graphic user interface (GUI). The database component performs the core functionality, such as creating, managing, and delivering courses, authenticating users, and serving data, etc. The program runs through a GUI to allow user friendly-ness, which is used by administrators, professors, and students.

The system allows admins to create content (student information, class section details, etc.), organize it into courses, enroll students to said courses, and finally, monitor and assess the students’ performances (assignments, grades, etc.). Functions such as configuring the LMS, such as adding, editing, deleting, and monitoring student accounts, enrollment, courses, and staff members are under the jurisdiction of the administrators. Faculty can also access the students’ progress. Students can view the classes they’re enrolled in, their grades, and their final GPA.

These are the requirements for the LMS application:

* The software product should provide a GUI into a DBMS for the users. The database should provide the standard features expected of a DBMS, including data transfer, storage, modifications, management of entity relationships, and access control. The users will connect to the DBMS via a GUI, as appropriate to the project.
* The software product should restrict access of manipulation and customization to levels of authorized users aka the admins.
* The software product should provide the ability to enter and update users (student, admin, and professor) information, including names, login credentials, access levels, etc.
* The software product should provide the ability to enter, update, and submit assignments and grades

As a team, we agreed to have touch base after every class and work on independent aspects of the code. Because of the GitHub repository, it was a combined team effort. And we often had group meetings to discuss further steps. Often, having meetings in twos or threes, if all five members were not available.

* 1. Project Deliverables

The team was to present a small presentation in the third week of September to let the client know the approach being taken. After which, a final presentation in the last week of November was due. The complete project was due the first of December. This submission included lists of the artifacts used and well as source code, UML diagrams, SPMP, version control documentation, test cases, data storage files, etc.

2. Project Organization

For the lifecycle model, the team originally decided to go with the rapid prototype model and ultimately became a hybrid with the code & fix method.

2.1 Process Model

Despite our client’s firm requirements, unlikely to change, we chose the rapid prototype lifecycle model to follow. This lifecycle model allows flexibility and revisitation to the requirements. If the client did change the requirements, this lifecycle model would allow us to incorporate those changes. Furthermore, the rapid prototype model would allow us to develop a prototype and see if it’s potential could be the final project.

In this case, after building a prototype of the software in the language of C++, it was determined by the team that it was not feasible to continue. With minimal database experience, the team determined it was more appropriate to use SQ lite. Along with this change, the team also transitioned to use python as the foundation of the program. This allowed them to maintain control and make maintenance easier.

Following this transition, the team devolved into the code & fix life cycle model. It is the most time consuming and in retrospect, quite difficult to follow.

2.2 Organizational Structure

The internal management of the project was structured around the foundation of the democratic team model. After every meeting, progress was measured and further directions were taken and given.

2.4 Project Responsibilities

The project was divided into the following milestones:

|  |  |  |
| --- | --- | --- |
| Task | Project Manager | Date of Completion |
| UML and Diagrams | Puspa and Tooba | Sept 25th |
| Database compilation | William | October 15th |
| Functions to associate with database | William | October 25th |
| Population of database | Tooba | November 15th |
| GUI structure built | Hilario | October 25th |
| GUI structure finalized | Puspa | November 15th |
| SPMP report prepared | Long | November 23rd |
| Test cases compiled | Long | November 23rd |

3. Managerial Process

This section of the SPMP specifies the management process for this project.

3.1 Management Objectives and Priorities

The goal for this project was understanding how the software management process works. More than developing this learning management system, the goal was to work together in a team, with a lifecycle and team model in place, to understand the intricacies of how a piece of software is compiled, both back and front end, how one must commit to a timeline and unexpected hurdles, etc.

3.2 Assumptions, Dependencies, and Constraints

The completion of the project was based on certain assumptions such as, the completion deadline was December 1st 2019. The project would be rated according to a criteria, pre-declared.

It was dependent on the ability of the programmers to work together as a group and to be able to handle any unexpected complications.

Fortunately, no complication or constraints were associated with the project.

3.5 Staffing Approach.

The following are skills suggested to complete this project:

* Knowing a basic programming language
* minor software engineering background
* knowledge of how to manipulate a GUI and Database
* Think analytically
* Know how to build use cases, class diagrams, etc

4. Technical Process

4.1 Methods, Tools, and Techniques

* Computing system: Windows
* Development method: standard policies
* Team Structure: Democratic
  + LMS was developed by using a democratic team model. The concept of a democratic team model is surrounded by the idea of egoless programming. This is the basic concept because it allows the team to develop a group identity. They now collectively act and one; rather than individuals and their egos vying for attention and appreciation. The ultimate product belongs to all the group members. Decisions are made in a democratic fashion and finding faults is a group effort.
* Programming language: C++, Python
* Tools: Python IDEs, DB browser, SQ lite 3, PyGUi, Visual studio

4.2 Software Documentation

Specify the work products to be built for this project and the types of peer reviews to be held for

those products. It may be useful to include a table that is adapted from the organization's standard

collection of work products and reviews. Identify any relevant style guide, naming conventions

and documentation formats. In either this documentation plan or the project schedule provide a

summary of the schedule and resource requirements for the documentation effort.

To ensure that the implementation of the software satisfies the requirements, the following

documentation is required as a minimum:

4.2.1 Software Requirements Specification (SRS)

These are the requirements for the LMS application:

* The software product should provide a GUI into a DBMS for the users. The database should provide the standard features expected of a DBMS, including data transfer, storage, modifications, management of entity relationships, and access control. The users will connect to the DBMS via a GUI, as appropriate to the project.
* The software product should restrict access of manipulation and customization to levels of authorized users aka the admins.
* The software product should provide the ability to enter and update users (student, admin, and professor) information, including names, login credentials, access levels, etc.
* The software product should provide the ability to enter, update, and submit assignments and grades

4.2.2 Software Design Description (SDD)

LMS will perform different functions depending on the type of user: student or staff. The staff is comprised of faculty who can function as admins. The administrators will have the control of the system, thus they will have administrative rights.

The features available to Admin:

* Add/Remove Student
* Manage Existing Students
* Manage Existing Logins
* Enroll Students Into Class
* Manage Existing Enrollments
* Add Student Assignment
* Manage Existing Assignments
* Add/Remove Grade
* Manage Existing Grades

The features available to Student:

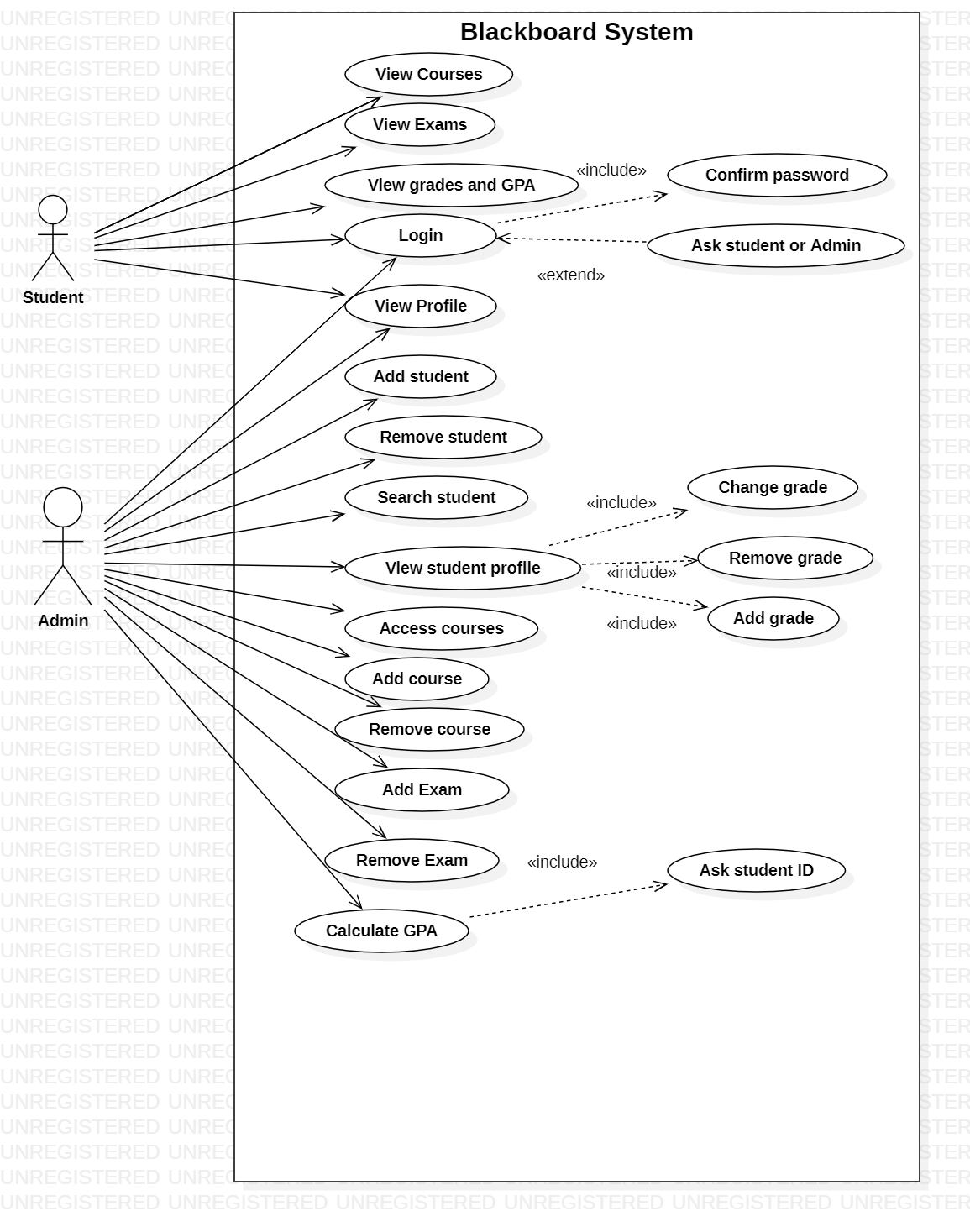
* View Class/Course list
* View GPA

All users can log in and log out. After each user logs in, the next page displays their list of available features as buttons

These available features are visually represented in the following UML Use Case diagram shown in the following photo

.

.



4.2.3 Software Test Plan

4.2.3 Software Test Plan

During this phase, the programmers coded and tested each code artifact separately, linked together all the code artifacts, and tested the product as a whole. Every programmer is personally responsible for making sure that his/her work is correct before committing it to GitHub. The available features outlined in the Analysis – User Characteristics section are used to create test cases:

* Add a student
* Add a staff
* Modify a student
* Modify a staff
* Delete a student
* Delete a staff
* Compute GPA

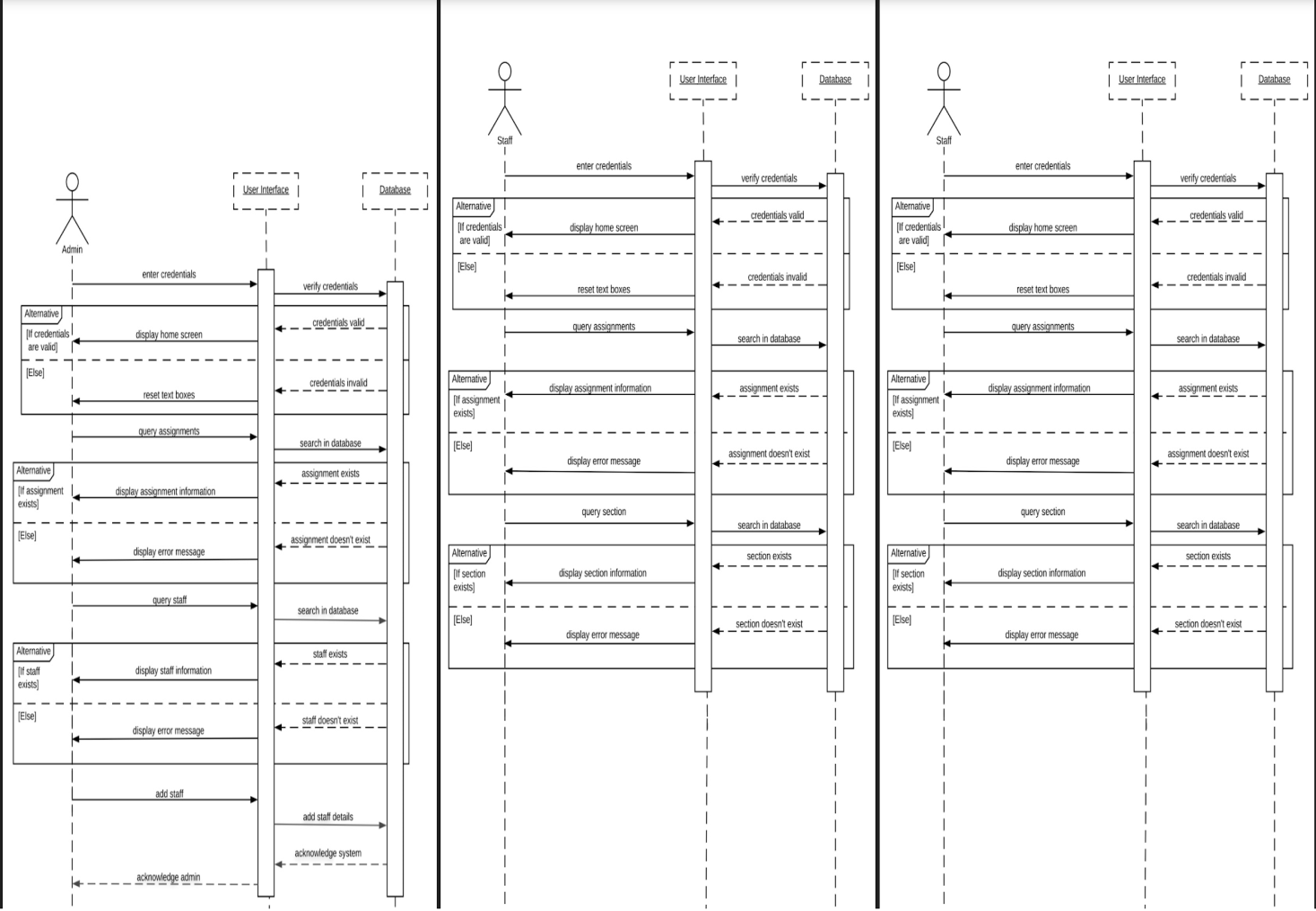
In addition to these direct tests, it is necessary to perform the following additional tests:

* Attempt to add a student that is already on file.
* Attempt to add a staff that is already on file.
* Attempt to delete a student that is not on file.
* Attempt to delete a staff that is not on file.
* Attempt to modify a student that is not on file.
* Attempt to modify an assignment that is not on file.
* Attempt to update each field of a staff twice and check that the second version is stored.

We ran through each test case, and every time we found something undesirable or something didn’t output correctly. For example, if the labels looked like name\_id, then we would return to the code and change it to Name ID. We reiterated these test cases until the software product met the project expectations.

4.3 User Documentation

In our first meeting, we listed out the software requirements and drew up how the Home page/Login screens is supposed to look like.



The above sequence diagrams for each user are visual summaries of their individual use cases. These diagrams show the sequences of the login use case for all users. The sequence is as follows: User enters credentials in the user interface; the DB verifies the credentials from the list in the database. The database sends back a message whether the credentials are valid or not. If they are, then the GUI displays the next view per user. If not, then the GUI displays an error message and resets the textboxes. There are similar sequences for query assignments for student and faculty.

5. Schedule

As a team, we agreed to have touch base after every class and work on independent aspects of the code. Because of the GitHub it was still combined team effort. And we often had group meetings to discuss further steps. Often, having meetings in twos or threes. We all committed to be present in these meetings, whether it was in person or chat via GroupMe.

5.1 Detailed Schedule

* Formation of Group
* Analysis of requirements
* Discussion of approach
  + Establish team model
  + Establish lifecycle model
  + Choose language
  + Create Github repository
* Establish timeline
* Create Class Diagrams, UML diagrams, Use Case diagrams
* Start work on prototype
* Presentation
* Continue working on project
* Issue with file and program
* Discussion to change language and new approach
* Approved
* Begin second prototype in Python
* Establish database
* Establish and connect functions
* Begin construction on GUI
* Populate the database
* Finish the GUI
* Semi-Final Team meeting
* Prepare presentation
* Final presentation
* Final team meeting
* Submission of project