**Software Project Management Plan**

Project Title: **LMS UHD 2020**

Contributors: **Jeffrey Tijerina**

Date: **May 2020**

Version Release: 1.0

**Table of Contents**

**1.0 Introduction/Overview**

This project was primarily created to be integrated and used as a platform in all grades and levels of learning environments from kindergarten through university. This application’s primary function is in the management of student accounts including personal records, and gradebooks through each grade level as well as to have the flexibility to be repurposed for use in other industries.

**1.1 Project Overview**

This application aims to provide a centralized location for the management of all students, staff, and administrators to view and manage commonly used administrative functions for each respective role. Comparable third party alternatives such as Blackboard and Moodle can be purchased and licensed but this project’s goal is to remain completely free and open-source through its entire life cycle.

**1.2 Project Deliverables**

Project deliverables to the client include a standalone application that can create and manage accounts, store student, staff, and administrator data, create, modify, and manage existing data, and provide commonly accessed administrative functions for each role respectively. Data security has been considered and will be further improved and refined in later version revisions.

**1.3 Evolution of SPMP**

The project will be developed through an evolutionary life cycle model and will use a github repository to update, create revisions, and centrally store all relevant files and folders associated with the project’s purpose.

**Project Organisation**

**2.1 Process Model**

The evolution life cycle model was chosen as the primary development model because it gave the most flexibility and ease of development compared to other traditional software development models. The ability to create the initial, concept software and then modify and improve it as time went on was an integral part of this project’s creation.

**2.2 Organizational Structure**

Having been limited by environmental factors and circumstances, the chief programmer style was used to get this project done on time and within the requirements of the lms platform.

**2.3 Project Responsibilities**

Jeffrey Tijerina - Chief Programmer - Led project, designed, and programmed all functionalities within the application. Updated, fixed, tested, and ran all versions that built off of core code.

**Managerial Process**

**3.1 Management Objectives and Priorities**

The primary objective was to create a standalone application with a usable, feature rich, and user friendly GUI that met all client requirements and specifications of a learning management system. Delivering an application on schedule and within the allotted time was essential to this project with added features to be added as time permitted.

|  |  |  |  |
| --- | --- | --- | --- |
| **PROJECT** | **FIXED** | **LIMITED** | **FLEXIBLE** |
| Cost |  | **X** |  |
| Schedule | **X** |  |  |
| Features/Functionality |  |  | **X** |

**3.2. Assumptions, Dependencies, and Constraints**

This project was completely dependent on the continued contribution of the chief programmer. COVID-19 hindered this project from continuing as a group effort mid way through the semester but it was able to be completed independently just the same. Having to switch from a group contribution/effort to individual group work constrained some of the features planned for this lms but core functionality is present and working.

**3.3 Risk Management**

Risk management was handled by keeping all files, code, updates, changes, and associated plans in an online backup service as well as an online repository. By keeping all files locally for the most recent changes and pushing regular backups to two online locations I minimized the risk of losing any important changes or updates performed. As updates were added and bugs fixed they were tested to ensure compatibility throughout the entire program. If something completely broke a feature or core functionality it was rolled back and re-designed to implement this feature in a better way.

**3.4 Monitoring and Controlling Mechanisms**

Monitoring of code was done locally as files and code was changed and tracked by pushing all updated files online through the repository. Version control was used to ensure any features or updates that affected functionality could be rolled back and corrected before testing and delivering to the client.

**3.5 Staffing Approach**

Group members were initially chosen in class but as COVID-19 affected the semester this project was taken on and finished in an individual effort.

**Technical Process**

**4.1 Methods, Tools, and Techniques**

This project was built and compiled in NetBeans. Oracle’s Java API was extensively used as well as resource pages such as stackoverflow, stackexchange, superuser, vscode, and geeksforgeeks. Google Drive was used to backup all files and associated changes as well as github for backup and version control of all changes made. Core code and features were tested as they were created to ensure compatibility throughout the entire program.

**4.2 Software Documentation**

All documentation is organized and displayed in the SPMP and UML diagrams of this project.

**4.2.1 Software Requirements Specification (SRS)**

The requirements of the project have been defined as needing to include student name, student ID, registered courses in the current semester, each exam’s score in each course, GPA calculation in the current semester, admin privileges to add a student and modify data as needed, and a simple GUI to make the system user friendly.

**4.2.2 Software Design Description (SDD)**

The SDD describes the major components of the software design including databases and internal interfaces.

**4.2.3 Software Test Plan**

Each core function or feature implementation will be tested independently before being integrated into the main project. Once independent testing passes and works as desired it will be integrated and added into the core project. If testing within the core project fails, the new implementation should be rolled back, assessed, redesigned if necessary, and re-coded to find and fix all bugs within that new functionality.

**4.3 User Documentation**

I will be using github for all documentation and repositories. Comments will be included throughout code for future developers to be able to jump in and understand the core programming easier. Troubleshooting and version control can also be coordinated and kept track of through github.

**4.4 Project Support Functions**

Provide either directly or by reference, plans for supporting functions for the software project. These functions may include, but are not limited to, configuration management, software quality assurance, and verification and validation. Plans for project support functions are developed to a level of detail consistent with other sections of the SPMP. In particular, the responsibilities, resource requirements, schedules and budgets for each supporting function must be specified. The nature and type of support functions required will vary from project to project. The absence of a software quality assurance, configuration management, or verification and validation plan, however, must be explicitly justified in project plans that do not include them.

**Work Packages, Schedule, and Budget**

**5.1 Methods, Tools, and Techniques**

Specify the work packages for the activities and tasks that must be completed in order to satisfy the project agreement. Each work package is uniquely identified. A diagram depicting the breakdown of project activities and tasks (a work breakdown structure) may be used to depict hierarchical relationships among work packages.

**5.2 Dependencies**

Specify the ordering relations among work packages to account for interdependencies among them and dependencies on external events.

Techniques such as dependency lists, activity networks, and the critical path method may be used to depict dependencies among work packages.

**5.3 Resource Requirements**

Provide, as a function of time, estimates of the total resources required to complete the project. Numbers and types of personnel, computer time, support software, computer hardware, office and laboratory facilities, travel, and maintenance requirements for the project resources are typical resources that should be specified.

**5.4 Budget and Resource Allocation**

Specify the allocation of budget and resources to the various project functions, activities, and tasks.

**5.5 Schedule**

Provide the schedule for the various project functions, activities, and tasks, considering the precedence relations and the required milestone dates. Schedules may be expressed in absolute calendar time or in increments relative to a key project milestone.

**Additional Components**

**6.1 Index**

An index to the key terms and acronyms used throughout the SPMP is optional, but recommended to improve usability of the SPMP.

**6.2 Appendices**

Appendices may be included, either directly or by reference, to provide supporting details that could detract from the SPMP if included in the body of the SPMP. Suggested appendices include:

Current Top 10 Risk Chart

Current Project Work Breakdown structure

Current Detailed Project Schedule

**LMS UML Diagram**

