**Software Project Management Plan**

**for**

***Air Traffic Controller LCMS***

***<Greg Friedman, Mark Holson, Brennan McPherson, and Howard Wheeler>***

Template derived from IEEE Standard 1058-1998

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Release Date** | **Responsible Party** | **Major Changes** |
| 0.1 | 3/6/2015 | Team 1 | Initial Document Release for Comment |
| 0.2 | 4/2/2015 | Team 1 | Updated Document with further details |
| 0.3 | 5/8/2015 | Team 1 | Updated schedule with further details |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**1. Introduction**

1.1 - Project Overview

In the FAA project, a system has to be put in place to modernize the training of air traffic controllers. The system will comprise of two state of the art training modules. Managers will be able to create “smart” lesson plans that are based upon trainees performance on quizzes and simulators.

1.2 - Project Deliverables

|  |
| --- |
| User Requirements Document |
| Use Cases Document |
| Class Diagrams |
| Sequence Diagrams |
| Risks & Mitigation Document |
| Software Project Management Plan |
| Software Process Selection Document |
| Design |
| Source Code |

1.3 - Evolution of the SPMP

This document is subject to multiple changes and edits as the project continues. When circumstances cause changes in the constraints, requirements, or planning, then these new additions will lead to a SPMP with an updated version number. If these changes cause a change in a milestone, then they will be discussed with the Senior Management prior to incorporating them into the document. Weekly meetings will ensure that all project members are up to date with all changes and how they affect the SPMP. The Management Plan will be stored in our GitHub repository to manage different versions of the document.

1.4 - Definitions and Acronyms

LCMS: Learning Content Management System

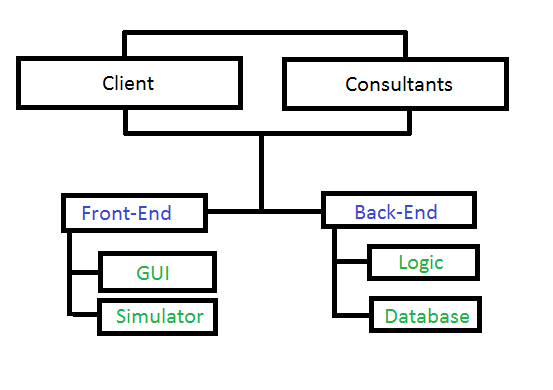
SPMP: Software Project Management Plan

**2. Project Organization**

2.1 - Use-Cases

* User tries to log in, but forgets password and needs to generate a new one
* Training Manager adds a new simulation to the program and views reports for a specific Employee
* Training Manager adds a list of new training goals
* Training Manager hires a new Employee and adds the Employee to the system
* Employee takes a simulation test, views the score report of the simulation, and then quits the program
* Auditor views a score report for a specific Employee and then checks the total time spent training by that Employee.
* Administrator adds new Training managers to the system. Administrator then removes a Training Manager from the system.

2.2 - Organizational Structure



2.3 - Organizational Interfaces

|  |  |  |
| --- | --- | --- |
| **Organization** | **Liaison** | **Contact Information** |
| Customer: FAA | Professor Broadwater | rbroadwater@towson.edu |

2.4 - Project Responsibilities

|  |  |  |
| --- | --- | --- |
| **Role** | **Description** | **Person** |
| Project Manager | Leads project team; responsible for project deliverables | Greg Friedman |
| Technical Team Leader | Leads testing and debugging process; in charge of finding up-to-date technologies. | Brennan McPherson |
| Front-end programmer | Designs and manages front-end GUI and simulator | Mark Holson |
| Back-end programmer | Writes program logic and manages database | Howard Wheeler |

**3. Managerial Process**

3.1 - Management Objectives and Priorities

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Dimension** | **Fixed** | **Constrained** | **Flexible** |
| Cost |  |  | X |
| Schedule | X |  |  |
| Scope (functionality) |  | X |  |

Table F-3: Flexibility Matrix

3.2 - Assumptions, Dependencies, and Constraints

Due to the deadline of May 2015, the project team is under a fixed schedule. This being the only fixed project dimension, it holds the highest priority. The user requirements detailed out the functionalities of the project, each of which will be assigned a priority. In case of a rapidly approaching deadline, certain functionalities can be omitted that are less important.

3.3 - Risk Management

The four categories of risk that are relevant to this project:

1. Risks with respect to the work
2. Risks with respect to the management
3. Risks with respect to the resources
4. Risks with respect to the customer

For each category, we will detail possible risks for the project and steps for prevention.

3.3.1 - Risks with respect to the work

1. **Miscommunication**

Prevention: Weekly meetings should be used to go over all current projects thoroughly in detail. Meeting minutes are kept so team members know who is working on what section. In case of a miscommunication arising, all members know who to go to with questions or concerns.

1. **Time shortage**

Prevention: Because the deadline is fixed, dealing with a time shortage puts pressure on all members of the team. To avoid issues, user requirements are ranked based upon priority. With clearance from the client and management, requirements with low priority are put aside.

1. **Errors in design**

Prevention: Proper planning and understanding of design process. Approach design in incremental steps.

1. **Absent team members**

Prevention: Keeping everyone informed and up-to-date on what is happening with the project.

3.4 - Monitoring and Controlling Mechanisms

The project team will have group meetings every week on Thursday’s at 7pm. Meetings are held to review and refine the work from the week before, plan for the next week in regards to deadlines of deliverables, and the progress towards each milestone.

3.5 - Staffing Approach

We will need at least two programmers, one that will focus on front-end (GUI) and one that will focus on back-end (logic). We already have the team set, so we do not need to recruit anyone else. Programming team will need to learn how to integrate the flight simulator into the GUI as well as the programming logic.

**4. Technical Process**

4.1 - Methods, Tools, and Techniques

This project will be designed in Java. As an air traffic control simulator, we are going to use an open source software called jATC. (found here: <http://sourceforge.net/projects/jatc/>) It will allow us to modify difficulties and other functionalities of the simulator. To keep track of version control, we will be using GitHub. The project repository can be found at: <https://github.com/Lifeboat13/Software-Engineering-Team-1>. For development methodology, we will be taking an Agile approach, allowing for significant change in project requirements, which was suggested by the customer.

4.2 - Naming Conventions

Table taken from: http://en.wikipedia.org/wiki/Naming\_convention\_%28programming%29#Java

|  |  |  |
| --- | --- | --- |
| Identifier Type | Rules for Naming | Example |
| Classes | Class names should be nouns in UpperCamelCase, with the first letter of every word capitalised. Use whole words — avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML). | * class Raster; * class ImageSprite; |
| Methods | Methods should be verbs in lowerCamelCase or a multi-word name that begins with a verb in lowercase; that is, with the first letter lowercase and the first letters of subsequent words in uppercase. | * run(); * runFast(); * getBackground(); |
| Variables | Local variables, instance variables, and class variables are also written in lowerCamelCase. Variable names should not start with underscore (\_) or dollar sign ($) characters, even though both are allowed. This is in contrast to other coding conventions that state that underscores should be used to prefix all instance variables.  Variable names should be short yet meaningful. The choice of a variable name should be mnemonic — that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters. | * int i; * char c; * float myWidth; |
| Constants | Constants should be written in uppercase characters separated by underscores. Constant names may also contain digits if appropriate, but not as the first character. | * static final int MAX\_PARTICIPANTS = 10; |

4.2.1 - Software Requirements Specification (SRS)

The SRS clearly and precisely describes each of the essential requirements (functions, performances, design constraints, and attributes) of the software and the external interfaces. Each requirement is defined such that its achievement is capable of being objectively verified and validated by a prescribed method, for example, inspection, analysis, demonstration, or test.

4.2.2 – Functional Requirements

* Administrators, Training Managers, Employees, and Auditors must be able to login to the program
* Software must store user information including: Username, Password, First Name, Last Name, Employee ID (EID) and Department of Employment.
* Software must store lessons and simulation tests based on specific Air Traffic Control information.
* Software shall allow Employees to select “lesson goals” based off of lessons stored in the program.
* From those “lesson goals,” software will compile numerous lessons and exercises related to the chosen lesson goals to create a unique “Lesson Plan” for the employee to study.
* Software shall launch an Air Traffic Control simulation for the employee to take and score the employee based on performance as well as elapsed time of the simulation.
* Software shall store passed lesson scores and a total elapsed time of training for every employee.
* Software shall allow Employees to view their scores for each simulation test taken.
* Software shall also allow Training Managers to view scores of every employee in their department.
* Software shall allow Auditors with valid “EID” to view times of employees simulation trials for legal reasons.

4.2.3 - Non-Functional Requirements

* Software must be fully documented
* Software must be privately accessed in house or via secure remote connection
* Software must have an intuitive GUI with easy to follow instructions
* Software must be able to be maintained and kept up to date

4.2.4 - Software Test Plan

We will use unit testing throughout the software and ensure that every commit to the repository passes all unit tests.

4.3 - User Documentation

The LCMS will provide online documentation and manuals how to use the system. In addition, the code will be well commented and documented for future updates.

**5. Work Packages, Schedule, and Budget**

5.1 - Work Packages

Setup the data structures required

Create the database

Create functionality according to use-cases

Create GUI and tie together functionality (no simulator)

Create/modify simulator to our needs and integrate

5.2 - Schedule

Provide the schedule for the various project functions, activities, and tasks, taking into account 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.

|  |  |  |
| --- | --- | --- |
| Item | Due Date | Submission Date |
| Requirement & Use Case Doc. | Fed 17 | Feb 17, in Repo Apr 1 |
| High Level Architecture | Feb 26 | Feb 30, in Repo May 7 |
| Object Classes | Feb 26 | Feb 30, in Repo May 7 |
| Sequence Diagram | Feb 26 | Feb 30, in Repo May 7 |
| Repository Setup | March 6 | Feb 26 |
| Software Management Plan | March 6 | March 6, in Repo April 2 |
| Presentation to client | March 26 | March 26 |
| Technical Review | April 30 | April 29, in Repo May 7 |
| Unit Tests | May 7 | May 6 |
| Integration Tests | May 7 | May 6 |
| System Test Plan | May 7 | May 7 |
| Deployment Plan | May 7 | May 7 |
| Present Product to Client | May 14 | Estimated date May 13 |

5.3 - Budget

There is not a budget specified for this project. Any costs will be discussed with the customer and then the SPMP will be updated.