**PROFESSOR SCHEDULE MANAGEMENT SYSTEM (PMS)**

**SPECIFICATION-BASED TEST DOCUMENT**

**Prepared by Team 5**

Alejandro Perez

David Rocha

Raul Espinosa

Dayana Gil

Samuel Yorizzo

**Prepared For**

Professor Peter Clarke

Fundamentals of Software Testing, CEN4072 - U01

Florida International University, Spring 2019

February 22nd, 2019

**Abstract:**

This document contains the specification-based testing techniques applied to the Professor Schedule Management System (PMS) software. Testing applied to the software serves as validation of its intended use and is based on the use cases outlined in the software documentation and features implemented by the software designers.

Outlined in the document is the system overview, system requirements and testing methodology. Procedures and documentation are included for testing done using JUnit, Mockito and Rational Functional Tester in the Eclipse IDE. Test cases and results are documented for all unit, subsystem and system test conducted by the testing team.

Table of Contents

**Abstract1**

**Table of Contents2**

**1. Introduction 3**

1.1 Overview of System 3

1.2 Requirements of the System 3

1.2.1 Functional Requirements 3

1.2.2 Non-Functional Requirements 7

1.3 Overall Testing Approach 7

1.3.1 Unit Testing Approach 9

1.3.2 Subsystem Testing Approach 9

1.3.4 System Testing Approach 9

1.4 Terminology 9

1.5 Document Organization 10

**1. Introduction**

This chapter contains an overview of the system being tested, its requirements, as well as documentation on the testing approach, testing terminology, and overall organization of the document.

The software being tested, Professor Schedule Management System, is henceforth referred as PMS throughout the length of this document. All references to PMS encompass the entire original artifact being tested.

**1.1 Overview of System**

PMS software was designed with the intent to provide Florida International University professors with a tool to add, manage, and track class schedules. At the time of the software’s conception, no tool was available to provide this functionality, and this served as the motivation behind the creation of the program.

The intended functionality of the software would provide professors with an easily accessible interface to add class schedules, identify schedule conflicts, manage and edit current schedules, and provide time alerts during classes.

**1.2 Requirements of the System**

This section contains the full list of system requirements for the PMS software. Section 1.2.1 contains the functional requirements, and section 1.2.2 contains the non-functional requirements. All requirements presented are elicited from the original PMS software documentation.

**1.2.1 Functional Requirements**

Functional requirements are taken from the documentation provided by the software designers and are displayed with the use case identification and description.

|  |  |
| --- | --- |
| PMS\_001 – Login | * The system shall provide the user with a login template. * The system shall take the user’s username and password. * The system shall validate the user’s username and password. * The system shall allow or deny access to the user. |
| PSM\_002 – Logout | * The system shall provide the user with a logout template. * The system shall process logout request by the user. * The system shall provide the option to cancel logout process. |
| PSM\_003 – Security | * The system shall provide the user with a template to manage idle auto logout. * The system shall provide option to select best feature according to user’s preference. * The system shall provide the option to cancel request process. |
| PSM\_004 – Schedule Setup | * The system shall provide the user with a schedule setup template. * The system shall process user’s entered data. * The system shall validate user’s entered data. * The system shall check for conflicts in the schedule. * The system shall save the new schedule. * The system shall provide the option to cancel schedule setup process. |
| PSM\_006 - Alarms | * The system shall allow the users to set up timing related alarms with different messages and sounds. |
| PSM\_007 – Custom Warning | * The system shall allow the users to set up custom warnings |
| PSM\_008 – Message Popup | * The system shall warn the user about the current situation by displaying a pop-up message and audible sound. |
| PSM\_009 – Program Priority | * The system shall give warnings and pop-ups high priority over other programs. |
| PSM\_010 – Exam Set-up | * The system shall provide the user with a template to schedule exams. * The system shall process user’s entered data. * The system shall allow the user to set up warnings for exams. * The system shall save the changes made. * The system shall display current time and time left. * The system shall display warnings and notifications * The system shall provide the option to cancel exam set-up process. |
| PSM\_011 – Presentation setup | * The system shall provide the user with a presentation setup template. * The system shall process the user’s entered data. * The system shall record changes made. * The system shall display team’s name and time left. * The system shall display a notification when the time for presentation has ended. * The system shall display time until next presentation starts. * The system shall provide the option to cancel presentation setup process. |
| PSM\_012 – Edit Schedule | * The system shall provide the user with a template to edit schedules. * The system shall process the user’s entered data. * The system shall validate the user’s entered data. * The system shall check for conflicts in the schedule. * The system shall save the edited schedule. * The system shall provide the option to cancel edit schedule process. |
| PSM\_013 – Remove Schedule | * The system shall provide the user with a template to remove schedules * The system shall display confirmation message before removing a schedule * The system shall remove the schedule data. * The system shall provide the option to cancel schedule remove process. |
| PSM\_014 – End of Semester Schedule Clear | * The system shall provide the user with a template to input first schedule. * The system shall allow the user to input the last day of schedule * The system shall compare the data with professor’s last day. Later days will get priority. * The system shall provide the user the option to modify last day of schedule. |
| PSM\_015 – Data Validation | * The system shall verify every field of inputted data is correct * The system shall notify the user when data is incorrect or wrong format. |
| PSM\_016 – Schedule Conflicts | * The system shall verify every field of inputted data format * The system shall notify the user when data is incorrect or wrong format |
| PSM\_017 – Password Conflicts | * The system shall verify every field of inputted data is correct * The system shall give an error to user letting know the password is not correct or does not match the system. * The system shall enable the use of password problem button * The system shall open a new page letting user answer a question in or der to reset password. |
| PSM\_018 – Custom Feature | * The system shall provide the user with a template for features. * The system shall display the features selection of options and tab menus. |
| PSM\_019 – Late Notice for Professor | * The system shall check login time with current. * The system shall notify professor when he/she is late. |
| PSM\_020 – Single Day | * The system shall display a menu that the user can enter data. * The system shall create a single day function to apply data. * The system shall display a window to confirm creation of single day with inputted data. * The system will allow the user to confirm creation of single day. |
| PSM\_021 – Professor Attendance | * The system shall check the time and date that the login occurred and store in records. * The system shall check to see if during the previous log in to the most recent if the professor has missed any classes during the period. * The system shall notify the professor the dates he was missing from class and the closes to continue normal system run time. |
| PSM\_005 – Help Password | * The system shall validate username * The system shall provide the user with a security password template. * The system shall validate user’s secret question. * The system shall provide a new password template for user. * The system shall validate new password. * The system shall save new password. |

**1.2.2 Non-Functional Requirements**

The system is specifically design for Florida International University (henceforth referred to as FIU) professors and all design features and implementations were created based on this premise. All features are specific to the FIU professors needs and FIU’s system architecture.

Organizational structure of the software must be compatible with FIU’s scheduling system.

PSM’s scheduling setup and overall features must be dependable for professor’s usage.

PSM’s performances must be reliable and predictable for professor’s usage.

PSM must be available and compatible with every professor’s system.

**1.3 Overall Testing Approach**

This section contains information on the overall testing approach used to validate PSM’s implementation against the use cases outlined in section 1.2.

The testing of the program is done using specification-based testing techniques and all testing is done following the specifications outlined in the requirements. Testing follows the program’s three tier architecture (interface tier, logic tier and data tier) to test each tier separately and together.

The selection of the test set for test cases is done through equivalence partitioning and boundary analysis. Furthermore, in order to test the tiers thoroughly, testing is done in three categories: Unit testing, subsystems testing and systems testing. Following information shows further details on each of these subdivisions of testing.

A close up of a sign

Description automatically generatedFigure 1.3.1 Test Plan Flowchart.

**1.3.1 Unit Testing Approach**

Unit testing is conducted on two specific classes of the PSM software, the appController class and the DBConnection class. The classes are isolated from the system and states, variable and methods are tested independent of the rest of the system. Test cases are created using equivalence partitioning and boundary analysis. Tests are conducted in the Eclipse environment using JUnit and Mockito testing tools.

**1.3.2 Subsystem Testing Approach**

Subsystem testing is conducted on the logic package of the PSM software. The package is isolated from the rest of the system and tested independently from others. Dependency classes and calls are mocked to control input and output for testing purposes. Test cases are created using equivalence partitioning and boundary analysis. Tests are conducted in the Eclipse environment using JUnit and Mockito testing tools.

**1.3.3 System Testing Approach**

System testing is conducted on the entire PSM software. Test cases are created using equivalence partitioning and boundary analysis and validated against the software’s use cases. Test are conducted in the Eclipse environment using IBM’s Rational Functional Tester.

**1.4 Terminology**

|  |  |
| --- | --- |
| PMS | Professor Schedule Management System |
| JUnit | Testing framework used for unit and subsystem testing. Creates and calls test cases. |
| Mockito | Testing framework used for unit and subsystem testing. Creates mocks classes to use in testing. |
| RFT | Rational Functional Tester. Testing framework used in system testing. Automated system testing that mimics the actions of user. |
| Eclipse IDE | Java environment used to create and run the software and tests. |
| FIU | Florida International University |
| SQL | Relational Database System |

**1.5 Document Organization**

This section explains the organization and characteristics of the documentation in the following chapters and sections. Beyond this section, the following 5 chapters will contain documentation relating to the testing of the PSM software.

Chapter 2 explains the initial setup and preparations for testing, including details on the roles of testing members, test planning, test scheduling, hardware and software used for testing, and features tested and not tested.

Chapters 3, 4 and 5 detail test case documentation for unit testing, subsystem testing and system testing respectively.

Chapter 6 contains the summary report of the test cases documented in sections 3, 4 and 5.

Chapter 7 contains risks and contingencies encountered and analyzed in the process of testing.

**2. Specification Test Plan**

**2.1 Organization**

**2.2 Hardware and Software Requirements**

**2.3 Test References Items**

**2.4 Tested Features**

**2.5 Features Not Tested**

**2.6 Work Breakdown**

**3. Unit Testing**

**3.1 Unit Test Cases**

**3.1.1 Test Identification and Objective**

**3.1.2 Test Criteria and Procedures**

**3.1.3 Test Cases**

**3.2 Actual Test Results**

**4. Subsystem Testing**

**4.1 Unit Test Cases**

**4.1.1 Test Identification and Objective**

**4.1.2 Test Criteria and Procedures**

**4.1.3 Test Cases**

**4.2 Actual Test Results**

**5. System Testing**

**5.1 Unit Test Cases**

**5.1.1 Test Identification and Objective**

**5.1.2 Test Criteria and Procedures**

**5.1.3 Test Cases**

**5.2 Actual Test Results**

**6. Test Summary Report**

**7. Risks and Contingencies**

**8. Approvals**

**9. Glossary**

**10. Appendix**

**10.1 Appendix A – Test schedule**

**10.2 Appendix B – Use Cases (implemented only)**

**10.3 Appendix C – Example of well documented test drivers, stubs used during unit and subsystem testing**

**10.4 Appendix D – Screen shots of the GUI testing tools used for one system test case.**

**10.5 Appendix E – Diary of meeting and tasks**