

Professor Schedule Manager Implementation-Based Test Document

Team Number 3

Denis Ortega

Luis Gonzalez

Antonio Valdes

Daniel Neel

Miguel Martinez

Dario Quintanilla

CEN 4072 - Software Testing

Section U01

Professor Peter Clarke

04/19 /2019

ABSTRACT

Professor Schedule Manager is the system at test. PSM is a software that allows professors or instructors the ability to time a lecture and will be provided alerts periodically. We are trying to determine errors in the given software with specific software testing tools. Using software testing tools, such as JUnit, Mockito, and Rational Function Tester we are able to provide the accurate results needed to determine the functionality of the software. We were able to control the PSM_Logic, PSM_Storage, PSM_Interface of the program. The results varied depending on the dependencies between packages. During testing we found bugs that caused multiple failures of the program and thus can implement a plan to improve the functionality of the given software.

We also analyzed the coverage of our tests using various tools. We measured our initial coverage using tools like Clover, Eclipse, Code Cover, and Cobertura. For any section of the problem that gave us less than 80% initial branch and statement coverage, we wrote new test cases until we reached the correct threshold. Ultimately, we were able to obtain adequate test coverage for Unit, System, and SubSystem testing.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION	5
1.1 Overview of System	5
1.2 Requirements of the System	5
1.3 Overall testing approach	8
1.4 Terminology	11
1.5 Document Organization	11
CHAPTER 2: IMPLEMENTATION TEST PLAN	12
2.1 Organization	12
2.2. Hardware and Software Requirements	13
2.3. Test Reference Items	16
2.4. Tested Features	16
2.6. Work Breakdown	17
CHAPTER 3: Unit Testing	18
3.1. Unit Test Cases	18
3.1.1. Test Identification and Objective	18
3.1.2. Test Criteria and Procedures	37
3.1.3. Test Cases	38
3.2. Actual Test Results	86
CHAPTER 4: Subsystem Testing	92
4.1. Subsystem Test Cases	92
4.1.1. Test Identification and Objective	93
4.1.2. Test Criteria and Procedures	99
4.1.3. Test Cases	101
4.2. Test Results	116
CHAPTER 5: System Testing	119
5.1. System Test Cases	119

5.1.1. Test Identification and Objective	119
5.1.2. Test Criteria and Procedures	126
5.1.3. Test Cases	127
5.2. Actual Test Results	144
CHAPTER 6: Test Summary Report	148
6.1 Failures and Possible Solutions	148
6.2. Code Coverage Results	157
CHAPTER 7: Risks and Contingencies	162
CHAPTER 8: Approvals	163
CHAPTER 9: Glossary	164
CHAPTER 10: Appendix	165
10.1. Appendix A	165
10.2. Appendix B	165
10.3. Appendix C	185
10.4. Appendix D	189
10.5. Appendix E	195

CHAPTER 1: INTRODUCTION

The introduction is organized by the following categories, overview of the system, requirements of the system which represents all of the used test cases, the overall testing approach – i.e., unit testing, subsystem, and system testing, the terminology used and the document organization.

1.1 Overview of System

The Professor Schedule Manager, or PSM for short, is intended to aid professors in tracking time which translates to optimized efficiency in the classroom setting. After the user logs in with valid credentials, the user can then access the software. Once logged in, the user can set up a schedule and begin using the system for functions such as setting reminder alarms for the end of class.

1.2 Requirements of the System

The specification document developed eight use cases. The cases are the following:

1.) PSM_001-Login

The user shall login to access the software. The user shall be provided a template in which a username and password can be entered. Once the credentials are entered, the system will allow access to the user after a validation process. As a post condition, the user shall be able to start

using the software. The system provides a constraint in the form of a help frame, an average of one minute should be taken to complete the form.

2.) PSM_002-Logout

Logout shall allow users to exit the system. The user shall request to be logged in and out of the system. The system shall provide a constraint in the form of the user taking thirty seconds to complete a log in or log out request.

3.) PSM_003-Security

Security case shall log the user out when software detects user to be idle over a period of time. The system will track the time and if positive shall log the user out. The system shall record changes made by the previous user. The system shall provide a constraint in the form of handling one request per user.

4.) PSM-004-Schedule Set-up

Schedule Setup is the first form the user shall utilize to record schedule into the system. A user will be logged onto the system. The system shall provide the user with a template to fill which is recorded and validated. The user shall be given access to the software after completion. The average time a user shall take to complete the form is fifteen minutes.

5.) PSM_008-Message Pop-up

Message Pop-up shall serve as a warning system. The application shall send a pop-up message to the teacher when time is about to finish. The system shall use the recorded end time to measure how much time was left. The system shall inform the user that time is about to finish. A user shall input acknowledgement to accept the pop-up.

6.) PSM_012-Schedule Edit

Edit Schedule will allow users to modify the schedule after initial setup. Users logged in the system shall click on “Edit Schedule” to proceed. The system shall provide the previously filled form given at the prior Schedule set-up. The user shall input and track the changes. These changes shall be recorded and validated. System shall record said changes. User shall not take longer than one minute.

7.) PSM_014- End of Semester Clear

The End of Semester Clear shall delete the information about the class after the semester end. The system shall use the the latest date the user has input as the final date of the semester and as the date when it deletes everything. The user will be able to modify the last day. A comparison shall be made and the system will delete the information in the later day

8.) PSM-017-Password Conflicts

Password conflicts shall allow the user to reset their password. It shall start after the wrong username/password combo is entered. The user shall click on the password problem button

which leads to a screen where they answer a security question. The user shall input newly created password into the login screen and the user is able to login.

1.3 Overall testing approach

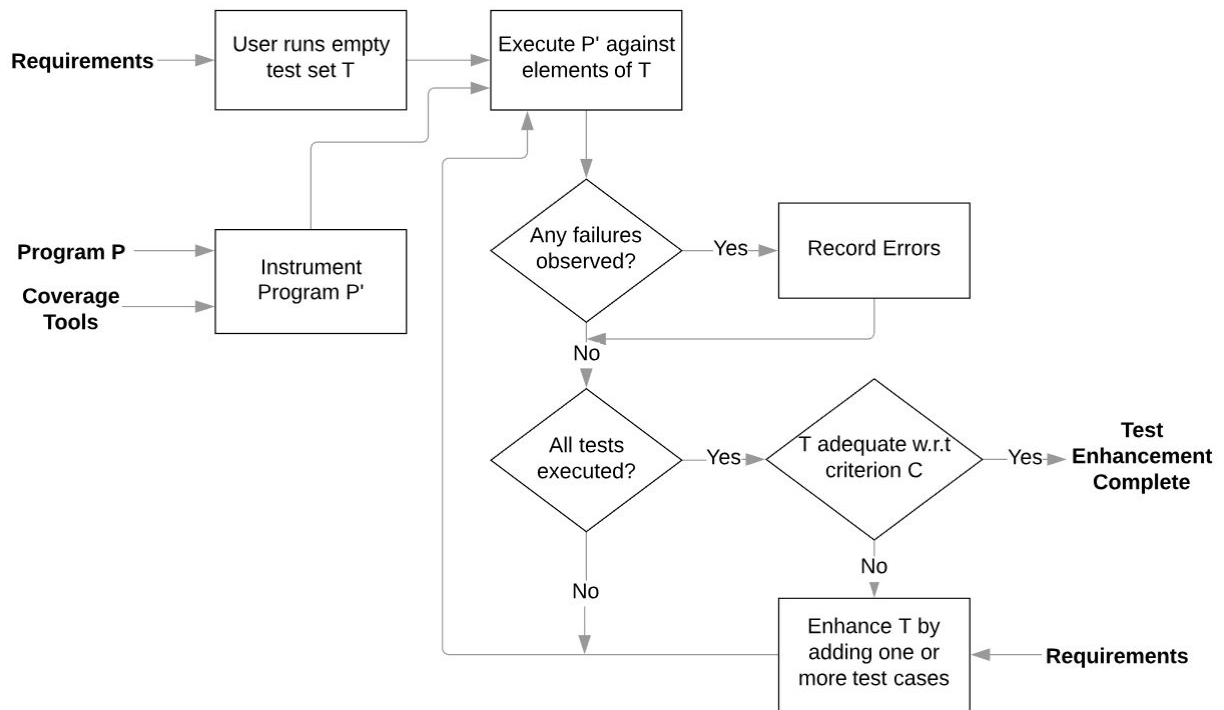


Figure 1.0 Overall Testing Approach:

Figure 1.0 is a chart of the general testing process used when testing against the PSM program. On our original test set we executed the tests, observed and recorded failures. However, in this

iteration we have a new criterion C to reach which is minimum 80% statement and branch coverage. The true goal is always 100% coverage but often time this is infeasible. If this goal was not reached, more test cases were created to meet this requirement and the process would restart. If this goal was met, then the test enhancement was complete.

Testing techniques that were used included suspicion based testing, equivalence partitioning, and control and data flow. Suspicion based testing was based on our intuition and understanding of how the functionality of the system is *supposed* to work. Equivalence partitioning was only used when testing for a legal or illegal value as an input to one of our tests. Control and data flow were used in regards to coverage, especially branch coverage. Understanding control and data flow let us see what values were necessary to reach certain branches in the flow of the code, for example in all the branches of the main, certain variables need to be changed in order to reach them. Using coverage tools such as Clover, EclEmma, and CodeCover allowed us to see what branches were missing coverage in the editor.

Unit testing - For unit testing, the process we used was based on the techniques mentioned above. Most methods either had an illegal or illegal value if it required input. The legal values were generally tested first, using the system as it should be used and not how the client might actually use it. The second method, suspicion based testing, was used after carefully analyzing each method and intuitively seeing how it was meant to be used and or how that method could be used incorrectly. Lastly, as stated above the coverage tools were used to see the data flow and find the missing coverage for branches and statements.

Subsystem testing - For subsystem testing, the main process included deleting the interface package. Upon deletion of the interface package, the connected classes to the interface package reveals the errors in the system between that interaction. These errors are connected to the logic package. The Interface_Controller class had errors in every method. The AppController class had errors in the main method, PopUp15Minute method, PopUp5Minute method, and the End_Of_Class method.

System testing - For system testing, the tool used was Rational Functional Tester(RFT) to record the exact steps the tester went through to complete the feature currently being tested. While you go through the program insert verification points to confirm the state is consistent with every execution of the test. Once recording is done, simply playback the recorded test and analysis the TPTP summary to see if the test passed or failed and at which points it failed and passed at.

1.4 Terminology

Due to the simplicity of the program there are some simple acronyms and words used to describe the Professor Schedule Manager software.

- PSM - Refers to Professor Schedule Manager
- Coverage - Refers to how much of our code is executed when are test suite runs

1.5 Document Organization

This paper outlines the purpose, design, and testing of the system. In chapter 1 we discuss the Organization of team roles. Chapter 2 focuses on the hardware and software requirements of this document, including but not limited to, the resources required to complete testing of the project. In Chapters 3, 4, and 5, we discuss unit, subsystem, and system testing respectively including test cases, results, and other information. Chapter 6 discusses the test summary report which is compilation of all testing results, including coverage and failures. Chapter 7 discusses the risks and contingencies of testing. Chapter 8 has the team members' approvals. Chapter 9 is the glossary for this document. Finally, Chapter 10 is the appendix which discusses other project-related information, such as the work breakdown for the entire project and the milestones reached as a group.

CHAPTER 2: IMPLEMENTATION TEST PLAN

The Implementation test plan is organized by the following categories, organization, hardware and software requirements, which are the resources required to complete proper testing of the project. Test reference items, tested features, features not tested, and work breakdown, which describes milestones reached during testing.

2.1 Organization

Below is a table that shows the organization of the roles for the team throughout the project.

Group Member	Deliverable 1	Deliverable 2
Denis Ortega	Lead Unit Tester	Minute Taker/ Lead Unit Tester
Miguel Martinez	Lead Subsystem Tester/ Minute Taker	Lead Subsystem Tester
Dario Quintanilla	Subsystem Tester/Time Keeper	Team Leader/Backup Subsystem Tester
Daniel Neel	Lead System Tester/ Subsystem Tester	Time Keeper/Lead System Tester
Antonio Valdes	System Tester/ Subsystem Tester	Backup System Tester
Luis Gonzalez	Team Leader/Backup Unit	Backup Unit Tester/ Backup

	Tester	Subsystem Tester
--	--------	------------------

2.2. Hardware and Software Requirements

Software Requirements

The software needed to execute these tests and review coverage are listed below:

- Windows 10
- Eclipse IDE - 4.11 - used as the IDE to run our testing
- Eclipse Kepler - 4.3 used to run CodeCover and EclEmma 1.5.3
- Rational Functional Tester 9.2.1 - used for systems testing
- JUnit 5.4.2 - used to run unit and subsystems tests on Eclipse
- Mockito 1.10.1 - used to mock and stub classes
- PowerMockito 1.7.1 - an extension of Mockito used to mock static classes and use reflection
- MySQL Workbench 8.0 - used to stage the database
- Apache Ant 1.10.5- used to compile Java files and run Cobertura with Rational Functional Tester

Coverage Tools:

- Cobertura 2.11
- Clover 4.12
- EclEmma 1.5.3 and 3.1.2
- CodeCover 1.0.1.2

Hardware Requirements

The most demanding softwares used are the Eclipse IDE, Eclipse Kepler, IBM Rational Functional Tester, and MySQL Workbench. All other jar files such as JUnit, Mockito and the coverage tools were only a few MB so they are negligible.

Minimum Requirements:

- Memory = 4 GB RAM
- Free Disk Space = ~5 GB
- Processor: Intel i5 3 Ghz

The software testing tools include JUnit, Mockito, and Rational Function Tester. **JUnit** is a unit testing framework for the Java programming language. JUnit is linked as a JAR at compile-time. The framework resides under package org.junit for at least JUnit 4.

Mockito is an open source testing framework for Java released under the MIT license. This framework allows for the creation of double objects formally known as “mock objects”. The purpose for Mockito is test driven development.

Rational Functional Tester is a tool for automated testing of software applications. It allows users to create tests that mimic actions and assessments of a human tester. RFT is used by quality assurance teams. Testers create scripts using a test recorder which captures a users actions against their application that is being tested.

Other Software used for testing included:

- Windows 10/Mac
- Eclipse IDE - used as the IDE to run our testing
- PowerMockito 1.7.1 - an extension of Mockito used to mock static classes and use reflection
- MySQL Workbench 8.0 - used to stage the database

Hardware Requirements

The most demanding softwares used are the Eclipse IDE and IBM Rational Functional Tester. All other jar files such as JUnit and Mockito were less than ~1MB, so they are negligible.

Minimum Requirements:

- Memory = 2 GB RAM
- Free Disk Space = ~1.5 GB
- Processor Speed = 1.5 Ghz

2.3. Test Reference Items

In chapter 4, page 8 of the software application document, the proposed system and functionality gets introduced. The proceeding page describes the interface of logging into the Professor Schedule Manager software.

2.4. Tested Features

Features Tested	Features Not Tested
PSM_001 - Login	
PSM_002 - Logout	
PSM_003 - Security - IdleLogout	
PSM_004 - ScheduleSetUp	
PSM_008 - MessagePopUp	
PSM_012 - ScheduleEdit	
PSM_014 - EndofSemesterClear	
PSM_017- PasswordConflicts	
Logic Package - SubSytem Testing	
AppController - Unit Testing	

DBConnection - Unit Testing	
-----------------------------	--

2.6. Work Breakdown

This subsection has the identification of milestones and deliverables during testing of system i.e. (Refer to project schedule in Appendix A and the diary in appendix B).

Task	Date Range
Set Up Coverage Tools	March 4 - April 17
Write Additional Test Cases	March 4 - April 15
Develop Presentation	April 1 - April 15
Develop Deliverable Doc	March 23 - April 19

CHAPTER 3: Unit Testing

Unit testing focuses on testing individual features in a class, in this case the individual methods. For unit testing of the Professor Schedule Manager, we tested the ApplicationController class in the PSM_Logic package and the DBConnection class in the PSM_Storage package. (Refer to Section 1.3 for notes on the testing approach).

3.1. Unit Test Cases

The following section outlines all of the test cases beginning with the first set for specification based testing and then the new test cases added as part of implementation based testing.

3.1.1. Test Identification and Objective

(Summary) – unique identifier, purpose of each test.

Test ID	Unit001_AppCont001
Purpose	To assert that checkClear() checks if the calendar is reset and returns true

Test ID	Unit002_AppCont002
Purpose	To test that the checkTimes() method schedules the popup for 5 min, 15 min, and the end of class, if there is a course on that day.

Test ID	Unit003_AppCont003
Purpose	To ensure that the instance variables are set to the data from the database.

Test ID	Unit004_AppCont004
Purpose	To ensure that the current thread is set to sleep for 100 milliseconds with the sleep(int milli) method.

Test ID	Unit005_AppCont005
Purpose	To verify that the appController sets the loggedIn state to true when LogIn() is called.

Test ID	Unit006_AppCont006
Purpose	To ensure that the getCon() method returns the correct DBConnection object.

Test ID	Unit007_AppCont007
Purpose	To test that the dbClear.run() method attempts to clear the database.

Test ID	Unit008_AppCont008
Purpose	To test that the popup15min.run() attempts to display a fifteen minute warning message.

Test ID	Unit009_AppCont009
---------	--------------------

Purpose	To test that the popup5min.run() attempts to display a five minute warning message.
---------	---

Test ID	Unit010_AppCont010 - TimerTask endofclass
Purpose	To test that the endofclass.run() method attempts to display a end of class message and changes the classEnded state variable to the current System time.

Test ID	Unit011_AppCont011 - TimerTask systemExit
Purpose	To test that the systemExit.run() method attempts to run the System.exit(0) command.

Test ID	Unit012_AppCont012 - setTime(...)
Purpose	To test if the setTime() method in appController sets the Calendar setRun state variable to the specified time.

Test ID	Unit013_AppCont013 - getTime()
Purpose	To test if the getTime() method returns the time set using setTime().

Test ID	Unit014_AppCont014 - getTimeMillis()
Purpose	To test that the getTimeMillis() method returns the time in milliseconds since the default time defined in the Calendar class after the time was set with setTime().

Test ID	Unit015_AppCont015 - timerParser()
---------	------------------------------------

Purpose	To test that the timerParser() method parses the inputted time into the respective state variables.
---------	---

Test ID	Unit016_AppCont016 - dateParser()
Purpose	To test that the dateParser() method parses the inputted date into the respective state variables.

Test ID	Unit017_AppCont017 - returnHr()
Purpose	To verify that the returnHr() method returns the correct value for the hr state variable.

Test ID	Unit018_AppCont018 - returnMin()
Purpose	To verify that the returnMin() method returns the correct value for the min state variable.

Test ID	Unit019_AppCont019 - getEndTime()
Purpose	To test the getEndTime() method returns today's Date but 1 mins before the specified parameter's time.

Test ID	Unit020_AppCont020 - setSemesterClear()
Purpose	To test that the setSemesterClear() method updates the autoClear and date2 state variable to the specified inputs.

Test ID	Unit021_AppCont021 - getSemesterClear()
Purpose	To test that the getSemesterClear() method returns the date of the autoClear state variable.

Test ID	Unit022_AppCont022 - get15BeforeEnd()
Purpose	To test the get15BeforeEnd() method returns today's Date but 15 mins before the specified parameter's time.

Test ID	Unit023_AppCont023 - get5BeforeEnd()
Purpose	To test the get5BeforeEnd() method returns today's Date but 5 mins before the specified parameter's time.

Test ID	Unit024_AppCont024 - autoExit()
Purpose	To test that the autoExit() method schedules a system exit for immediate execution.

Test ID	Unit025_AppCont025 - autoClear()
Purpose	To test that the autoClear() method schedules to clear the database for immediate execution.

Test ID	AppCont026 - checkClear()
Purpose	To assert that checkClear() checks if the calendar is reset and returns false.

Test ID	Unit027_AppCont027 - checkTimes()
Purpose	To test that the checkTimes() method doesn't schedule the popup for 5 min, 15 min, and the end of class since there are no courses.

Test ID	Unit028_AppCont028 - getData(int course)
Purpose	To ensure that the instance variables are empty string when no data is returned from the DBConnection.

Test ID	Unit029_AppCont029 - sleep(int milli)
Purpose	To test that when the Thread class attempts to sleep it throws an exception.

Test ID	Unit030_AppCont030 - LogIn()
Purpose	To verify that the appController sets the loggedIn state to false when LogIn() is called.

Test ID	Unit031_AppCont031 - getTime()
Purpose	To test if the getTime() method returns the time set using setTime().

Test ID	Unit032_AppCont032 - getTimeMillis()
Purpose	To test that the getTimeMillis() method returns the time in milliseconds from the setRun state variable.

Test ID	Unit033_AppCont033 - timerParser()
Purpose	To test that the timerParser() method throws an exception when given an invalid value.

Test ID	Unit034_AppCont034 - dateParser()
Purpose	To test that the dateParser() method throws an exception when given an invalid value.

Test ID	Unit035_AppCont035 - returnHr()
Purpose	To test that returnHr() method returns 0 when not set to any time.

Test ID	Unit036_AppCont036 - returnMin()
Purpose	To test that the returnMin() method returns 0 when no time is set.

Test ID	Unit037_AppCont037 - getEndTime()
Purpose	To test that getEndTime() still returns the correct value when given a time that has its minutes at 00.

Test ID	Unit038_AppCont038 - get15BeforeEnd()
Purpose	To test the get15BeforeEnd() method returns today's Date but 15 mins before the specified parameter's time.

Test ID	Unit039_AppCont039 - get5BeforeEnd()
---------	--------------------------------------

Purpose	To test the get5BeforeEnd() method returns today's Date but 5 mins before the specified parameter's time.
---------	---

Test ID	Unit041 - DBConnect001 - DBConnection.java - fetchCourseName(int courseID)
Purpose	To verify that DBConnection.fetchCourseName(int courseID) returns the proper data state variable when it is called

Test ID	Unit042 - DBConnect002 - DBConnection.java - fetchCourseID(int courseID)
Purpose	To verify that DBConnection.fetchCourseID(int courseID) returns the proper data state variable when it is called

New Test Cases

Test ID	Unit043_DBConnection003 - DBConnection.Java - fetchSemester(int courseID)
Purpose	To verify that DBConnection.fetchCourseSemester(int courseID) returns the proper data state variable when it is called

Test ID	ID: Unit044_DBConnection004 - DBConnection.Java - fetchCourseStart(int courseID)
Purpose	o verify that DBConnection.fetchCourseStart(int courseID) returns the proper data state variable when it is called

Test ID	Unit045_DBConnection005 - DBConnection.Java - fetchCourseSubjt(int courseID)
Purpose	To verify that DBConnection.fetchCourseSubj(int courseID) returns the proper data state variable when it is called.

Test ID	Unit046_DBConnection006 - DBConnection.Java - fetchCourseEnd(int courseID)
Purpose	To verify that DBConnection.fetchCourseEnd(int courseID) returns the proper data state variable when it is called

Test ID	Unit047_DBConnection007 - DBConnection.Java - fetchStartMon(int courseID)
Purpose	To verify that DBConnection.fetchStartMon(int courseID) returns the proper data state variable when it is called

Test ID	Unit048_DBConnection008 - DBConnection.Java - fetchEndMon(int courseID)
Purpose	To verify that DBConnection.fetchEndMon(int courseID) returns the proper data state variable when it is called

Test ID	Unit049_DBConnection009 - DBConnection.Java - fetchStartTue(int courseID)
---------	---

Purpose	o verify that DBConnection.fetchStartTue(int courseID) returns the proper data state variable when it is called
---------	---

Test ID	Unit050_DBConnection010 - DBConnection.Java - fetchendTue(int courseID)
Purpose	To verify that DBConnection.fetchEndTue(int courseID) returns the proper data state variable when it is called

Test ID	Unit051_DBConnection011 - DBConnection.Java - fetchStartWed(int courseID)
Purpose	To verify that DBConnection.fetchStartWed(int courseID) returns the proper data state variable when it is called

Test ID	Unit052_DBConnection012 - DBConnection.Java - fetchEndWed(int courseID)
Purpose	To verify that DBConnection.fetchStartWed(int courseID) returns the proper data state variable when it is called

Test ID	Unit053_DBConnection013 - DBConnection.Java - fetchStartThu(int courseID)
Purpose	To verify that DBConnection.fetchStartThu(int courseID) returns the proper data state variable when it is called

Test ID	Unit054_DBConnection014 - DBConnection.Java - fetchEndThu(int courseID)
Purpose	to verify that DBConnection.fetchEndThu(int courseID) returns the

	proper data state variable when it is called
--	--

Test ID	Unit055_DBConnection015 - DBConnection.Java - fetchStartFri(int courseID)
Purpose	To verify that DBConnection.fetchStartFri(int courseID) returns the proper data state variable when it is called

Test ID	Unit056_DBConnection016 - DBConnection.Java - fetchEndFri(int courseID)
Purpose	To verify that DBConnection.fetchEndFri(int courseID) returns the proper data state variable when it is called

Test ID	Unit057_DBConnection017 - DBConnection.Java - fetchStartSat(int courseID)
Purpose	To verify that DBConnection.fetchStartSat(int courseID) returns the proper data state variable when it is called

Test ID	Unit058_DBConnection018 - DBConnection.Java - fetchEndSat(int courseID)
Purpose	To verify that DBConnection.fetchEndSat(int courseID) returns the proper data state variable when it is called

Test ID	Unit059_DBConnection019 - DBConnection.Java - fetchEndSat(int courseID)
Purpose	To verify that DBConnection.fetchCourseID(int courseID) returns the proper data state variable when it is called

Test ID	Unit060_DBConnection020 - DBConnection.Java - fetchCourseSubj(int courseID)
Purpose	To verify that DBConnection.fetchCourseSubj(int courseID) returns the proper data state variable when it is called

Test ID	Unit061_DBConnection021 - DBConnection.Java - fetchCourseName(int courseID)
Purpose	To verify that DBConnection.fetchCourseName(int courseID) returns the proper data state variable when it is called

Test ID	Unit062_DBConnection022 - DBConnection.Java - fetchCourseSemester(int courseID)
Purpose	To verify that DBConnection.fetchCourseSemester(int courseID) returns the proper data state variable when it is called

Test ID	Unit063_DBConnection023 - DBConnection.Java - fetchCourseStart(int courseID)
Purpose	To verify that DBConnection.fetchCourseStart(int courseID) returns the proper data state variable when it is called

Test ID	Unit064_DBConnection024 - DBConnection.Java - fetchCourseEnd(int courseID)
---------	--

Purpose	To verify that DBConnection.fetchCourseEnd(int courseID) returns the proper data state variable when it is called
---------	---

Test ID	Unit065_DBConnection025 - DBConnection.Java - fetchStartMon(int courseID)
Purpose	To verify that DBConnection.fetchStartMon(int courseID) returns the proper data state variable when it is called

Test ID	Unit066_DBConnection026 - DBConnection.Java - fetchEndMon(int courseID)
Purpose	DBConnection.fetchStartMon(int courseID) returns the proper data state variable when it is called

Test ID	Unit067_DBConnection027 - DBConnection.Java - fetchStartTue(int courseID)
Purpose	To verify that DBConnection.fetchStartTue(int courseID) returns the proper data state variable when it is called

Test ID	Unit068_DBConnection028 - DBConnection.Java - fetchEndTue(int courseID)
Purpose	To verify that DBConnection.fetchEndTue(int courseID) returns the proper data state variable when it is called

Test ID	Unit069_DBConnection029 - DBConnection.Java - fetchStartWed(int courseID)
Purpose	To verify that DBConnection.fetchStartWed(int courseID)

	returns the proper data state variable when it is called
--	--

Test ID	Unit070_DBCConnection030 - DBCConnection.Java - fetchEndWed(int courseID)
Purpose	To verify that DBCConnection.fetchEndWed(int courseID) returns the proper data state variable when it is called

Test ID	Unit071_DBCConnection031 - DBCConnection.Java - fetchStartThu(int courseID)
Purpose	To verify that DBCConnection.fetchStartThu(int courseID) returns the proper data state variable when it is called

Test ID	Unit072_DBCConnection032 - DBCConnection.Java - fetchEndThu(int courseID)
Purpose	To verify that DBCConnection.fetchEndThu(int courseID) returns the proper data state variable when it is called

Test ID	Unit073_DBCConnection033 - DBCConnection.Java - fetchStartFri(int courseID)
Purpose	To verify that DBCConnection.fetchStartFri(int courseID) returns the proper data state variable when it is called

Test ID	Unit074_DBConnection034 - DBConnection.Java - fetchEndFri(int courseID)
Purpose	To verify that DBConnection.fetchEndFri(int courseID) returns the proper data state variable when it is called

Test ID	Unit075_DBConnection035 - DBConnection.Java - fetchStartSat(int courseID)
Purpose	To verify that DBConnection.fetchStartSat(int courseID) returns the proper data state variable when it is called

Test ID	Unit076_DBConnection036 - DBConnection.Java - fetchEndSat(int courseID)
Purpose	To verify that DBConnection.fetchEndSat(int courseID) returns the proper data state variable when it is called

Test ID	Unit077_DBConnection037 - DBConnection.Java - disconnect()
Purpose	To verify that DBConnection.disconnect() returns the proper data state variable when it is called

Test ID	Unit078_DBConnection038 - DBConnection.Java - disconnect()
Purpose	To verify that DBConnection.disconnect() returns the proper data state variable when it is called

Test ID	Unit079_DBConnection039 - DBConnection.Java - disconnect()
---------	--

Purpose	To verify that DBConnection.disconnect() returns the proper data state variable when it is called
---------	---

Test ID	Unit080_DBConnection040 - DBConnection.Java - clearDatabase()
Purpose	To verify that DBConnection.clearDatabase() returns the proper data state variable when it is called

Test ID	Unit081_DBConnection041 - DBConnection.Java - createClassTable()
Purpose	To verify that DBConnection.createClassTable() returns the proper data state variable when it is called

Test ID	Unit082_DBConnection042 - DBConnection.Java - createClassTable()
Purpose	To verify that DBConnection.createClassTable() returns the proper data state variable when it is called

Test ID	Unit083_DBConnection043 - DBConnection.Java - createClassTable()
Purpose	To verify that DBConnection.createClassTable() returns the proper data state variable when it is called

Test ID	Unit084_DBConnection044 - DBConnection.Java - storeClassInfo()
Purpose	To verify that DBConnection.storeClassInfo() returns the proper data state variable when it is called

Test ID	Unit085_DBConnection045 - DBConnection.Java - storeClassSched()
Purpose	To verify that DBConnection.storeClassSched()

	returns the proper data state variable when it is calle
--	---

Test ID	Unit086_DBConnection046 - DBConnection.Java - storeClassSched()
Purpose	To verify that DBConnection.storeClassSched() returns the proper data state variable when it is calle

Test ID	Unit087_DBConnection047 - DBConnection.Java - connect()
Purpose	Unit087_DBConnection047 - DBConnection.Java - connect() Purpose: To verify that DBConnection.createClassTable()

Test ID	Unit088_DBConnection048 - DBConnection.Java - connect()
Purpose	To verify that DBConnection.createClassTable() returns the proper data state variable when it is called

Test ID	Unit089_DBConnection049 - DBConnection.Java - getEndDates()
Purpose	To verify that DBConnection.getEndDates() returns the proper data state variable when it is called

Test ID	Unit090_DBConnection050 - DBConnection.Java - getEndDates()
Purpose	To verify that DBConnection.getEndDates() returns the proper data state variable when it is called

Test ID	Unit091_DBConnection051 - DBConnection.Java - fetchCourses()
Purpose	To verify that DBConnection.fetchCourses() returns the proper data state variable when it is called

Test ID	Unit089_AppCont040 - main()
Purpose	To test the main() method to ensure it executes from the login, iterates through the branches with forms, and then logs out.

Test ID	Unit090_AppCont041 - sleep()
Purpose	To test that sleep() catches an interrupted thread exception.

Test ID	Unit091_AppCont042 - autoExit()
Purpose	To test that autoExit() method attempts to system exit after scheduling an exit.

Test ID	Unit092_AppCont043 - autoClear()
Purpose	To test that autoClear() method attempts to schedule to clear the database after the designated time.

Test ID	Unit093_AppCont044 - checkTimes()
Purpose	To test that checkTimes() parses the time for Monday correctly when the date is Monday and there is a class for Monday.

Test ID	Unit094_AppCont045- checkTimes()
---------	----------------------------------

Purpose	To test that checkTimes() parses the time for Tuesday correctly when the date and class is on Tuesday.
---------	--

Test ID	Unit095_AppCont046 - checkTimes()
Purpose	To test that checkTimes() parses the time for Wednesday correctly when the date and class is on Wednesday.

Test ID	Unit096_AppCont047 - checkTimes()
Purpose	To test that checkTimes() parses the time for Thursday correctly when the date and class is on Thursday.

Test ID	Unit097_AppCont048 - checkTimes()
Purpose	To test that checkTimes() parses the time for Friday correctly when the date and class is on Friday.

Test ID	Unit098_AppCont049 - checkTimes()
Purpose	To test that checkTimes() parses the time for Saturday correctly when the date and class is on Saturday.

Test ID	Unit098_AppCont049 - checkTimes()
Purpose	To test that checkTimes() parses the time for Saturday correctly when the date and class is on Saturday.

3.1.2. Test Criteria and Procedures

This subsection explains the use of drivers and stubs. The focus of the tests will be to improve code coverage and generate new test cases based on control flow or data flow of the code. Note that the procedures include the type of code coverage expected and the percentage of coverage.

The test criteria for these test cases required the usage of JUnit, Mockito, and PowerMockito. These tools allowed us to easily implement a test suite to run a test for each test case. Before all tests, the appController was initialized and the DBConnection reference in appController was mocked with Mockito.

For testing purposes, the appController class was edited in a minor way to make the code testable. All private variables were made public, and since they were also static it was necessary to reset all the variables to null on the tear down method of JUnit. The InterfaceController was also made public in order to mock the instantiation of that class. DBConnection also received minor edits in order to ensure testability. Get methods were implemented in order to pull the mocked objects into the class itself.

An attempt was made to reach 100% branch and statement coverage. In order to achieve this, it was necessary to use control and data flow to trace the loops and if-else branches to cover all these blocks of code. For unit testing, it was expected that branch and statement coverage would reach a high percentage due to all functionality of other classes being ignored and mocked.

The following are certain procedures that were taken when creating test cases against these unit methods:

- Whenever a function accessed an outside class, that class and its method calls were mocked with Mockito.
- SQL calls or other database functions had to be mocked and their functionality had to be simulated using Mockito.
- When a local variable was used in a method, the logic used for that variable was replicated for the test to ensure the functionality was correct. For the new test cases, if a new class was trying to be constructed, PowerMockito was able to mock that instantiation call.
- When static classes/variables were involved, they were mocked using PowerMockito and verified if those method calls were invoked.
- Assertions were either made against the expected behavior, or if all functionality needed to be mocked then the method calls to those respective mocks were verified.
- For coverage, multiple test cases were made to cover multiple branches of methods in order to obtain a high branch coverage. If the method had a loop that iterated multiple times, then this was opportunity was taken to cover all the branches in that loop.

3.1.3. Test Cases

Unique identifier, purpose, environment set up (preconditions), input, expected output.

Test Case ID	Unit001_AppCont001 - checkClear()
Purpose	To assert that checkClear() checks if the calendar is reset and returns true.
Test Setup	appController initialized

	DBConnection mocked ArrayList <String> endDates; endDates.add("01/01/12");
Test Input	appController.checkClear()
Expected Output	checkClear() method returns true

Test Case ID	Unit002_AppCont002 - checkTimes()
Purpose	To test that the checkTimes() method schedules the popup for 5 min, 15 min, and the end of class, if there is a course on that day.
Test Setup	appController instantiated DBConnection mocked ArrayList<Integer> mockCourseList; mockCourseList.add(1); Set each class end time to 6:15.
Test Input	appController.checkTimes()
Expected Output	State variables endofclass, popup5min, and popup15min are scheduled for execution at 6:15, 6:10, 6:00 respectively.

Test Case ID	Unit003 - AppCont003 - getData(int course)
Purpose	To ensure that the instance variables are set to the data from the database.
Test Setup	appController instantiated DBConnection mocked Database data mocked Set course subject to "CEN". Set course semester to "Spring". Set course name to "Software".

	All course start times are set to 5:00. All course end times are set to 6:15.
Test Input	appController.getData(0)
Expected Output	State variables should be: defSub = "CEN" defSemester = "Spring" defCourseName = "Software" For all course starts = "5:00" For all course ends = "6:15"

Test Case ID	Unit004_AppCont004 - sleep(int milli)
Purpose	To ensure that the current thread is set to sleep for 100 milliseconds with the sleep(int milli) method.
Test Setup	appController instantiated PowerMockito.mockStatic(Thread.class);
Test Input	appController.sleep(100);
Expected Output	Verified true that the mocked Thread.class was called

Test Case ID	Unit005 - AppCont005 - LogIn()
Purpose	To verify that the appController sets the loggedIn state to true when LogIn() is called.
Test Setup	appController instantiated DBConnection mocked String user = "username"; String pass = "password"; Mockito.when(db.connect(user, pass)).thenReturn(0);

	Set appController private variables: username = user; password = pass;
Test Input	appController.LogIn();
Expected Output	After setting state variables and calling the LogIn() method, state variable loggedIn in the appController should return true.

Test Case ID	Unit006_AppCont006 - getCon()
Purpose	To ensure that the getCon() method returns the correct DBConnection object.
Test Setup	appController instantiated DBConnection mocked and injected into appController
Test Input	appController.getCon();
Expected Output	Object returned from getCon() should be equal to appController.db state variable.

Test Case ID	Unit007_AppCont007 - TimerTask dbClear
Purpose	To test that the dbClear.run() method attempts to clear the database.
Test Setup	appController instantiated DBConnection mocked Mock the call to clearDatabase() to do nothing. Access dbClear from appController.

Test Input	dbClear.run()
Expected Output	Verify db.clearDatabase() was attempted to run but that the mock was called instead.

Test Case ID	Unit008_AppCont008 - TimerTask popup15min
Purpose	To test that the popup15min.run() attempts to display a fifteen minute warning message.
Test Setup	appController instantiated Mock the messages class Mock the InterfaceController class. Do nothing when msg.FifteenMinWarning(). Access popup15min variable from appController.
Test Input	popup15min.run()
Expected Output	The msg.FifteenMinWarning() mock was attempted to be called but was stubbed.

Test Case ID	Unit009_AppCont009 - TimerTask popup5min
Purpose	To test that the popup5min.run() attempts to display a five minute warning message.
Test Setup	appController instantiated Mock the messages class Do nothing when msg.FiveMinWarning(). Access popup5min variable from appController.
Test Input	popup5min.run()
Expected Output	msg.FiveMinWarning() mock was attempted to be called but was stubbed.

Test Case ID	Unit010_AppCont010 - TimerTask endofclass
Purpose	To test that the endofclass.run() method attempts to display a end of class message and changes the classEnded state variable to the current System time.
Test Setup	appController instantiated Mock the messages class. Do nothing when msg.endClassWarning() is called. Access endofclass variable from appController.
Test Input	endofclass.run();
Expected Output	msg.endClassWarning() was attempted to be called and the classEnded variable was set to the current time.

Test Case ID	Unit011 - AppCont011 - TimerTask systemExit
Purpose	To test that the systemExit.run() method attempts to run the System.exit(0) command.
Test Setup	appController instantiated PowerMock the static class System.
Test Input	systemExit.run();
Expected Output	System.exit(0) was attempted to be executed but was stubbed.

Test Case ID	Unit012 - AppCont012 - setTime(...)
--------------	-------------------------------------

Purpose	To test if the setTime() method in appController sets the Calendar setRun state variable to the specified time
Test Setup	appController instantiated int year = 2012; int month = 1; int date = 25; int hours = 10; int min = 0;
Test Input	appController.setTime(year, month, date, hours, min);
Expected Output	setRun state variable is set to the time 1/25/2012 10:45.

Test Case ID	Unit013 - AppCont013 - getTime()
Purpose	To test if the getTime() method returns the time set using setTime().
Test Setup	appController instantiated int year = 2012; int month = 1; int date = 25; int hours = 10; int min = 0; appController.setTime(year, month, date, hours, min);
Test Input	appController.getTime();
Expected Output	getTime() returns a Date object at data 1/25/2012 at time 10:00.

Test Case ID	Unit014 - AppCont014 - getTimeMillis()
--------------	--

Purpose	To test that the getTimeMillis() method returns the time in milliseconds since the default time defined in the Calendar class after the time was set with setTime().
Test Setup	appController instantiated int year = 2012; int month = 1; int date = 25; int hours = 10; int min = 45; appController.setTime(year, month, date, hours, min);
Test Input	appController.getTimeMillis();
Expected Output	getTimeMillis() should return the value set using setTime() in milliseconds.

Test Case ID	Unit015 - AppCont015 - timerParser()
Purpose	To test that the timerParser() method parses the inputted time into the respective state variables.
Test Setup	appController instantiated String time = "10:15"
Test Input	appController.timerParser(time);
Expected Output	State variables: appController.hr = 10 appController.min = 15

Test Case ID	Unit016 - AppCont016 - dateParser()
--------------	-------------------------------------

Purpose	To test that the dateParser() method parses the inputted date into the respective state variables.
Test Setup	appController instantiated String date = "12/25/19";
Test Input	appController.dateParser(date);
Expected Output	State variables: appController.clearMonth = 12 appController.clearDate = 25 appController.clearYear = 19

Test Case ID	Unit017 - AppCont017 - returnHr()
Purpose	To verify that the returnHr() method returns the correct value for the hr state variable.
Test Setup	appController instantiated String time = "12:25" appController.timerParser(time)
Test Input	appController.returnHr()
Expected Output	Method returns 12.

Test Case ID	Unit018 - AppCont018 - returnMin()
Purpose	To verify that the returnMin() method returns the correct value for the min state variable.
Test Setup	appController instantiated String time = "12:25" timerParser(time)

Test Input	appController.returnMin()
Expected Output	Method returns 25.

Test Case ID	Unit019 - AppCont019 - getEndTime()
Purpose	To test the getEndTime() method returns today's Date but 1 mins before the specified parameter's time.
Test Setup	appController instantiated int hrs = 10; int mins = 15;
Test Input	appController.getEndTime(hrs, mins);
Expected Output	The date returned by getEndTime() should be today's date at 10:14.

Test Case ID	Unit020 - AppCont020 - setSemesterClear()
Purpose	To test that the setSemesterClear() method updates the autoClear and date2 state variable to the specified inputs.
Test Setup	appController instantiated int year = 2012; int mon = 12; int date = 25; int hrs = 10; int min = 15;
Test Input	appController.setSemesterClear(year, mon, date, hrs, min)

Expected Output	State variable autoClear should be set to the inputted time, and state variable date2 should be set to the date set in the autoClear variable.
-----------------	--

Test Case ID	Unit021 - AppCont021 - getSemesterClear()
Purpose	To test that the getSemesterClear() method returns the date of the autoClear state variable.
Test Setup	appController instantiated int year = 2012; int mon = 12; int date = 25; int hrs = 10; int min = 15; appController.setSemesterClear(year, mon, date, hrs, min);
Test Input	appController.getSemesterClear()
Expected Output	getSemesterClear() returns the same date as Calendar set with the same inputs.

Test Case ID	Unit022 - AppCont022 - get15BeforeEnd()
Purpose	To test the get15BeforeEnd() method returns today's Date but 15 mins before the specified parameter's time.
Test Setup	appController instantiated int hrs = 10; int mins = 15;
Test Input	appController.get15BeforeEnd(hrs, mins);

Expected Output	The date returned by get15BeforeEnd() should be today's date at 10:00.
-----------------	--

Test Case ID	Unit023 - AppCont023 - get5BeforeEnd()
Purpose	To test the get5BeforeEnd() method returns today's Date but 5 mins before the specified parameter's time.
Test Setup	appController instantiated int hrs = 10; int mins = 15;
Test Input	appController.get5BeforeEnd(hrs, mins);
Expected Output	The date returned by get5BeforeEnd() should be today's date at 10:10.

Test Case ID	Unit024 - AppCont024 - autoExit()
Purpose	To test that the autoExit() method schedules a system exit for immediate execution.
Test Setup	appController instantiated Mock static class System. Initialize date to 1/1/2012.
Test Input	appController.autoExit()
Expected Output	System.exit(0) should be attempted to be called when the run() function is called from the systemExit TimerTask in the appController.

Test Case ID	Unit025 - AppCont025 - autoClear()
Purpose	To test that the autoClear() method schedules to clear the database for immediate execution.
Test Setup	appController instantiated Mock database and do nothing when db.clearDatabase() is called. Initialize date to 1/1/2012.
Test Input	appController.autoClear();
Expected Output	The autoClear() method attempted to call db.clearDatabase() and calls the mocked database object.

Test Case ID	Unit026 - AppCont026 - checkClear()
Purpose	To assert that checkClear() checks if the calendar is reset and returns false.
Test Setup	appController instantiated Database mocked Do nothing when db.getEndDates() is called ArrayList <String> endDates = new ArrayList<String>(); endDates.add("01/01/20");
Test Input	appController.checkClear()
Expected Output	checkClear() returns false

Test Case ID	Unit027 - AppCont027 - checkTimes()
Purpose	To test that the checkTimes() method doesn't schedule the popup for 5 min, 15 min, and the end of class since there are no courses.

Test Setup	appController instantiated ArrayList<Integer> mockCourseList; mockCourseList.add(1); Set each class end time to "".
Test Input	appController.checkTimes()
Expected Output	All popup execution times are set to 0.

Test Case ID	Unit028 - AppCont028 - getData(int course)
Purpose	To ensure that the instance variables are empty string when no data is returned from the DBConnection.
Test Setup	appController instantiated appController.db mocked Database data mocked to all return ""
Test Input	appController.getData(0)
Expected Output	State variables for course times, dates, and other information should be empty string.

Test Case ID	Unit029 - AppCont029 - sleep(int milli)
Purpose	To test that when the Thread class attempts to sleep it throws an exception.
Test Setup	appController instantiated PowerMockito.mockStatic(Thread.class); PowerMockito.when(Thread.class).thenThrow (Exception.class);
Test Input	appController.sleep(100);

Expected Output	sleep() throws an exception.
-----------------	------------------------------

Test Case ID	Unit030 - AppCont030 - LogIn()
Purpose	To verify that the appController sets the loggedIn state to false when LogIn() is called.
Test Setup	appController instantiated String user = "username"; String pass = "password"; Mockito.when(db.connect(user, pass)).thenReturn(1); Set appController private variables: username = user; password = pass;
Test Input	appController.LogIn();
Expected Output	Variable loggedIn in the appController should return false.

Test Case ID	Unit031 - AppCont031 - getTime()
Purpose	To test if the getTime() method returns the time set using setTime().
Test Setup	appController instantiated setRun state is set to 0 milliseconds time.
Test Input	appController.getTime();
Expected Output	getTime() returns a date of 0.

Test Case ID	Unit032 - AppCont032 - getTimeMillis()
Purpose	To test that the getTimeMillis() method returns the time in milliseconds from the setRun state variable.
Test Setup	appController instantiated Set the setRun calendar to 0 milliseconds time.
Test Input	appController.getTimeMillis();
Expected Output	getTimeMillis() should return 0.

Test Case ID	Unit033 - AppCont033 - timerParser()
Purpose	To test that the timerParser() method throws an exception when given an invalid value.
Test Setup	appController instantiated String time = ""
Test Input	appController.timerParser(time);
Expected Output	StringIndexOutOfBoundsException thrown.

Test Case ID	Unit034 - AppCont034 - dateParser()
Purpose	To test that the dateParser() method throws an exception when given an invalid value.
Test Setup	appController instantiated String date = "";
Test Input	appController.dateParser(date);
Expected Output	StringIndexOutOfBoundsException thrown.

Test Case ID	Unit035 - AppCont035 - returnHr()
Purpose	To test that returnHr() method returns 0 when not set to any time.
Test Setup	appController instantiated No time is parsed into state variables.
Test Input	appController.returnHr()
Expected Output	Method returns 0.

Test Case ID	Unit036 - AppCont036 - returnMin()
Purpose	To test that the returnMin() method returns 0 when no time is set.
Test Setup	appController instantiated No time is parsed into state variables.
Test Input	appController.returnMin()
Expected Output	Method returns 0.

Test Case ID	Unit037 - AppCont037 - getEndTime()
Purpose	To test that getEndTime() still returns the correct value when given a time that has it's minutes at 00.

Test Setup	appController instantiated int hrs = 10; int mins = 0;
Test Input	appController.getTime(hrs, mins);
Expected Output	When the minute is subtracted by 1 the time should subtract an hour from the hr field and min should go to 59. hr = 9; min = 59;

Test Case ID	Unit038 - AppCont038 - get15BeforeEnd()
Purpose	To test the get15BeforeEnd() method returns today's Date but 15 mins before the specified parameter's time.
Test Setup	appController instantiated int hrs = 10; int mins = 0;
Test Input	appController.get15BeforeEnd(hrs, mins);
Expected Output	The date returned by get15BeforeEnd() should be today's date at 9:45.

Test Case ID	Unit039 - AppCont039 - get5BeforeEnd()
Purpose	To test the get5BeforeEnd() method returns today's Date but 5 mins before the specified parameter's time.
Test Setup	appController instantiated int hrs = 10;

	int mins = 0;
Test Input	appController.get5BeforeEnd(hrs, mins);
Expected Output	The date returned by get5BeforeEnd() should be today's date at 9:55.

Test Case ID:	Unit041 - DBConnect001 - DBConnection.java - fetchCourseName(int courseID)
Purpose:	To verify that DBConnection.fetchCourseName(int courseID) returns the proper data state variable when it is called
Test Setup:	<p>Connection is mocked Statement is mocked DBConnection object is initialized Mocks are injected into DBConnection object</p> <pre>Mockito.when(myCon.createStatement()).thenReturn(s); Mockito.when(s.executeQuery(Mockito.anyString())).thenReturn(res); Mockito.when(s.getResultSet()).thenReturn(res); Mockito.when(res.getString(Mockito.anyString())).thenReturn("Software Testing");</pre>
Test Input:	db.fetchCourseName(4010)
Expected Output:	"Software Testing"

Test Case ID:	Unit042 - DBConnect002 - DBConnection.java - fetchCourseID(int courseID)
Purpose:	To verify that DBConnection.fetchCourseID(int courseID) returns the proper data state variable when it is called
Test Setup:	<p>Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented</p> <p>Mockito.when(myCon.createStatement()).thenReturn(s);</p> <p>Mockito.when(s.executeQuery(Mockito.anyString())).thenReturn(res);</p> <p>Mockito.when(s.getResultSet()).thenReturn(res);</p> <p>Mockito.when(res.getInt(Mockito.anyString())).thenReturn(9);</p>
Test Input:	db.fetchCourseID(907);
Expected Output:	9

NEW TEST CASES

Test Case ID	Unit043_DBConnection003 - DBConnection.Java - fetchSemester(int courseID)
Purpose	To verify that DBConnection.fetchCourseSemester(int courseID)returns the proper data state variable when it is called
Test Setup	Connection is mocked

	Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchCourseSemester(907);
Expected Output	"Fall"

Test Case ID	Unit044_DBConnection004 - DBConnection.Java - fetchCourseStart(int courseID)
Purpose	To verify that DBConnection.fetchCourseStart(int courseID) returns the proper data state variable when it is called
Test Setup	Test Setup: Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchCourseStart(907);
Expected Output	"08/24/19"

Test Case ID	Unit045_DBConnection005 - DBConnection.Java - fetchCourseSubjt(int courseID)
Purpose	To verify that DBConnection.fetchCourseSubjt(int courseID) returns the proper data state variable when it is called

Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchCourseSubj(907);
Expected Output	"Software Testing"

Test Case ID	Unit046_DBConnection006 - DBConnection.Java - fetchCourseEnd(int courseID)
Purpose	To verify that DBConnection.fetchCourseEnd(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchStartMon(907);
Expected Output	"Start Monday"

Test Case ID	Unit047_DBConnection007 - DBConnection.Java - fetchStartMon(int courseID)
Purpose	To verify that DBConnection.fetchStartMon(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented

Test Input	db.fetchEndMon(907);
Expected Output	"End Monday"

Test Case ID	Unit048_DBConnection008 - DBConnection.Java - fetchEndMon(int courseID)
Purpose	To verify that DBConnection.fetchEndMon(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchStartTue(907);
Expected Output	"Start Tuesday"

Test Case ID	Unit049_DBConnection009 - DBConnection.Java - fetchStartTue(int courseID)
Purpose	To verify that DBConnection.fetchStartTue(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchStartTue(907);
Expected Output	"Start Tuesday"

Test Case ID	Unit050_DBConnection010 - DBConnection.Java - fetchendTue(int
--------------	---

	courseID)
Purpose	To verify that DBConnection.fetchEndTue(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchEndTue(907);
Expected Output	"End Tuesday"

Test Case ID	Unit051_DBConnection011 - DBConnection.Java - fetchStartWed(int courseID)
Purpose	To verify that DBConnection.fetchStartWed(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchStartWed(907);
Expected Output	"Start Wednesday"

Test Case ID	Unit052_DBConnection012 - DBConnection.Java - fetchEndWed(int courseID)
Purpose	To verify that DBConnection.fetchStartWed(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked

	Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchEndWed(907);
Expected Output	"End Wednesday"

Test Case ID	Unit053_DBConnection013 - DBConnection.Java - fetchStartThu(int courseID)
Purpose	To verify that DBConnection.fetchStartThu(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchStartThu(907);
Expected Output	"Start Thursday"

Test Case ID	Unit054_DBConnection014 - DBConnection.Java - fetchEndThu(int courseID)
Purpose	to verify that DBConnection.fetchEndThu(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchEndThu(907);

Expected Output	"End Thursday"
-----------------	----------------

Test Case ID	Unit055_DBConnection015 - DBConnection.Java - fetchStartFri(int courseID)
Purpose	To verify that DBConnection.fetchStartFri(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchStartFri(907);
Expected Output	"Start Friday"

Test Case ID	Unit056_DBConnection016 - DBConnection.Java - fetchEndFri(int courseID)
Purpose	To verify that DBConnection.fetchEndFri(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchEndFri(907);
Expected Output	"End Friday"

Test Case ID	Unit057_DBConnection017 - DBConnection.Java - fetchStartSat(int courseID)
--------------	---

Purpose	To verify that DBConnection.fetchStartSat(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchStartSat(907);
Expected Output	"Start Saturday"

Test Case ID	Unit058_DBConnection018 - DBConnection.Java - fetchEndSat(int courseID)
Purpose	To verify that DBConnection.fetchEndSat(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.When()thenReturn mocks are properly implemented
Test Input	db.fetchEndSat(907);
Expected Output	"End Saturday"

Test Case ID	Unit059_DBConnection019 - DBConnection.Java - fetchCourseID(int courseID)
Purpose	To verify that DBConnection.fetchCourseID(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked

	DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchCourseID(907);
Expected Output	-1

Test Case ID	Unit060_DBConnection020 - DBConnection.Java - fetchCourseSubj(int courseID)
Purpose	To verify that DBConnection.fetchCourseSubj(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchCourseSubj(908);
Expected Output	null

Test Case ID	Unit061_DBConnection021 - DBConnection.Java - fetchCourseName(int courseID)
Purpose	To verify that DBConnection.fetchCourseName(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();

Test Input	db.fetchCourseName(907);
Expected Output	Null

Test Case ID	Unit062_DBCConnection022 - DBCConnection.Java - fetchCourseSemester(int courseID)
Purpose	To verify that DBCConnection.fetchCourseSemester(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBCConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchCourseSemester(907);
Expected Output	Null

Test Case ID	Unit063_DBCConnection023 - DBCConnection.Java - fetchCourseStart(int courseID)
Purpose	To verify that DBCConnection.fetchCourseStart(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBCConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchCourseStart(907);

Expected Output	Null
-----------------	------

Test Case ID	Unit064_DBConnection024 - DBConnection.Java - fetchCourseEnd(int courseID)
Purpose	To verify that DBConnection.fetchCourseEnd(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchCourseEnd(908);
Expected Output	Null

Test Case ID	Unit065_DBConnection025 - DBConnection.Java - fetchStartMon(int courseID)
Purpose	To verify that DBConnection.fetchStartMon(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchStartMon(907);
Expected Output	Null

Test Case ID	Unit066_DBConnection026 - DBConnection.Java -
--------------	---

	fetchEndMon(int courseID)
Purpose	DBConnection.fetchEndMon(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchEndMon(907);
Expected Output	Null

Test Case ID	Unit067_DBConnection027 - DBConnection.Java - fetchStartTue(int courseID)
Purpose	To verify that DBConnection.fetchStartTue(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchStartTue(907);
Expected Output	""

Test Case ID	Unit068_DBConnection028 - DBConnection.Java - fetchEndTue(int courseID)
Purpose	To verify that DBConnection.fetchEndTue(int courseID) returns the proper data state variable when it is called

Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchEndTue(907)
Expected Output	null

Test Case ID	Unit069_DBConnection029 - DBConnection.Java - fetchStartWed(int courseID)
Purpose	To verify that DBConnection.fetchStartWed(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchStartWed(907);
Expected Output	null

Test Case ID	Unit070_DBConnection030 - DBConnection.Java - fetchEndWed(int courseID)
Purpose	To verify that DBConnection.fetchEndWed(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new

	<code>SQLException()).when(myCon).createStatement();</code>
Test Input	<code>db.fetchEndWed(907);</code>
Expected Output	null

Test Case ID	Unit071_DBConnection031 - DBConnection.Java - fetchStartThu(int courseID)
Purpose	To verify that DBConnection.fetchStartThu(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	<code>db.fetchStartThu(907);</code>
Expected Output	null

Test Case ID	Unit072_DBConnection032 - DBConnection.Java - fetchEndThu(int courseID)
Purpose	To verify that DBConnection.fetchEndThu(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	<code>db.fetchEndThu(907);</code>

Expected Output	null
-----------------	------

Test Case ID	Unit073_DBConnection033 - DBConnection.Java - fetchStartFri(int courseID)
Purpose	To verify that DBConnection.fetchStartFri(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchStartFri(907);
Expected Output	null

Test Case ID	Unit074_DBConnection034 - DBConnection.Java - fetchEndFri(int courseID)
Purpose	To verify that DBConnection.fetchEndFri(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchEndFri(907);
Expected Output	null

Test Case ID	Unit075_DBConnection035 - DBConnection.Java -
--------------	---

	fetchStartSat(int courseID)
Purpose	To verify that DBConnection.fetchStartSat(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchStartSat(907);
Expected Output	null

Test Case ID	Unit076_DBConnection036 - DBConnection.Java - fetchEndSat(int courseID)
Purpose	To verify that DBConnection.fetchEndSat(int courseID) returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).createStatement();
Test Input	db.fetchEndSat(907);
Expected Output	null

Test Case ID	Unit077_DBConnection037 - DBConnection.Java - disconnect()
Purpose	To verify that DBConnection.disconnect() returns the proper data state variable when it is called

Test Setup	Test Setup: Connection is mocked Statement is mocked DBConnection object is initialized
Test Input	db.disconnect();
Expected Output	0

Test Case ID	Unit078_DBConnection038 - DBConnection.Java - disconnect()
Purpose	To verify that DBConnection.disconnect() returns the proper data state variable when it is called
Test Setup	Test Setup: Connection is mocked Statement is mocked DBConnection object is initialized Mockito.doThrow(new SQLException()).when(myCon).close();
Test Input	db.disconnect();
Expected Output	-1

Test Case ID	Unit079_DBConnection039 - DBConnection.Java - disconnect()
Purpose	To verify that DBConnection.disconnect() returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized myCon set to null
Test Input	db.disconnect();
Expected Output	-1

Test Case ID	Unit080_DBConnection040 - DBConnection.Java - clearDatabase()
Purpose	To verify that DBConnection.clearDatabase() returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized myCon set to null Mockito.when(s.execute(Mockito.anyString())).thenReturn(true); Mockito.when(myCon.createStatement()).thenReturn(s);
Test Input	db.disconnect();
Expected Output	Mockito.verify(s).execute(Mockito.anyString()); passes

Test Case ID	Unit081_DBConnection041 - DBConnection.Java - createClassTable()
Purpose	To verify that DBConnection.createClassTable() returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized myCon set to null Mockito.when(myCon.createStatement()).thenReturn(s); Mockito.when(s.executeUpdate(Mockito.anyString())).thenReturn(0);
Test Input	db.createClassTable();
Expected Output	0

Test Case ID	Unit082_DBConnection042 - DBConnection.Java - createClassTable()
Purpose	To verify that DBConnection.createClassTable() returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized myCon set to null
Test Input	db.createClassTable();
Expected Output	-1

Test Case ID	Unit083_DBConnection043 - DBConnection.Java - storeClassInfo()
Purpose	To verify that DBConnection.storeClassInfo() returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized myCon set to null Mockito.when(myCon.createStatement()).thenReturn(s); Mockito.when(s.executeUpdate(Mockito.anyString())).thenReturn(1);
Test Input	db.storeClassInfo(906,"","","");
Expected Output	0

Test Case ID	Unit084_DBConnection044 - DBConnection.Java - storeClassInfo()
--------------	--

Purpose	To verify that DBConnection.storeClassSched() returns the proper data state variable when it is calle
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized
Test Input	db.storeClassInfo(906,"","","");
Expected Output	-1

Test Case ID	Unit087_DBConnection047 - DBConnection.Java - connect()
Purpose	Unit087_DBConnection047 - DBConnection.Java - connect() Purpose: To verify that DBConnection.createClassTable()
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized
Test Input	db.connect("","");
Expected Output	-1

Test Case ID	Unit088_DBConnection048 - DBConnection.Java - connect()
Purpose	To verify that DBConnection.createClassTable() returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized
Test Input	db.connect("","");
Expected Output	-1

Test Case ID	Unit089_DBConnection049 - DBConnection.Java - getEndDates()
Purpose	To verify that DBConnection.getEndDates() returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.when(myCon.createStatement()).thenReturn(s); Mockito.when(s.executeQuery(Mockito.anyString())).thenReturn(res); Mockito.when(s.getResultSet()).thenReturn(res); Mockito.when(res.next()).thenReturn(true,false); Mockito.when(res.getString(Mockito.anyString())).thenReturn(""); testArray initialized as string arrayList "" stored in testArray
Test Input	db.getEndDates();
Expected Output	array containing only ""

Test Case ID	Unit090_DBConnection050 - DBConnection.Java - getEndDates()
Purpose	To verify that DBConnection.getEndDates() returns the proper data state variable when it is called
Test Setup	Connection is mocked Statement is mocked DBConnection object is initialized Mockito.when(myCon.createStatement()).thenReturn(s); Mockito.when(s.executeQuery(Mockito.anyString())).thenReturn(res); Mockito.when(s.getResultSet()).thenReturn(res); Mockito.when(res.next()).thenReturn(true,false); Mockito.when(res.getInt(Mockito.anyString())).thenReturn(1); testArray initialized as integer arrayList

	stored in testArray
Test Input	db.getCourse();
Expected Output	array containing only 1

Test Case ID	Unit091_DBConnection051 - DBConnection.Java - fetchCourses()
Purpose	To verify that DBConnection.fetchCourses() returns the proper data state variable when it is called
Test Setup	<p>Connection is mocked Statement is mocked DBConnection object is initialized Mockito.when(myCon.createStatement()).thenReturn(s); Mockito.when(s.executeQuery(Mockito.anyString())).thenReturn(res); Mockito.when(s.getResultSet()).thenReturn(res); Mockito.when(res.next()).thenReturn(true,false); Mockito.when(res.Int(Mockito.anyString())).thenReturn(1); String result initialized to "1, "</p>
Test Input	db.getEndDates();
Expected Output	String containing "1, "

Test Case ID	Unit089_AppCont40 - main()
Purpose	To test the main() method to ensure it executes from the login to logout.
Test Setup	<p>appController instantiated. All incoming class dependencies from appController mocked.</p> <p>InterfaceController ic = Mockito.mock(InterfaceController.class); LoginForm log = Mockito.mock(LoginForm.class);</p>

	<pre> Messages msg = Mockito.mock(Messages.class); DBConnection dbMain = Mockito.mock(DBConnection.class); MainMenu mm = Mockito.mock(MainMenu.class); Authenticate auth = Mockito.mock(Authenticate.class); PowerMockito.mockStatic(Thread.class); PrefilledScheduleForm psf = Mockito.mock(PrefilledScheduleForm.class); All dependent object method calls are mocked. </pre>
Test Input	<code>appController.main(null);</code>
Expected Output	<p>Verify that all mocked dependencies to methods were called, and the branches for <code>logoutSelected</code>, <code>editSchedSelected</code>, and <code>InitSetupSelected</code> were taken.</p> <p>Verify that mocked dependencies were called during the process:</p> <ul style="list-style-type: none"> Verify <code>InterfaceController</code> mocks were called. Verify <code>LoginForm</code> mocks were called. Verify <code>Messages</code> mocks called. Verify <code>courseSelect</code> mocks were called. Verify <code>MainMenu</code> mocks were called. Verify <code>PrefilledScheduleForm</code> mocks were called. Verify <code>Authenticate</code> mocks were called. Verify <code>Thread</code> mocks were called. Verify <code>Authentication</code> mocks were called.

Test Case ID	Unit090_AppCont041 - sleep()
Purpose	To test that <code>sleep()</code> catches an interrupted thread exception.
Test Setup	<pre> appController instantiated Thread.currentThread().interrupt(); PowerMockito.mockStatic(Logger.class); </pre>
Test Input	<code>appController.sleep(100);</code>

Expected Output	Verify that the Logger mock was called in the catch block because the current thread was interrupted.
-----------------	---

Test Case ID	Unit091_AppCont042 - checkTimes()
Purpose	To test that autoExit() method attempts to system exit after scheduling an exit.
Test Setup	appController instantiated Timer t = Mockito.mock(Timer.class) Mockito.doNothing().when(t).schedule(Mockito.any(), Mockito.any());
Test Input	appController.autoExit()
Expected Output	Verify that PSM attempts to system exit by verifying that Timer mock was called instead.

Test Case ID	Unit092_AppCont043 - checkTimes()
Purpose	To test that autoClear() method attempts to schedule to clear the database after the designated time.
Test Setup	appController instantiated Timer t = Mockito.mock(Timer.class) appController.timer = t; Mockito.doNothing().when(t).schedule(Mockito.any(), Mockito.any());
Test Input	appController.autoClear()
Expected Output	Verify PSM attempts to clear the database but the Timer mock was called instead.

Test Case ID	Unit093_AppCont044 - checkTimes()
Purpose	To test that checkTimes() parses the time for Monday correctly when the date is Monday and there is a class for Monday.
Test Setup	appController instantiated ArrayList<Integer> mockCourseList; mockCourseList.add(1); Set the Monday end time to 1:15. Set each other class end time to "". Mock the GregorianCalendar class and set the day to Monday.
Test Input	appController.checkTimes()
Expected Output	The time returned from appController should be 1:15. hr = 1 min = 15

Test Case ID	Unit094_AppCont045 - checkTimes()
Purpose	To test that checkTimes() parses the time for Tuesday correctly when the date is Tuesday and there is a class for Tuesday.
Test Setup	appController instantiated ArrayList<Integer> mockCourseList; mockCourseList.add(1); Set the Tuesday end time to 1:15. Set each other class end time to "". Mock the GregorianCalendar class and set the day to Tuesday.
Test Input	appController.checkTimes()
Expected Output	The time returned from appController should be 1:15. hr = 1

	min = 15
--	----------

Test Case ID	Unit095_AppCont046 - checkTimes()
Purpose	To test that checkTimes() parses the time for Wednesday correctly when the date is Wednesday and there is a class for Wednesday.
Test Setup	appController instantiated ArrayList<Integer> mockCourseList; mockCourseList.add(1); Set the Wednesday end time to 1:15. Set each other class end time to "". Mock the GregorianCalendar class and set the day to Wednesday.
Test Input	appController.checkTimes()
Expected Output	The time returned from appController should be 1:15. hr = 1 min = 15

Test Case ID	Unit096_AppCont047- checkTimes()
Purpose	To test that checkTimes() parses the time for Thursday correctly when the date is Thursday and there is a class for Thursday.
Test Setup	appController instantiated ArrayList<Integer> mockCourseList; mockCourseList.add(1); Set the Thursday end time to 1:15. Set each other class end time to "". Mock the GregorianCalendar class and set the day to Thursday.
Test Input	appController.checkTimes()
Expected Output	The time returned from appController should be 1:15.

	hr = 1 min = 15
--	--------------------

Test Case ID	Unit097_AppCont048 - checkTimes()
Purpose	To test that checkTimes() parses the time for Friday correctly when the date is Friday and there is a class for Friday.
Test Setup	appController instantiated ArrayList<Integer> mockCourseList; mockCourseList.add(1); Set the Friday end time to 1:15. Set each other class end time to "". Mock the GregorianCalendar class and set the day to Friday.
Test Input	appController.checkTimes()
Expected Output	The time returned from appController should be 1:15. hr = 1 min = 15

Test Case ID	Unit098_AppCont049 - checkTimes()
Purpose	To test that checkTimes() parses the time for Saturday correctly when the date is Saturday and there is a class for Saturday.
Test Setup	appController instantiated ArrayList<Integer> mockCourseList; mockCourseList.add(1); Set the Saturday end time to 1:15. Set each other class end time to "". Mock the GregorianCalendar class and set the day to Saturday.
Test Input	appController.checkTimes()

Expected Output	The time returned from appController should be 1:15. hr = 1 min = 15
-----------------	--

Test Case ID	Unit099_AppCont050- main()
Purpose	To test the main() method to ensure it visits the branches with incorrect login, password lock, and then logout.
Test Setup	appController instantiated. All incoming classes from appController mocked. InterfaceController ic = Mockito.mock(InterfaceController.class); LoginForm log = Mockito.mock(LoginForm.class); Messages msg = Mockito.mock(Messages.class); DBConnection dbMain = Mockito.mock(DBConnection.class); MainMenu mm = Mockito.mock(MainMenu.class); Authenticate auth = Mockito.mock(Authenticate.class); PowerMockito.mockStatic(Thread.class);
Test Input	appController.main(null);
Expected Output	Verify that mocked dependencies were called during the process: Verify InterfaceController mocks were called. Verify LoginForm mocks were called. Verify Messages mocks called. Verify courseSelect mocks were called. Verify MainMenu mocks were called. Verify Authenticate mocks were called. Verify Thread mocks were called.

3.2. Actual Test Results

The test results below will display the test results for each class and whether they resulted in a Pass or Failure. Directly after that the coverage percentage for branch and statement coverage will be displayed for each tested class and for each coverage tool that was used.

Test Results

Test ID	Pass/Fail/Error
AppCont001	Pass
AppCont002	Fail
AppCont003	Pass
AppCont004	Pass
AppCont005	Pass
AppCont006	Pass
AppCont007	Pass
AppCont008	Error
AppCont009	Error
AppCont010	Error
AppCont011	Error
AppCont012	Pass

AppCont013	Pass
AppCont014	Pass
AppCont015	Pass
AppCont016	Pass
AppCont017	Pass
AppCont018	Pass
AppCont019	Pass
AppCont020	Pass
AppCont021	Pass
AppCont022	Pass
AppCont023	Pass
AppCont024	Error
AppCont025	Error
AppCont026	Pass
AppCont027	Pass
AppCont028	Pass
AppCont029	Fail
AppCont030	Pass
AppCont031	Pass
AppCont032	Pass

AppCont033	Pass
AppCont034	Pass
AppCont035	Pass
AppCont036	Pass
AppCont037	Pass
AppCont038	Pass
AppCont039	Pass
AppCont040	Pass
AppCont041	Pass
AppCont042	Pass
AppCont043	Pass
AppCont044	Pass
AppCont045	Pass
AppCont046	Pass
AppCont047	Pass
AppCont048	Pass
AppCont049	Pass
AppCont050	Pass
DBConnection001	Pass
DBConnection002	Pass

DBConnection003	Pass
DBConnection004	Pass
DBConnection005	Pass
DBConnection006	Pass
DBConnection007	Pass
DBConnection008	Pass
DBConnection009	Pass
DBConnection010	Pass
DBConnection011	Pass
DBConnection012	Pass
DBConnection013	Pass
DBConnection014	Pass
DBConnection015	Pass
DBConnection016	Pass
DBConnection017	Pass
DBConnection018	Pass
DBConnection019	Pass
DBConnection020	Pass
DBConnection021	Pass
DBConnection022	Pass

DBConnection023	Pass
DBConnection024	Pass
DBConnection025	Pass
DBConnection026	Pass
DBConnection027	Pass
DBConnection028	Pass
DBConnection029	Pass
DBConnection030	Pass
DBConnection031	Pass
DBConnection032	Pass
DBConnection033	Pass
DBConnection034	Pass
DBConnection035	Pass
DBConnection036	Pass
DBConnection037	Pass
DBConnection038	Pass
DBConnection039	Pass
DBConnection040	Pass
DBConnection041	Pass
DBConnection042	Pass

DBConnection043	Pass
DBConnection044	Pass
DBConnection045	Pass
DBConnection046	Pass
DBConnection047	Pass
DBConnection048	Pass
DBConnection049	Pass
DBConnection050	Pass
DBConnection051	Pass

Original Coverage

Coverage Type:	Statement Coverage			Branch Coverage		
Coverage Tool:	EclEmma	Clover	CodeCover	EclEmma	Clover	CodeCover
appController	54.2%	48.2%	X	X	26.8%	X
DBConnection	7.5%	6.7%	6.7%	0%	0%	3.4%

Updated Coverage

Coverage Type:	Statement Coverage			Branch Coverage		
Coverage Tool:	EclEmma	Clover	CodeCover	EclEmma	Clover	CodeCover
appController	97.8%	95.8	X	X	94.6%	X
DBConnection	93.6%	97.3%	95.9%	100%	100%	89.7%

CHAPTER 4: Subsystem Testing

Subsystem testing focuses on testing all the services provided to the user, in this case the Logic package. For Subsystem testing of the Professor Schedule Manager, we instantiated the ApplicationController class in the PSM_Logic package and then test all the calls that were made to and from this one.

4.1. Subsystem Test Cases

The following section outlines all of the test cases, this does not include the first set for specification based testing since all the test that were created on the first part of the project were wrong and there was the need to create new test cases that would work properly.

4.1.1. Test Identification and Objective

Test ID	SubSys01 - LoginForm
Purpose	Verify that when the method InitLoginForm() is called this one initiates a new login form.

Test ID	SubSys02 - DataReceived
Purpose	Verify that when the method DataReceived() is called this one would return a boolean value stating if data was received or not.

Test ID	SubSys03 - setData
---------	--------------------

Purpose	Verify that when the method SetDataRecRunner() is called, this one will set a variable dataRec in my.PSM.PSM_Interface.LoginForm.java to false
---------	--

Test ID	SubSys04 - GetUser
Purpose	Verify that when the method getUsernameRunner() is called this one will return the username entered to login.

Test ID	SubSys05 - getPassword
Purpose	Verify that when the method getPasswordRunner() is called this one will return the password entered to login.

Test ID	SubSys06 - ValidateTRUE
Purpose	Verify that when the method AuthRunner(username, password) is called this one will validate that the username and password are valid and will set loggedin variable to TRUE.

Test ID	SubSys07 - ValidateFALSE
Purpose	Verify that when the method AuthRunner(username, password) is called this one will validate that the username and password are not valid and will set loggedin variable to FALSE.

Test ID	SubSys08 - dbConnection
Purpose	Verify that when the method makeAuthenticateInstRunner(username, password) is called this one will create a new instance of dbConnection.

Test ID	SubSys09 - validateState
Purpose	Verify that when the method ValidationRunner() is called this one will return true if the state of the login is working.

Test ID	SubSys10 - MessageAckFlase
Purpose	Verify that when the method MsgAckRunner() is called this one will return false for the display of message.

Test ID	SubSys11 - MessageAckTrue
Purpose	Verify that when the method MsgAckRunner() is called this one will return true for the display of message.

Test ID	SubSys12 - PasswordRetryQuit
Purpose	Verify that when the method PasswordRetryRunner(false, 3) with attempt to login with password is called this one will verify if a password was used 3 times, if it was incorrect the program would quit.

Test ID	SubSys13 - passwordRetryQuitNot
Purpose	Verify that when the method PasswordRetryRunner(false, 2) is called this one will this one would not start the steps to quit the program.

Test ID	SubSys14 - DataReceived
Purpose	Verify that when the method DataRecRunner() is called this one will return true.

Test ID	SubSys15 - EditSched
Purpose	Verify that when the method EditSchedSelectedRunner() is called this one will return true for editing the schedule.

Test ID	SubSys16 - InitSetupSelected
---------	------------------------------

Purpose	Verify that when the method InitSetupSelectedRunner() is called this one will return true for the setup of the selected schedule.
---------	---

Test ID	SubSys17 - LogoutSelected
Purpose	Verify that when the method LogoutSelectedRunner() is called this one will return true.

Test ID	SubSys18 - ClassEnded
Purpose	Verify that when the method classEndedConditionRunner() is called with 2 long variables, this one will compare it and it will terminate the program if the both conditions of the if statement are met.

Test ID	SubSys19 - ClassEnded2
Purpose	Verify that when the method classEndedConditionRunner() is called with 2 long variables, this one will compare it and it will terminate the program if the both conditions of the if statement are met.

Test ID	SubSys20 - ClassEnded3
Purpose	Verify that when the method classEndedConditionRunner() is called with 2 long variables, this one will compare it and it will terminate the program if the both conditions of the if statement are met.

Test ID	SubSys21 - dataNotRecieved
Purpose	Verify that when the method whileDataNotReceivedRunner(false) is called this one will instantiate courseSelectObject and try again

Test ID	SubSys22 - courseSelect
---------	-------------------------

Purpose	Verify that when the method CourseSelectedRunner() is called this one will create an instantiation of the object courseSelect.
---------	--

Test ID	SubSys23 - courseSelectRunner
Purpose	Verify that when the method whileForCourseSelectedRunner() is called check that the method app1.CourseSelectedRunner() is called at least once.

Test ID	SubSys24 - CourseSelectedVarFalse
Purpose	Verify that when the method setCourseSelectedRunner() is called this one would set the variable courseSelected to false.

Test ID	SubSys25 - preFilledFormRunner
Purpose	Verify that the method preFilledFormRunner is called.

Test ID	SubSys26 - checkClearTrue
Purpose	Verify that when the method checkClear() is called this one will return true.

Test ID	SubSys27 - AutoExit()
Purpose	Verify that when the method autoExit() is called this one will auto exit the program.

Test ID	SubSys28 - autoClear
Purpose	Verify that when the method autoClear() is called this one will auto clear the schedule.

Test ID	SubSys29 - get5Before
---------	-----------------------

Purpose	Verify that when the method get5BeforeEnd() is called this one will return a 5 minute warning about the program closing.
---------	--

Test ID	SubSys30 - storeClassSched
Purpose	Verify that when the method storeClassSched() is called this one store class schedule in the database.

Test ID	SubSys31 - StoreClassSched2
Purpose	Verify that when the method storeClassSched() is called this one does not store class schedule in the database.

Test ID	SubSys32 - logIn
Purpose	Verify that when the method LogIn() is called this one return 0 if the login information connects to the database.

Test ID	SubSys33 - logIn2
Purpose	Verify that when the method MsgAckRunner() is called this one will false for the display of message

Test ID	SubSys34 - getData
Purpose	Verify that when the method getData() is called with an integer, this one would return the data assign to the course number 5 in the database.

Test ID	SubSys35 - fetchCoursesEmpty
Purpose	To test that the call to DBConnection.fetchCourses()when called would return an empty string.

Test ID	SubSys36 - fetchCoursesString
Purpose	To test that the call to DBConnection.fetchCourses()when called would return a string.

4.1.2. Test Criteria and Procedures

The focus of the tests will be to improve code coverage and generate new test cases based on control flow or data flow of the code. Note that the procedures include the type of code coverage expected and the percentage of coverage.

The test criteria for these test cases required the usage of JUnit, Mockito, and PowerMockito. These tools allowed us to easily implement a test suite to run a test for each test case. Before all tests, the appController was initialized and InterfaceController was mocked when this one was called by the appController.

For testing purposes, the appController class was edited in a minor way to make the code testable. As well as some of the return statements in different classes such as Interface.LoginForm. Also there was a need to make the InteraceController class public, this way the we gained access to all of the calls from the instantiated appController into the InterfaceCONtroller .The InterfaceController was also made public in order to mock the instantiation of that class.

An attempt to reach 100% branch and statement coverage was made . For Subsytem

testing, if more test cases were created, it was expected that branch and statement coverage would reach a high percentage.

The following are certain procedures that were taken when creating test cases against these unit methods:

- Every time there was a call made to a method and this one was supposed to return a type, this method would be mocked, that way we would always be testing the function of the call.
- In order to test the fetchCourses() call, first the program is run to find out what this call would return from the database and then mock it so it would return the same information.
- When a method call was supposed to change the value of a private variable in a different class the tool Whitebox from PowerMockito was used in order to get the value of said variable in that state.
- Assertions were either made against the expected behavior, or if all functionality needed to be mocked then the method calls to those respective mocks were verified.
- For coverage, multiple test cases were made to cover multiple branches of methods in order to obtain a high branch coverage. If the method had a loop that iterated multiple times, then this was opportunity was taken to cover all the branches in that loop.

4.1.3. Test Cases

Test Case ID	SubSys01 - LoginForm
Purpose	Verify that when the method InitLoginForm() is called this one initiates a new login form.

Test Setup	appController instantiated assertFalse when boolean value is return
Test Input	app1.DataReceived();
Expected Output	Call to the method should return false

Test Case ID	SubSys02 - DataReceived
Purpose	Verify that when the method DataReceived() is called this one would return a boolean value stating if data was received or not.
Test Setup	appController instantiated assertFalse when boolean value is return
Test Input	app1.DataReceived();
Expected Output	Call to the method should return false

Test Case ID	SubSys03 - setData
Purpose	Verify that when the method SetDataRecRunner() is called, this one will set a variable dataRec in my.PSM.PSM_Interface.LoginForm.java to false.
Test Setup	appController instantiated boolean bool = (boolean) Whitebox.getInternalState(ic.log, "dataRec"); assertFalse(bool);

Test Input	app1.SetDataRecRunner();
Expected Output	Call to the method should set the value of dataRec to false.

Test Case ID	SubSys04 - GetUser
Purpose	Verify that when the method getUsernameRunner() is called this one will return the username entered to login.
Test Setup	appController instantiated String user; assertEquals("root", user);
Test Input	user = app1.getUsername();
Expected Output	Call to the method should set the username "root".

Test Case ID	SubSys05 - getPassword
Purpose	Verify that when the method getPasswordRunner() is called this one will return the password entered to login.
Test Setup	appController instantiated String pass; assertEquals("12345", pass);
Test Input	pass = app1.getPasswordRunner();
Expected Output	Call to the method should set the password "12345".

Test Case ID	SubSys06 - ValidateTRUE
Purpose	Verify that when the method AuthRunner(username, password) is called this one will validate that the username and password are valid and will set loggedin variable to TRUE.

Test Setup	appController instantiated Mockito.doReturn(auth).when(app1).makeAuthenticateInstRunner("root","password"); Mockito.doReturn(true).when(app1).ValidationRunner();
Test Input	app1.AuthRunner("root", "password");
Expected Output	Call to the method should set the variable loggedin to true.

Test Case ID	SubSys07 - ValidateFALSE
Purpose	Verify that when the method AuthRunner(username, password)is called this one will validate that the username and password are not valid and will set loggedin variable to FALSE.
Test Setup	appController instantiated Mockito.doReturn(auth).when(app1).makeAuthenticateInstRunner("root","password"); Mockito.doReturn(false).when(app1).ValidationRunner();
Test Input	app1.AuthRunner("root", "password");
Expected Output	Call to the method should set the variable loggedin to false.

Test Case ID	SubSys08 - dbConnection
Purpose	Verify that when the method makeAuthenticateInstRunner(username, password) is called this one will create a new instance of dbConnection.
Test Setup	appController instantiated String username = "root"; String password = "12345";

Test Input	<code>app1.AuthRunner("root", "password");</code>
Expected Output	Call to the method should create a new instance of dbConnection object.

Test Case ID	SubSys09 - validateState
Purpose	Verify that when the method ValidationRunner() is called this one will return true if the state of the login is working.
Test Setup	appController instantiated Mockito.doReturn(true).when(app1).ValidationRunner(); boolean bool; assertTrue(bool);
Test Input	<code>bool = app1.ValidationRunner();</code>
Expected Output	Call to the method should return true

Test Case ID	SubSys10 - MessageAckFlase
Purpose	Verify that when the method MsgAckRunner() is called this one will return false for the display of message.
Test Setup	appController instantiated boolean bool; assertFalse(bool);
Test Input	<code>bool = app1.MsgAckRunner();</code>

Expected Output	Call to the method should return false
-----------------	--

Test Case ID	SubSys11 - MessageAckSet
Purpose	Verify that when the method MsgAckRunner() is called this one will return true for the display of message.
Test Setup	appController instantiated boolean bool; assertTrue(bool);
Test Input	bool = app1.MsgAckRunner();
Expected Output	Call to the method should return true

Test Case ID	SubSys12 - PasswordRetryQuit
Purpose	Verify that when the method PasswordRetryRunner(false, 3) with attempt to login with password is called this one will verify if a password was used 3 times, if it was incorrect the program would quit.
Test Setup	appController instantiated Mockito.when(app1.ExitRunner()).thenReturn("System Exiting"); Mockito.doNothing().when(app1).PasswordLockRunner(); Mockito.doNothing().when(app1).DataReceivedAck(false); assertEquals("System Exiting", app1.ExitRunner());
Test Input	app1.PasswordRetryRunner(false, 3);

Expected Output	Call to the method should quit the program
-----------------	--

Test Case ID	SubSys13 - passwordRetryQuitNot
Purpose	Verify that when the method PasswordRetryRunner(false, 2) is called this one will this one would not start the steps to quit the program.
Test Setup	appController instantiated VerificationMode mode = Mockito.never(); Mockito.verify(app1, mode).PasswordLockRunner(); Mockito.verify(app1, mode).DataReceivedAck(false); Mockito.verify(app1, mode).ExitRunner();
Test Input	app1.PasswordRetryRunner(false, 2)
Expected Output	Call to the method should set the quitting steps to never take action

Test Case ID	SubSys14 - DataReceived
Purpose	Verify that when the method DataRecRunner() is called this one will return true.
Test Setup	appController instantiated boolean bool; assertTrue(bool);
Test Input	bool = app1.DataRecRunner();
Expected Output	Call to the method should return true

Test Case ID	SubSys15 - EditSched
--------------	----------------------

Purpose	Verify that when the method EditSchedSelectedRunner() is called this one will return true for editing the schedule.
Test Setup	appController instantiated boolean bool; assertTrue(bool);
Test Input	bool = app1.EditSchedSelectedRunner();
Expected Output	Call to the method should return true

Test Case ID	SubSys16 - InitSetupSelected
Purpose	Verify that when the method InitSetupSelectedRunner() is called this one will return true for the setup of the selected schedule.
Test Setup	CappController instantiated boolean bool; assertTrue(bool);
Test Input	bool = app1.InitSetupSelectedRunner();
Expected Output	Call to the method should return true

Test Case ID	SubSys17 - LogoutSelected
Purpose	Verify that when the method LogoutSelectedRunner() is called this one will return true.
Test Setup	appController instantiated boolean bool; assertTrue(bool);
Test Input	bool = app1.LogoutSelectedRunner();
Expected Output	Call to the method should return true

Test Case ID	SubSys18 - ClassEnded
Purpose	Verify that when the method classEndedConditionRunner() is called with 2 long variables, this one will compare it and it will terminate the program if the both conditions of the if statement are met.
Test Setup	appController instantiated
Test Input	app1.classEndedConditionRunner(0, 700000000);
Expected Output	call should branch through the if statement and quit the program

Test Case ID	SubSys19 - ClassEnded2
Purpose	Verify that when the method classEndedConditionRunner() is called with 2 long variables, this one will compare it and it will terminate the program if the both conditions of the if statement are met.
Test Setup	appController instantiated
Test Input	app1.classEndedConditionRunner(1, 500000);
Expected Output	call should branch through the if statement and quit the program

Test Case ID	SubSys20 - ClassEnded3
Purpose	Verify that when the method classEndedConditionRunner() is called with 2 long variables, this one will compare it and it will terminate the program if the both conditions of the if statement are met.
Test Setup	appController instantiated
Test Input	classEndedConditionRunner(0, 500000);
Expected Output	call should branch through the if statement and quit the program

Test Case ID	SubSys21 - dataNotRecieved
Purpose	Verify that when the method whileDataNotReceivedRunner(false) is called this one will instantiate courseSelectObject and try again
Test Setup	appController instantiated Mockito.doReturn(true).when(app1).DataRecRunner(); Mockito.doReturn(true).when(app1).EditSchedSelectedRunner(); Mockito.doReturn(false).when(app1).InitSetupSelectedRunner(); Mockito.doReturn(false).when(app1).LogoutSelectedRunner();
Test Input	app1.whileDataNotReceivedRunner(false);
Expected Output	Call to the method should check if it received data if not instantiate the courseSelect object and try again.

Test Case ID	SubSys22 - courseSelect
Purpose	Verify that when the method CourseSelectedRunner() is called this one will create an instantiation of the object courseSelect.
Test Setup	appController instantiated Mockito.verify(app1).CourseSelectedRunner();
Test Input	app1.CourseSelectedRunner();
Expected Output	Call to the method should create a new instantiation of the object courseSelect

Test Case ID	SubSys23 - courseSelectRunner
Purpose	Verify that when the method whileForCourseSelectedRunner() is called check that the method app1.CourseSelectedRunner() is called at least once.

Test Setup	appController instantiated Mockito.doReturn(true).when(app1).CourseSelectedRunner();
Test Input	app1.whileForCourseSelectedRunner();
Expected Output	Call to the method should return true and app1.CourseSelectedRunner() should be called at least once

Test Case ID	SubSys24 - CourseSelectedVarFalse
Purpose	Verify that when the method setCourseSelectedRunner() is called this one would set the variable courseSelected to false.
Test Setup	appController instantiated boolean bool = (boolean) Whitebox.getState(ic.cs, "courseSelected"); assertFalse(bool);
Test Input	app1.setCourseSelectedRunner(false);
Expected Output	Call to the method should set courseSelected to false

Test Case ID	SubSys25 - preFilledFormRunner
Purpose	Verify that the method preFilledFormRunner is called.
Test Setup	appController instantiated
Test Input	app1.preFilledFormRunner();
Expected Output	Call to the method should prefill the schedule form with values = null.

Test Case ID	SubSys26 - checkClearTrue
Purpose	Verify that when the method checkClear() is called this one will return true.

Test Setup	appController instantiated boolean bool; assertTrue(bool);
Test Input	app1.checkClear();
Expected Output	Call to the method should return true

Test Case ID	SubSys27 - AutoExit()
Purpose	Verify that when the method autoExit() is called this one will auto exit the program.
Test Setup	appController instantiated
Test Input	app1.autoExit();
Expected Output	Call to the method should exit the program

Test Case ID	SubSys28 - autoClear
Purpose	Verify that when the method autoClear() is called this one will auto clear the schedule.
Test Setup	appController instantiated
Test Input	app1.autoClear();
Expected Output	Call to the method should clear the schedule pre filled

Test Case ID	SubSys29 - get5Before
Purpose	Verify that when the method get5BeforeEnd() is called this one will return a 5 minute warning about the program closing.
Test Setup	appController instantiated
Test Input	app1.get5BeforeEnd(5, 5);

Expected Output	Call to the method should return date
-----------------	---------------------------------------

Test Case ID	SubSys30 - storeClassSched
Purpose	Verify that when the method storeClassSched() is called this one store class schedule in the database.
Test Setup	appController instantiated Mockito.doReturn(0).when(db).storeClassSched(4072, "CEN", "04/25/2019", "6:30PM", "7:40PM", "N/A", "N/A", "6:30PM", "7:40PM", "N/A", "N/A", "N/A", "N/A", "N/A", "N/A"); int test;
Test Input	test = db.storeClassSched(4072, "CEN", "04/25/2019", "6:30PM", "7:40PM", "N/A", "N/A", "6:30PM", "7:40PM", "N/A", "N/A", "N/A", "N/A", "N/A", "N/A");
Expected Output	Call to the method should return 0 if it store the schedule in the database properly.

Test Case ID	SubSys31 - StoreClassSched2
Purpose	Verify that when the method storeClassSched() is called this one does not store class schedule in the database.
Test Setup	appController instantiated Mockito.doReturn(-1).when(db).storeClassSched(4072, "CEN", "04/25/2019", "6:30PM", "7:40PM", "N/A", "N/A", "6:30PM", "7:40PM", "N/A", "N/A", "N/A", "N/A", "N/A", "N/A");
Test Input	app1.storeClassSchedRunner();
Expected Output	Call to the method should not store the schedule and return -1

Test Case ID	SubSys32 - logIn
--------------	------------------

Purpose	Verify that when the method LogIn() is called this one return 0 if the login information connects to the database.
Test Setup	appController instantiated Mockito.doReturn(0).when(db).connect("root", "12345");
Test Input	app1.LogIn();
Expected Output	Call to the method should return 0

Test Case ID	SubSys33 - logIn2
Purpose	Verify that when the method MsgAckRunner() is called this one will false for the display of message
Test Setup	appController instantiated Mockito.doReturn(-1).when(db).connect("root", "12345");
Test Input	app1.LogIn();
Expected Output	Call to the method should return -1

Test Case ID	SubSys34 - getData
Purpose	Verify that when the method getData() is called with an integer, this one would return the data assign to the course number 5 in the database.
Test Setup	appController instantiated
Test Input	app1.getData(5);
Expected Output	Call to the method should return data from the database

Test Case ID	SubSys35 - fetchCoursesEmpty
Purpose	To test that the call to DBConnection.fetchCourses()when called would return an empty string.

Test Setup	Mockito.when(db.fetchCourses()).thenReturn("");
Test Input	String ans = app.db.fetchCourses();
Expected Output	Output: ans = "";

Test Case ID	SubSys36 - fetchCoursesString
Purpose	To test that the call to DBConnection.fetchCourses()when called would return a string.
Test Setup	Mockito.when(db.fetchCourses()).thenReturn("CEN4072,COP3337,CDA4101,STA4106,COP2401");
Test Input	String ans = app.db.fetchCourses();
Expected Output	ans = "CEN4072,COP3337,CDA4101,STA4106,COP2401";

Old test cases are not included in this section because the team made a mistake on the first part of the project and all the test cases created were wrong. We did include a chart and pictures of the coverage obtained on the first part.

4.2. Test Results

Test ID	Pass/Fail/Error
SubSys01	Pass
SubSys02	Pass

SubSys03	Pass
SubSys04	Pass
SubSys05	Pass
SubSys06	Pass
SubSys07	Pass
SubSys08	Pass
SubSys09	Pass
SubSys10	Pass
SubSys11	Pass
SubSys12	Pass
SubSys13	Pass
SubSys14	Pass
SubSys15	Pass
SubSys16	Pass
SubSys17	Pass
SubSys18	Pass
SubSys19	Pass
SubSys20	Pass
SubSys21	Pass
SubSys22	Error

SubSys23	Pass
SubSys24	Error
SubSys25	Pass
SubSys26	Pass
SubSys27	Error
SubSys28	Pass
SubSys29	Pass
SubSys30	Error
SubSys31	Fail
SubSys32	Fail
SubSys33	Fail
SubSys34	Pass
SubSys35	Error
SubSys36	Pass

Original Coverage

Coverage Type:	Statement Coverage	Branch Coverage
----------------	--------------------	-----------------

Coverage Tool:	EclEmma	Clover	EclEmma	Clover
Logic Package	16.2%	0.5%	0.0%	0.0%

Updated Coverage

Coverage Type:	Statement Coverage		Branch Coverage	
Coverage Tool:	EclEmma	Clover	EclEmma	Clover
Logic Package	42.9%	47.6%	20.0%	21.2%

CHAPTER 5: System Testing

System testing focuses on more than the features outlined in the documentation. For system testing, we tested the entire system using Rational Functional Tester (RFT) and apache ANT.

5.1. System Test Cases

The following section outlines all of the test cases we conducted, beginning with the first set for specification based testing and then the new test cases added as part of implementation based testing. We had only a couple test cases for the specification based testing part of the class and added 32 new test cases for the implementation based testing segment.

5.1.1. Test Identification and Objective

(Summary) – unique identifier, purpose of each test.

Test ID	Security 001 PSM_Security_Sunny001
Purpose	To test whether you would get a “Too Many Tries” pop up with three incorrect password/username inputs

Test ID	Security 002 PSM_Security_Sunny002
Purpose	To test whether you would get a “Too Many Tries” pop up with an additional three incorrect password/username inputs

Test ID	Security 003 PSM_Security_Sunny003
Purpose	To test whether you would get a “Too Many Tries” pop up with a third set of three incorrect password/username inputs

The following test cases were added:

Test ID	System 004 PSM_Clear_Sunny001
Purpose	To test whether you can clear schedule immediately after logging in

Test ID	System 005 PSM_Clear_Sunny002
Purpose	To test whether you can clear schedule on the last day of semester

Test ID	System 006 PSM_Clear_Sunny003
Purpose	To test whether you can clear a schedule immediately after adding schedule

Test ID	System 007 PSM_Clear_Rainy001
Purpose	To test whether you can clear a schedule immediately after adding a schedule

Test ID	System 008 PSM_Clear_Rainy002
Purpose	To test whether you can clear a schedule immediately after selecting advanced features (advanced features was not implemented)

Test ID	System 009 PSM_Clear_Rainy003
Purpose	To test whether you can clear a schedule immediately after adding an alarm (alarm feature was not implemented)

Test ID	System 010 PSM_Edit_Schedule002
Purpose	To test whether you can edit an already existing schedule

Test ID	System 011 PSM_Edit_Schedule005
Purpose	To test whether you can edit a schedule that was just added

Test ID	System 012 PSM_Login_Sunny001
Purpose	To test whether you can successfully login with a correct username/password input

Test ID	System 013 PSM_Login_Rainy001
Purpose	To test whether you can not login with an incorrect username/password input

Test ID	System 014 PSM_Login_Rainy002
Purpose	To test whether you can not login with a different incorrect username/password input

Test ID	System 015 PSM_Login_Rainy003
Purpose	To test whether you can not login with a different incorrect username/password input

Test ID	System 016 PSM_Logout_Rainy001
Purpose	To test whether you can logout by pressing 'X' after editing a class.

Test ID	System 017 PSM_Logout_Rainy002
---------	--------------------------------

Purpose	To test whether you can logout by pressing 'X' after clicking on features (features option not implemented)
---------	---

Test ID	System 018 PSM_Logout_Rainy003
Purpose	To test whether you can logout by pressing 'X' after adding a class.

Test ID	System 019 PSM_Logout_Sunny001
Purpose	To test whether you can logout after logging in.

Test ID	System 020 PSM_Logout_Sunny002
Purpose	To test whether you can logout after editing a class.

Test ID	System 021 PSM_Logout_Sunny003
Purpose	To test whether you can logout after adding a class.

Test ID	System 022 PSM_MessagePopUp_Rainy001
Purpose	To check whether no incorrect message pop-up appears when you log in

Test ID	System 023 PSM_MessagePopUp_Sunny001
---------	--------------------------------------

Purpose	To check whether a successfully logged out message pop-up appears when you log out
---------	--

Test ID	System 024 PSM_MessagePopUp_Sunny002
Purpose	To check whether an incorrect password message pop-up appears when you log in

Test ID	System 025 PSM_MessagePopUp_Sunny003
Purpose	To check whether a 'Too many tries' message pop-up appears after trying to log in with 3 incorrect username/password inputs

Test ID	System 026 PSM_PasswordConflict_Sunny001
Purpose	To check whether a message inquiring a password change appeared after an input of incorrect password/username (not implemented)

Test ID	System 027 PSM_PasswordConflict_Sunny002
Purpose	To check whether a message inquiring a password change appeared after an input of incorrect password/username (not implemented)

Test ID	System 028 PSM_PasswordConflict_Sunny003
Purpose	To check whether a message inquiring a password change appeared after an input of incorrect password/username (not implemented)

Test ID	System 029 PSM_PasswordConflict_Rainy001
Purpose	To check whether a message inquiring a password change did not appear after an input of incorrect password/username (not implemented)

Test ID	System 030 PSM_PasswordConflict_Rainy002
Purpose	To check whether a message inquiring a password change did not appear after an input of incorrect password/username (not implemented)

Test ID	System 031 PSM_PasswordConflict_Rainy003
Purpose	To check whether a message inquiring a password change did not appear after an input of incorrect password/username (not implemented)

Test ID	System 032 PSM_ScheduleSetUp_Rainy001
Purpose	To check whether a schedule could be added with incorrect information (invalid year)

Test ID	System 033 PSM_ScheduleSetUp_Rainy002
Purpose	To check whether a schedule could be added with incorrect information (invalid semester)

Test ID	System 034 PSM_ScheduleSetUp_Rainy003
---------	---------------------------------------

Purpose	To check whether a schedule could be added with incorrect information (invalid meeting time)
---------	--

Test ID	System 035 PSM_ScheduleSetUp_Sunny001
Purpose	To check whether a schedule could be added with a correct input set

Test ID	System 036 PSM_ScheduleSetUp_Sunny002
Purpose	To check whether a schedule could be added with a different correct input set

Test ID	System 037 PSM_ScheduleSetUp_Sunny003
Purpose	To check whether a schedule could be added with a different correct input set

Test Case ID	System038_Security_Rainy
Purpose	To test whether logging in using the correct credentials works

5.1.2. Test Criteria and Procedures

The focus of the tests will be to improve code coverage and generate new test cases based on control flow or data flow of the code. Note that the procedures include the type of code coverage expected and the percentage of coverage.

The test criteria for these test cases required the usage of Rational Functional Tester and Apache Ant. These tools allowed us to run all test cases in a single batch..

An attempt was made to reach 100% branch and statement coverage. For system testing, we did not expect that branch and statement coverage would reach a very high percentage due to a large section of the code being unreachable, as well as certain areas belonging to features that were not completely implemented so they could not be tested.

5.1.3. Test Cases

Unique identifier, purpose, environment set up (preconditions), input, expected output.

Initial Database:

course_id	course_subject	course_name	semester	start_date	end_date	start_mon	end_mon	start_tue	end_tue	start_wed	end_wed	start_thu	end_thu	start_fri	end_fri
4610	COP	OS	Fall	01/07/08	04/20/08	11:00	11:50	NULL	NULL	11:00	11:50	NULL	NULL	11:00	11:50
4625	CDA	Mobile Robo...	Summer A	01/07/08	04/20/08	NULL		12:00	12:50	NULL	NULL	12:00	12:50	NULL	NULL
4010	CEN	Software En...	Sorina	01/07/08	04/20/08	12:00	12:50	NULL	NULL	12:00	12:50	NULL	NULL	12:00	12:50
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Test Case ID	System001_Security_Sunny
Purpose	To test whether you would get a “Too Many Tries” pop up with a set of incorrect password/username inputs
Test Setup	User starts PSM
Test Input	Use record 0:

	<table><tr><th></th><th>Username::String</th><th>Password::String</th></tr><tr><td>0</td><td>root</td><td>rude awakening</td></tr><tr><td>1</td><td>scooter123</td><td>root</td></tr><tr><td>2</td><td>root</td><td>scoot</td></tr></table>		Username::String	Password::String	0	root	rude awakening	1	scooter123	root	2	root	scoot
	Username::String	Password::String											
0	root	rude awakening											
1	scooter123	root											
2	root	scoot											
Expected Output	Program displays Too Many Tries System Exiting pop up												

Test Case ID	System002_Security_Sunny												
Purpose	To test whether you would get a “Too Many Tries” pop up with a 2nd set of three incorrect password/username inputs												
Test Setup	User starts PSM												
Test Input	Use record 1: <table><thead><tr><th></th><th>Username::String</th><th>Password::String</th></tr></thead><tbody><tr><td>0</td><td>root</td><td>rude awakening</td></tr><tr><td>1</td><td>scooter123</td><td>root</td></tr><tr><td>2</td><td>root</td><td>scoot</td></tr></tbody></table>		Username::String	Password::String	0	root	rude awakening	1	scooter123	root	2	root	scoot
	Username::String	Password::String											
0	root	rude awakening											
1	scooter123	root											
2	root	scoot											
Expected Output	Program displays Too Many Tries System Exiting pop up												

Test Case ID	System003_Security_Sunny
Purpose	To test whether you would get a “Too Many Tries” pop up with a third set of three incorrect password/username inputs
Test Setup	User starts PSM

Test Input	<div>Use record 2:</div> <table><thead><tr><th></th><th>Username::String</th><th>Password::String</th></tr></thead><tbody><tr><td>0</td><td>root</td><td>rude awakening</td></tr><tr><td>1</td><td>scooter123</td><td>root</td></tr><tr><td>2</td><td>root</td><td>scoot</td></tr></tbody></table>		Username::String	Password::String	0	root	rude awakening	1	scooter123	root	2	root	scoot
	Username::String	Password::String											
0	root	rude awakening											
1	scooter123	root											
2	root	scoot											
Expected Output	Program displays Too Many Tries System Exiting pop up												

Test Case ID	System004_Clear_Sunny001
Purpose	To test that the end of semester clear function deletes schedules after logging in
Test Setup	Be logged in to the program
Test Input	Click on “End of Clear Semester”
Expected Output	System exits

Test Case ID	System005_Clear_Sunny002
Purpose	To test that the end of semester clear function deletes schedules the last day of the semester
Test Setup	Be logged in to the program
Test Input	(No input, should be done automatically)
Expected Output	System exits

Test Case ID	System006_Clear_Sunny003
Purpose	To test that the end semester of clear function deletes schedules right after logging in
Test Setup	Be logged in to the program
Test Input	(No input, should be done automatically)
Expected Output	System exits

Test Case ID	System007_Clear_Rainy001
Purpose	To test that the end of clear function deletes schedules the last day of the semester after you login on last day
Test Setup	Be logged in to the program
Test Input	(No input, should be done automatically)
Expected Output	System exits

Test Case ID	System008_Clear_Rainy002
Purpose	To test that the end of semester clear function deletes schedules after clicking on advanced features
Test Setup	Be logged in to the program
Test Input	Click on 'Advanced features' button
Expected Output	System exits

Test Case ID	System009_Clear_Rainy003
Purpose	To test that the end of semester clear function deletes schedules after setting up an alarm
Test Setup	Be logged in to the program
Test Input	Click on 'Alarm' button
Expected Output	System exits

Test Case ID	System010_EditSchedule_002
Purpose	To test whether a schedule that already exists
Test Setup	The user is logged in to the program
Test Input	Click "Edit Schedule" button Semester = "Spring" Start Time Tues: 18:00 End Time Thurs: 19:25 Start Time Tues: 18:00 End Time Thurs: 19:25 Click on 'Save' button
Expected Output	Form submits, goes back to main menu. Database is updated with the specified new values

Test Case ID	System011_EditSchedule_002
Purpose	To test whether a schedule that was just added

Test Setup	Be logged in to the program
Test Input	Click "Edit Schedule" button Semester = "Summer A" Click on 'Save' button
Expected Output	Form submits, goes back to main menu. Database is updated with the specified new values

Test Case ID	System012_Login_Sunny001
Purpose	To test that you can log in
Test Setup	Program is running
Test Input	Click on 'Username' box Username: "root" Click on 'Password' box Password: "scooter123"
Expected Output	Application accepts username/password, shows main screen

Test Case ID	System013_Login_Rainy001
Purpose	To test that you can not log in
Test Setup	Program is running
Test Input	Click on 'Username' box Username: "root123" Click on 'Password' box Password: "scooter123456"

Expected Output	Application does not accept username/password input, gives incorrect password pop-up
-----------------	--

Test Case ID	System014_Login_Rainy002
Purpose	To test that you can not log in
Test Setup	Program is running
Test Input	Click on 'Username' box Username: "root123" Click on 'Password' box Password: "scooter123456789"
Expected Output	Application does not accept username/password input, gives incorrect password pop-up

Test Case ID	System015_Login_Rainy003
Purpose	To test that you can not log in
Test Setup	Program is running
Test Input	Click on 'Username' box Username: "root1" Click on 'Password' box Password: "scoot123"
Expected Output	Application does not accept username/password input, gives incorrect password pop-up

Test Case ID	System016_Logout_Rainy001
Purpose	To test if you can log out by clicking 'X' button after editing a schedule
Test Setup	Program is logged in
Test Input	Click on 'Edit Schedule' button Semester = "Spring" Start Date: 1/12/12 Course Number: "4010" Start Time Tues: 18:00 End Time Thurs: 19:25 Start Time Tues: 18:00 End Time Thurs: 19:25 Click on 'Save' button Click on 'X' button
Expected Output	Program cannot logout, must click logout button

Test Case ID	System017_Logout_Rainy002
Purpose	To test if you can log out by clicking 'X' button after clicking on features (features not implemented)
Test Setup	Program is logged in
Test Input	Click on 'Features' button Click on 'X' button
Expected Output	Program cannot logout, must click logout button

Test Case ID	System018_Logout_Rainy003
--------------	---------------------------

Purpose	To test if you can log out by clicking 'X' button after adding a class
Test Setup	Program is logged in
Test Input	Click on 'Edit Schedule' button Semester = "Spring" Start Date: "1/12/12" End Date: "1/12/13" Course Number: "4010" Course Title: "COP" Course Name: "Computer Testing" Start Time Tues: "12:00" End Time Thurs: "12:50" Start Time Tues: "12:00" End Time Thurs: "12:50" Click on 'Save' button Click on 'X' button
Expected Output	Program cannot logout, must click logout button

Test Case ID	System019_Logout_Sunny001
Purpose	To test if you can log out after logging in
Test Setup	Program is logged in
Test Input	Click on 'Logout' button
Expected Output	Program logs out successfully

Test Case ID	System020_Login_Sunny002
Purpose	To test if you can log out after editing a class

Test Setup	Program is logged in
Test Input	Click on 'Edit Schedule' button Semester = "Fall" Click on 'Save' button Click on 'Logout' button
Expected Output	Program logs out successfully.

Test Case ID	System021_Logout_Sunny003
Purpose	To test if you can log out after adding a class
Test Setup	Program is logged in
Test Input	Click on 'Edit Schedule' button Semester = "Fall" Start Date: "1/12/12" End Date: "2/12/13" Course Number: "4010" Course Title: "CEN" Course Name: "Software Engineering" Start Time Tues: "12:00" End Time Thurs: "12:50" Start Time Tues: "12:00" End Time Thurs: "12:50" Click on 'Save' button Click on 'Logout' button
Expected Output	Program logs out successfully.

Test Case ID	System022_MessagePopUp_Rainy001
--------------	---------------------------------

Purpose	To test whether no message pop up appears when you log in
Test Setup	Program is running
Test Input	Click 'Username' box Username: "root" Click 'Password' box Username: "scooter123" Click 'Log In'
Expected Output	Program displays no message pop up

Test Case ID	System023_MessagePopUp_Sunny001
Purpose	To test whether a message pops up when you log out
Test Setup	User is logged in
Test Input	Click 'Log out' button
Expected Output	Program displays a 'Successful Logout' message pop-up

Test Case ID	System024_MessagePopUp_Sunny002
Purpose	To test whether a message pops up when you input incorrect password
Test Setup	User is logged in
Test Input	Click 'Log out' button
Expected Output	Program displays a 'Successful Logout' message pop-up

Test Case ID	System025_MessagePopUp_Sunny003
Purpose	To test whether a message pops up when you input incorrect password
Test Setup	User is logged in
Test Input	Click 'Log out' button
Expected Output	Program displays a 'Successful Logout' message pop-up

Test Case ID	System026_PassCon001
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: "root" Password: "scooter12" Click 'Login' button
Expected Output	Program displays a 'incorrect username or password' message pop-up with a password reset button.

Test Case ID	System027_PassCon002
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: "root" Password: "ab"

	Click 'Login' button
Expected Output	Program displays a 'incorrect username or password' message pop-up with a password reset button.

Test Case ID	System028_PassCon003
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: "root" Password: "ads" Click 'Login' button
Expected Output	Program displays a 'incorrect username or password' message pop-up with a password reset button.

Test Case ID	System029_PassCon004
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: "root" Password: "asde" Click 'Login' button
Expected Output	Program displays a 'incorrect username or password' message pop-up with a password reset button.

Test Case ID	System030_PassCon005
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: “pop” Password: “star” Click ‘Login’ button
Expected Output	Program displays a ‘incorrect username or password’ message pop-up with a password reset button.

Test Case ID	System031_PassCon006
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: “clarky” Password: “boi” Click ‘Login’ button
Expected Output	Program displays a ‘incorrect username or password’ message pop-up with a password reset button.

Test Case ID	System032_ScheduleSetup_Rainy
Purpose	To test whether add schedule accepts a set of invalid values for start date, end date, semester, subject, course number, course name.
Test Setup	User is logged in to system, click “add class schedule”

Test Input	Use record 0: <table><tr><th></th><th>Subject::String</th><th>End date::String</th><th>Semester::java.la...</th><th>Start date::String</th><th>Course number::...</th><th>Course name::St...</th></tr><tr><td>0</td><td>CEN</td><td>23/12/12</td><td>Fall</td><td>10/02/12</td><td>1000</td><td>gender studies</td></tr><tr><td>1</td><td>123</td><td>12/12/12</td><td>Spring</td><td>12/11/12</td><td>123</td><td>420</td></tr><tr><td>2</td><td>facebook</td><td>10/10/10</td><td>Fall</td><td>24/07/12</td><td>000</td><td>sesame street</td></tr></table>		Subject::String	End date::String	Semester::java.la...	Start date::String	Course number::...	Course name::St...	0	CEN	23/12/12	Fall	10/02/12	1000	gender studies	1	123	12/12/12	Spring	12/11/12	123	420	2	facebook	10/10/10	Fall	24/07/12	000	sesame street
	Subject::String	End date::String	Semester::java.la...	Start date::String	Course number::...	Course name::St...																							
0	CEN	23/12/12	Fall	10/02/12	1000	gender studies																							
1	123	12/12/12	Spring	12/11/12	123	420																							
2	facebook	10/10/10	Fall	24/07/12	000	sesame street																							
Expected Output	Program saves invalid data to database																												

Test Case ID	System033_ScheduleSetup_Rainy																												
Purpose	To test whether add schedule accepts a 2nd set of invalid values for start date, end date, semester, subject, course number, course name.																												
Test Setup	User is logged in to system, click “add class schedule”																												
Test Input	<div>Use record 1:</div> <table><tr><th></th><th>Subject::String</th><th>End date::String</th><th>Semester::java.la...</th><th>Start date::String</th><th>Course number:...</th><th>Course name::St...</th></tr><tr><td>0</td><td>CEN</td><td>23/12/12</td><td>Fall</td><td>10/02/12</td><td>1000</td><td>gender studies</td></tr><tr><td>1</td><td>123</td><td>12/12/12</td><td>Spring</td><td>12/11/12</td><td>123</td><td>420</td></tr><tr><td>2</td><td>facebook</td><td>10/10/10</td><td>Fall</td><td>24/07/12</td><td>000</td><td>sesame street</td></tr></table>		Subject::String	End date::String	Semester::java.la...	Start date::String	Course number:...	Course name::St...	0	CEN	23/12/12	Fall	10/02/12	1000	gender studies	1	123	12/12/12	Spring	12/11/12	123	420	2	facebook	10/10/10	Fall	24/07/12	000	sesame street
	Subject::String	End date::String	Semester::java.la...	Start date::String	Course number:...	Course name::St...																							
0	CEN	23/12/12	Fall	10/02/12	1000	gender studies																							
1	123	12/12/12	Spring	12/11/12	123	420																							
2	facebook	10/10/10	Fall	24/07/12	000	sesame street																							
Expected Output	Program saves invalid data to database																												

Test Case ID	System034_ScheduleSetup_Rainy																												
Purpose	To test whether add schedule accepts a 3rd set of invalid values for start date, end date, semester, subject, course number, course name.																												
Test Setup	User is logged in to system, click “add class schedule”																												
Test Input	Use record 2: <table><tr><th></th><th>Subject::String</th><th>End date::String</th><th>Semester::java.la...</th><th>Start date::String</th><th>Course number::...</th><th>Course name::St...</th></tr><tr><td>0</td><td>CEN</td><td>23/12/12</td><td>Fall</td><td>10/02/12</td><td>1000</td><td>gender studies</td></tr><tr><td>1</td><td>123</td><td>12/12/12</td><td>Spring</td><td>12/11/12</td><td>123</td><td>420</td></tr><tr><td>2</td><td>facebook</td><td>10/10/10</td><td>Fall</td><td>24/07/12</td><td>000</td><td>sesame street</td></tr></table>		Subject::String	End date::String	Semester::java.la...	Start date::String	Course number::...	Course name::St...	0	CEN	23/12/12	Fall	10/02/12	1000	gender studies	1	123	12/12/12	Spring	12/11/12	123	420	2	facebook	10/10/10	Fall	24/07/12	000	sesame street
	Subject::String	End date::String	Semester::java.la...	Start date::String	Course number::...	Course name::St...																							
0	CEN	23/12/12	Fall	10/02/12	1000	gender studies																							
1	123	12/12/12	Spring	12/11/12	123	420																							
2	facebook	10/10/10	Fall	24/07/12	000	sesame street																							

Expected Output	Program saves invalid data to database
-----------------	--

Test Case ID	System035_ScheduleSetup_Sunny																																																																																																			
Purpose	To test whether add schedule accepts a set valid values into the database																																																																																																			
Test Setup	User is logged in to system, click “add class schedule”																																																																																																			
Test Input	<div>Use record 0:</div> <table><thead><tr><th>Semester:java.la...</th><th>Start date:String</th><th>End date:String</th><th>Subject:String</th><th>Course number:...</th><th>Course name:St...</th><th>MondayStartTim...</th><th>MondayEndTim...</th><th>TuesdayStartTim...</th><th>TuesdayEndTim...</th><th>WednesdayStart...</th></tr></thead><tbody><tr><td>0 Fall</td><td>01/07/08</td><td>04/20/08</td><td>COP</td><td>4610</td><td>OS</td><td>11:00</td><td>11:50</td><td></td><td></td><td>11:00</td></tr><tr><td>1 Summer A</td><td>01/07/08</td><td>04/20/08</td><td>CDA</td><td>4625</td><td>Mobile Robotics</td><td></td><td></td><td>12:00</td><td>12:50</td><td></td></tr><tr><td>2 Spring</td><td>01/07/08</td><td>04/20/08</td><td>CEN</td><td>4010</td><td>Software Engine...</td><td>12:00</td><td>12:50</td><td></td><td></td><td>12:00</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>WednesdayEndT...</td><td>ThursdayStartTi...</td><td>ThursdayEndTi...</td><td>FridayStartTime:...</td><td>FridayEndTime:...</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>11:50</td><td></td><td></td><td>11:00</td><td>11:50</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>12:00</td><td>12:50</td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>12:50</td><td></td><td></td><td>12:00</td><td>12:50</td></tr></tbody></table>	Semester:java.la...	Start date:String	End date:String	Subject:String	Course number:...	Course name:St...	MondayStartTim...	MondayEndTim...	TuesdayStartTim...	TuesdayEndTim...	WednesdayStart...	0 Fall	01/07/08	04/20/08	COP	4610	OS	11:00	11:50			11:00	1 Summer A	01/07/08	04/20/08	CDA	4625	Mobile Robotics			12:00	12:50		2 Spring	01/07/08	04/20/08	CEN	4010	Software Engine...	12:00	12:50			12:00																		WednesdayEndT...	ThursdayStartTi...	ThursdayEndTi...	FridayStartTime:...	FridayEndTime:...							11:50			11:00	11:50								12:00	12:50									12:50			12:00	12:50
Semester:java.la...	Start date:String	End date:String	Subject:String	Course number:...	Course name:St...	MondayStartTim...	MondayEndTim...	TuesdayStartTim...	TuesdayEndTim...	WednesdayStart...																																																																																										
0 Fall	01/07/08	04/20/08	COP	4610	OS	11:00	11:50			11:00																																																																																										
1 Summer A	01/07/08	04/20/08	CDA	4625	Mobile Robotics			12:00	12:50																																																																																											
2 Spring	01/07/08	04/20/08	CEN	4010	Software Engine...	12:00	12:50			12:00																																																																																										
						WednesdayEndT...	ThursdayStartTi...	ThursdayEndTi...	FridayStartTime:...	FridayEndTime:...																																																																																										
						11:50			11:00	11:50																																																																																										
							12:00	12:50																																																																																												
						12:50			12:00	12:50																																																																																										
Expected Output	Program saves valid data to database																																																																																																			

Test Case ID	System036_ScheduleSetup_Sunny																																																																																																														
Purpose	To test whether add schedule accepts a 2nd set of valid values into the database																																																																																																														
Test Setup	User is logged in to system, click “add class schedule”																																																																																																														
Test Input	<div>Use record 1:</div> <table><tr><th>Semester:java.la...</th><th>Start date:String</th><th>End date:String</th><th>Subject:String</th><th>Course number:...</th><th>Course name:St...</th><th>MondayStartTim...</th><th>MondayEndTim...</th><th>TuesdayStartTim...</th><th>TuesdayEndTim...</th><th>WednesdayStart...</th></tr><tr><td>0 Fall</td><td>01/07/08</td><td>04/20/08</td><td>COP</td><td>4610</td><td>OS</td><td>11:00</td><td>11:50</td><td></td><td></td><td>11:00</td></tr><tr><td>1 Summer A</td><td>01/07/08</td><td>04/20/08</td><td>CDA</td><td>4625</td><td>Mobile Robotics</td><td></td><td></td><td>12:00</td><td>12:50</td><td></td></tr><tr><td>2 Spring</td><td>01/07/08</td><td>04/20/08</td><td>CEN</td><td>4010</td><td>Software Engine...</td><td>12:00</td><td>12:50</td><td></td><td></td><td>12:00</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>WednesdayEndT...</td><td>ThursdayStartTi...</td><td>ThursdayEndTi...</td><td>FridayStartTime:...</td><td>FridayEndTime:...</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>11:50</td><td></td><td></td><td>11:00</td><td>11:50</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>12:00</td><td>12:50</td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>12:50</td><td></td><td></td><td>12:00</td><td>12:50</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table>	Semester:java.la...	Start date:String	End date:String	Subject:String	Course number:...	Course name:St...	MondayStartTim...	MondayEndTim...	TuesdayStartTim...	TuesdayEndTim...	WednesdayStart...	0 Fall	01/07/08	04/20/08	COP	4610	OS	11:00	11:50			11:00	1 Summer A	01/07/08	04/20/08	CDA	4625	Mobile Robotics			12:00	12:50		2 Spring	01/07/08	04/20/08	CEN	4010	Software Engine...	12:00	12:50			12:00																		WednesdayEndT...	ThursdayStartTi...	ThursdayEndTi...	FridayStartTime:...	FridayEndTime:...							11:50			11:00	11:50								12:00	12:50									12:50			12:00	12:50											
Semester:java.la...	Start date:String	End date:String	Subject:String	Course number:...	Course name:St...	MondayStartTim...	MondayEndTim...	TuesdayStartTim...	TuesdayEndTim...	WednesdayStart...																																																																																																					
0 Fall	01/07/08	04/20/08	COP	4610	OS	11:00	11:50			11:00																																																																																																					
1 Summer A	01/07/08	04/20/08	CDA	4625	Mobile Robotics			12:00	12:50																																																																																																						
2 Spring	01/07/08	04/20/08	CEN	4010	Software Engine...	12:00	12:50			12:00																																																																																																					
						WednesdayEndT...	ThursdayStartTi...	ThursdayEndTi...	FridayStartTime:...	FridayEndTime:...																																																																																																					
						11:50			11:00	11:50																																																																																																					
							12:00	12:50																																																																																																							
						12:50			12:00	12:50																																																																																																					

5.2. Actual Test Results

The test results below will display the test results for each test and whether they resulted in a Pass or Failure. Directly after that the coverage percentage for branch and statement coverage will be displayed for each tested package and for each coverage tool that was used.

Test ID	Pass/Fail/Error
System001	Pass
System002	Pass
System003	Pass
System004	Fail
System005	Fail
System006	Fail
System007	Fail
System008	Fail
System009	Fail
System010	Pass

System011	Pass
System012	Pass
System013	Pass
System014	Pass
System015	Pass
System016	Pass
System017	Pass
System018	Pass
System019	Pass
System020	Pass
System021	Pass
System022	Pass
System023	Pass
System024	Pass
System025	Pass
System026	Fail
System027	Fail
System028	Fail
System029	Fail
System030	Fail

System031	Fail
System032	Pass
System033	Pass
System034	Pass
System035	Pass
System036	Pass
System037	Pass
System038	Pass`

Original Coverage

Coverage Type:	Statement Coverage	Branch Coverage
Coverage Tool:	Cobertura	Cobertura
Interface package	74%	37%
Logic package	40%	23%

Storage package	13%	37%
Total	61%	26%

Updated Coverage

Coverage Type:	Statement Coverage	Branch Coverage
Coverage Tool:	Cobertura	Cobertura
Interface package	88%	87%
Logic package	64%	50%
Storage package	77%	62%
Total	83%	54%

CHAPTER 6: Test Summary Report

This chapter describes a compilation of the results of all types of testing including failures, solutions, and code coverage.

6.1 Failures and Possible Solutions

The following subsection lists all the failures that occurred during each type of testing and proposed solutions on how to correct the errors.

Unit Testing

Test Case ID	Unit002_AppCont002
Purpose	To test that the checkTimes() method schedules the popup for 5 min, 15 min, and the end of class, if there is a course on that day.
Test Setup	appController instantiated DBConnection mocked ArrayList<Integer> mockCourseList; mockCourseList.add(1); Set each class end time to 6:15.
Test Input	appController.checkTimes()
Expected Output	State variables endofclass, popup5min, and popup15min are scheduled for execution at 6:15, 6:10, 6:00 respectively.

Actual Output	An assertion error message was displayed at the first assertion point: expected <6> but was <7> .
Possible Solution	Using a custom scheduling function or class may be necessary, since the java.util.Timer seems to not work as intended. Not only may this relieve the error, but the current Timer only returns the correct value for execution time of the scheduled task after the task has been executed. This is not ideal in seeing when a scheduled execution is going to occur in the future. Hence creating a custom scheduling class possibly derived from the existing Timer class could be a solution.

Test Case ID	Unit029_AppCont029
Purpose	To test that when the Thread class attempts to sleep it throws an exception.
Test Setup	appController instantiated PowerMockito.mockStatic(Thread.class); PowerMockito.when(Thread.class).thenThrow (Exception.class);
Test Input	appController.sleep(100);
Expected Output	sleep() throws an exception.
Actual Output	java.lang.AssertionError: Expected exception: java.lang.Exception
Possible Solution	Find another way to throw the exception with PowerMockito as this seems to be an error with Mockito not throwing the Exception.class.

Subsystem Testing

Test Case ID	SubSys31 - StoreClassSched2
Purpose	Verify that when the method storeClassSched() is called this one does not store class schedule in the database.
Test Setup	appController instantiated Mockito.doReturn(-1).when(db).storeClassSched(4072, "CEN", "04/25/2019", "6:30PM", "7:40PM", "N/A", "N/A", "6:30PM", "7:40PM", "N/A", "N/A", "N/A", "N/A", "N/A", "N/A");
Test Input	app1.storeClassSchedRunner();
Expected Output	Call to the method should not store the schedule and return -1
Actual Output	An assertion error message was displayed at the first assertion point: expected <-1> but was <0> .
Possible Solution	One possible solution is to find another way to stub the method call since it gives an error with the mock of the method and it returns different value.

Test Case ID	SubSys32 - logIn
Purpose	Verify that when the method LogIn() is called this one return 0 if the login information connects to the database.
Test Setup	appController instantiated Mockito.doReturn(0).when(db).connect("root", "12345");
Test Input	app1.LogIn();
Expected Output	Call to the method should return 0
Actual Output	Arguments are different wanted, expected: dBConnection.connect("root", "12345"); Invocation: dBConnection.connect(null, null);
Possible Solution	Part of the testing procedure was to refactor the appController class which mean that some of the call needed for the code to function properly is inside void methods. This error could be solved if setters and getters are added to the class.

Test Case ID	SubSys33 - logIn2
Purpose	Verify that when the method MsgAckRunner() is called this one will false for the display of message
Test Setup	appController instantiated Mockito.doReturn(-1).when(db).connect("root", "12345");
Test Input	app1.LogIn();
Expected Output	Call to the method should return -1
Actual Output	Arguments are different wanted, expected: dBConnection.connect("root", "12345"); Invocation: dBConnection.connect(null, null);
Possible Solution	Part of the testing procedure was to refactor the appController class which mean that some of the call needed for the code to function properly is inside void methods. This error could be solved if setters and getters are added to the class.

System Testing

Test Case ID	System004_Clear_Sunny001
Purpose	To test that the end of semester clear function deletes schedules after logging in
Test Setup	Be logged in to the program
Test Input	Click on "End of Clear Semester"
Expected Output	Schedule is deleted.
Actual Output	Schedule is not cleared.
Possible Solution	Implement a method that deletes the schedule.

Test Case ID	System005_Clear_Sunny002
Purpose	To test that the end of semester clear function deletes schedules the last day of the semester
Test Setup	Be logged in to the program
Test Input	(No input, should be done automatically)
Expected Output	Schedule is deleted.
Actual Output	Schedule is not deleted
Possible Solution	Implement a method that deletes the schedule during the last day of the semester. Might need a timer function.

Test Case ID	System006_Clear_Sunny003
Purpose	To test that the end semester of clear function deletes schedules right after logging in
Test Setup	Be logged in to the program
Test Input	(No input, should be done automatically) (Not implemented)
Expected Output	Schedule is deleted.
Actual Output	Schedule is not deleted.
Possible Solution	Implements a method that deletes the schedule when the end of semester clear button is clicked.

Test Case ID	System007_Clear_Rainy001
--------------	--------------------------

Purpose	To test that the end of clear function deletes schedules the last day of the semester after you login on last day
Test Setup	Be logged in to the program
Test Input	(No input, should be done automatically)
Expected Output	Schedule is deleted.
Actual Output	Schedule is not deleted.
Possible Solution	Implements a method that deletes the schedule when the end of semester clear button is clicked.

Test Case ID	System008_Clear_Rainy002
Purpose	To test that the end of semester clear function deletes schedules after clicking on advanced features
Test Setup	Be logged in to the program
Test Input	Click on 'Advanced features' button
Expected Output	Schedule is deleted.
Actual Output	Schedule is not deleted.
Possible Solution	Implements a method that deletes the schedule when the end of semester clear button is clicked.

Test Case ID	System009_Clear_Rainy003
Purpose	To test that the end of semester clear function deletes schedules after setting up an alarm

Test Setup	Be logged in to the program
Test Input	Click on 'Alarm' button (Alarm not implemented)
Expected Output	Schedule is deleted. (Not implemented)
Actual Output	Schedule is not deleted.
Possible Solution	Implements a method that deletes the schedule when the end of semester clear button is clicked.

Test Case ID	System026_PassCon001
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: "root" Password: "scooter12" Click 'Login' button
Expected Output	Program displays a 'incorrect username or password' message pop-up with a password reset button.
Actual Output	Program displays a 'incorrect username or password' message pop-up with no password reset button.
Possible Solution	Implement a button that allows to reset the password for a specific user.

Test Case ID	System027_PassCon002
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM

Test Input	Username: “root” Password: “ab” Click ‘Login’ button
Expected Output	Program displays a ‘incorrect username or password’ message pop-up with a password reset button.
Actual Output	Program displays a ‘incorrect username or password’ message pop-up with no password reset button.
Possible Solution	Implement a button that allows to reset the password for a specific user.

Test Case ID	System028_PassCon003
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: “root” Password: “ads” Click ‘Login’ button
Expected Output	Program displays a ‘incorrect username or password’ message pop-up with a password reset button.
Actual Output	Program displays a ‘incorrect username or password’ message pop-up with no password reset button.
Possible Solution	Implement a button that allows to reset the password for a specific user.

Test Case ID	System029_PassCon004
Purpose	To test whether the password reset button appears

Test Setup	User starts PSM
Test Input	Username: “root” Password: “asde” Click ‘Login’ button
Expected Output	Program displays a ‘incorrect username or password’ message pop-up with a password reset button.
Actual Output	Program displays a ‘incorrect username or password’ message pop-up with no password reset button.
Possible Solution	Implement a button that allows to reset the password for a specific user.

Test Case ID	System030_PassCon005
Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: “pop” Password: “star” Click ‘Login’ button
Expected Output	Program displays a ‘incorrect username or password’ message pop-up with a password reset button.
Actual Output	Program displays a ‘incorrect username or password’ message pop-up with no password reset button.
Possible Solution	Implement a button that allows to reset the password for a specific user.

Test Case ID	System031_PassCon006
--------------	----------------------

Purpose	To test whether the password reset button appears
Test Setup	User starts PSM
Test Input	Username: "clarky" Password: "boi" Click 'Login' button
Expected Output	Program displays a 'incorrect username or password' message pop-up with a password reset button.
Actual Output	Program displays a 'incorrect username or password' message pop-up with no password reset button.
Possible Solution	Implement a button that allows to reset the password for a specific user.

6.2. Code Coverage Results

This subsection compares the results for all code coverage (unit, subsystem, system) using at least three(3) tools. It shows the number of test cases, bugs found, and coverage, for each tool, for both specification-based and implementation-based testing for unit, subsystem and system testing. The types of code coverage each tool offers is explained.

All of the coverage tools offered statement and branch coverage except for EcEmma 1.5.3, which was used in implementation based testing for the appController unit testing.

Specification Based Testing

Unit Testing

	Test Cases	Bugs
appController	40	1
DBConnection	2	0

Coverage Type:	Statement Coverage			Branch Coverage		
Coverage Tool:	EclEmma	Clover	CodeCover	EclEmma	Clover	CodeCover
appController	54.2%	48.2%	X	X	26.8%	X
DBConnection	7.5%	6.7%	6.7%	0%	0%	3.4%

Subsystem Testing

	Test Cases	Bugs
Logic Package	20	0

Coverage Type:	Statement Coverage		Branch Coverage	
Coverage Tool:	EclEmma	Clover	EclEmma	Clover
Logic Package	16.2%	0.5%	0.0%	0.0%

This results were recorded with the test cases obtained on part 1 of the project which were wrong and had to be created again.

System Testing

	Test Cases	Bugs
System testing	8	0

Coverage Type:	Statement Coverage	Branch Coverage
Coverage Tool:	Cobertura	Cobertura

Interface package	74%	37%
Logic package	40%	23%
Storage package	13%	37%
Total	61%	26%

Implementation Based Testing

Unit Testing

	Test Cases	Bugs
appController	50	2
DBConnection	51	0

Coverage Type:	Statement Coverage			Branch Coverage		
Coverage Tool:	EclEmma	Clover	CodeCover	EclEmma	Clover	CodeCover

appController	97.8%	95.8	X	X	94.6%	X
DBConnection	93.6%	97.3%	95.9%	100%	100%	89.7%

Subsystem Testing

	Test Cases	Bugs
Logic Package	36	3

Coverage Type:	Statement Coverage		Branch Coverage	
Coverage Tool:	EclEmma	Clover	EclEmma	Clover
Logic Package	42.9%	47.6%	20.0%	21.2%

Reaching higher than 80% was not possible due to some errors encountered when trying to test calls that were supposed to connect to the database, and when the system needed to create new schedules. With a few more test cases it could have been possible to reach a higher percentage.

System Testing

	Test Cases	Bugs
--	------------	------

System testing	38	0
----------------	----	---

Coverage Type:	Statement Coverage	Branch Coverage
Coverage Tool:	Cobertura	Cobertura
Interface package	88%	87%
Logic package	64%	50%
Storage package	77%	62%
Total	83%	54%

We were unable to reach above 80% branch coverage for system testing due to the fact that a lot of the code in the Interface and Storage packages was unreachable and therefore unable to be tested.

CHAPTER 7: Risks and Contingencies

- 1.) Testing on a machine that is not compatible with the Java Virtual Machine.
- 2.) Not having Java installed.
- 3.) Having hardware that does not have at least these performance aspects: A 2.20Ghz processor speed, 4GB RAM, 1.5GB free hard drive space.
- 4.) Not having the proper Rational Functional Tester plug in. After a long time researching multiple tools that could work with RFT we discovered that they were all deprecated except for an old version of Cobertura, which needed an old version of Apache Ant.

- 5.) Not having the proper Eclipse environment.
- 6.) Classes in PSM_Interface, PSM_Logic, PSM_Storage may be refactored incorrectly.
- 7.) Missing dependencies such as coverage or testing tools.

CHAPTER 8: Approvals

Name	Date	E-Signature
Dario	4/19/19	Dario Quintinilla
Denis	4/19/19	Denis Ortega
Luis	4/19/19	Luis Gonzalez
Antonio	4/19/19	Antonio Valdes
Daniel	4/19/19	Daniel Neel
Miguel	4/19/19	Miguel A. Martinez

CHAPTER 9: Glossary

This chapter defines some terms used in the document, especially domain specific terms.

Control Flow: the order in which individual statements, instructions or function calls of an imperative program are executed or evaluated.

Data flow: test adequacy is concerned with the coverage of flow-graph paths that are significant for the data flow in the program.

Equivalence Classes: when the elements of some set S have a notion of equivalence defined on them, then one may naturally split the set S into equivalence classes.

CHAPTER 10: Appendix

10.1. Appendix A

Test schedule (Gantt chart or PERT chart).



10.2. Appendix B

All implemented use cases.

Use Case ID: PSM_001 - Login

Details:

- Actor: Florida International University Professor, system.
- Pre-conditions:
 1. The user shall start computer.
- Description:
 1. Use case begins when the system provides the user with a template for data entry.
 2. The user shall enter the following data: username, and password.
 3. The user shall request access by selecting the log in button.
 4. The system shall validate the data provided by user.
 5. Use case ends when the system allows access to user.
- Post-conditions:
 1. The system shall record the number of log in into the system.
 2. The user shall set up schedule.

Alternative Courses of Action

1. In step 2 of the description section the user has the option to cancel the request.
2. In step 3 of the description section if any of the required fields are blank the system shall request the user to make an entry in the appropriate field.

Exceptions

1. The log in button in the initial data entry template is not active.
2. The cancel button in the initial data entry template is not active.
3. The help button in the initial data entry template is not active.

Related Use Cases:

1. Misuse case – help password.
 2. Security – limit password attempts.
 3. Set up schedule during initial log in.
-

Decision Support

Frequency: the professor will implement use case at least three times a week.

Criticality: High. This use case allows the user to have access to the system in which the user can see, review, and manage his/her shchedule.

Risk: Medium. This use case is related to security tasks.

Constraints:

1. Usability:
 - a) Help facility should provide 1 help frame on the topic
 - b) On average the user should take up to 1 min to complete the send request form.
2. Reliability:
 - a) Availability – Down time for Login Back-up 5 minutes in a 24 hour period.
3. Performance:
 - a) Request should be sent and saved within 5 seconds.
 - b) System should be able to handle 1 request at a time per user.
4. Supportability:

- a) Since the program will be coded on Java, any system with the Java Virtual Machine should execute the use case correctly.
-

Modification History

Owner: CEN4010_team2sp08

Initiation date: 01/25/2008

Date last modified: 02/16/2008

Use Case ID: PSM_002 - Logout

Details:

- Actor: Florida International University Professor, system.
- Pre-conditions:
 - 1.The user has succesfully looged onto the system.
 - 2.The user shall request to log off the system
- Description:
 1. Use case begins when user submit request to log off the system..
 2. The system shall provide the user with a logout template.
 3. The user shall click the logout link in the logout template.
 4. Use case ends when the system allows logout request.
- Post-conditions:
 - 1.The system shall allow access if the user submitts a new request.

Alternative Courses of Action

1. In step 3 of description section the user has the option to cancel resquest of logout.

Exceptions:

1. The logout button in the logout template is not active.
2. The cancel button in the logout template is not active.

Related Use Cases:

1. Security - Custom idle auto logout.
-

Decision Support

Frequency: the professor will implement use case at least three times a week.

Criticality: High. This use case allows the user to shut down system for security purposes.

Risk: Medium. This use case is related to security tasks.

Constraints:

1. Usability:
 - a. Help facility should provide 1 help frame on the topic
 - b. On average the user should take up to 30 seconds to complete the logout request form.
2. Reliability:
 - a. Availability – Down time for Login Back-up 5 minutes in a 24 hour period.
3. Performance:
 - a. Request should be sent and saved within 5 seconds.
 - b. System should be able to handle 1 request at a time per user.
4. Supportability:

- a. Since the program will be coded on Java, any system with the Java Virtual Machine should execute the use case correctly.
-

Modification History

Owner: CEN4010_team2sp08

Initiation date: 01/25/2008

Date last modified: 02/16/2008

Use Case ID: PSM_003 - Security

Details:

- Actor: Florida International University Professor, system.
- Pre-conditions:
 1. The user has successfully logged onto the system.
- Description:
 1. Use case begins when the user access custom features.
 2. The system shall provide the user with a template to manage custom idle auto logout.
 3. The user shall select the best option regarding his/her class schedule.
 4. The user shall click the OK button in the template.
 5. Use case ends when the system allows changes.
- Post-conditions:

1. The system shall record the changes made by user. .

Alternative Courses of Action

1. In step 2 of the description section the user has the option to cancel the request.

Exceptions

1. The custom feature access button template is not active.
2. Custom options in the template are not active.
3. The OK button in the template is not active.
4. The system is not allowing new changes.

Related Use Cases:

1. Custom features
 2. Limit password attempts – log in use case
 3. Custom idle auto logout.
-

Decision Support

Frequency: the user will implement use case one a week.

Criticality: High. This use case is essentially important when dealing with security issues in the system.

Risk: Medium. This use case is related to security tasks.

Constraints:

1. Usability:

- a. On average the user should take up to 5 minutes to complete the custom features request form.
- 2. Reliability:
 - a. Availability – Down time for Login Back-up 5 minutes in a 24 hour period
- 3. Performance:
 - a. Request should be sent and saved within 5 seconds.
 - b. System should be able to handle 1 request at a time per user.
- 4. Supportability:
 - a. Since the program will be coded on Java, any system with the Java Virtual Machine should execute the use case correctly.

Modification History

Owner: CEN4010_team2sp08

Initiation date: 01/25/2008

Date last modified: 02/16/2008

Use Case ID: PSM_004 – Schedule Setup

Details:

· Actor: Florida International University Professor

· Pre-conditions:

1. The user shall start computer.
2. The user has successfully logged onto the system.

· Description:

1. Use case begins when the user requests schedule set up.
2. The system shall provide the user with a template for schedule data entry.
3. The user shall enter the following data: class name, start and end date of semester, days, and time.
4. The user shall click the OK button in the schedule data entry template.
5. The system shall validate data provided by user.
6. The system shall check for conflicts in the schedule.
7. The system shall save new shedule.
8. Use case ends when the system accepts data provided by user.

· Post-conditions:

1. The user is allow into the system.
2. The user can have acces to the schedule
3. The user is allow to manage

Alternative Courses of Action

1. In step 4 of description section the user has the option to cancel schedule setup form.

Exceptions:

1. The schedule set up option is not active.
2. The OK button in the schedule data entry template is not active.

Related Use Cases:

1. End of semester clear up.

2. Schedule Conflicts.
 3. Alarms.
 4. Exam set up
 5. Presentation set up
 6. Single day.
 7. Edit shedule
 8. Remove shedule
 9. Professor attendance
-

Decision Support

Frequency: the user will implement use case in a weekly basis.

Criticality: High. This use case enables the user to implement the system efficiently.

Risk: Medium to Low.

Constraints:

1. Usability:
 - a. On average the user should take up to 15 min to complete the setting up of schedule form.
2. Reliability:
3. Performance:
 - a. Request should be sent and saved within 5 sec.
 - b. System should be able to handle 1 request at a time per user.
4. Supportability:

- a. Since the program will be coded on Java, any system with the Java Virtual Machine should execute the use case correctly.
-

Modification History

Owner: CEN4010_team2sp08

Initiation date: 01/25/2008

Date last modified: 02/16/2008

Use Case ID: PSM_008 – Message Popup

Details:

Actor: User

Pre-Conditions:

1. PSM user is logged on.
2. Schedule has been set-up.

Description:

1. The use case begins in a pre-defined situation which requires a warning message pop-up.
2. The system shall warn the user about the current situation by displaying a pop-up message as well as an audible sound to let the user know there's a message to check if he or she is not in front of the computer.

3. The use case ends once the user acknowledges the message by clicking OK, the popup will disappear and it will mark the end of the use case.

Post-Conditions:

1. User will acknowledge the pop-up window and the schedule manager will continue to run as a background process.

Alternative Courses of Action: N/A

Exceptions:

1. The system pop-up may not occur.
2. The user may not acknowledge the message.

Related Use Cases:

1. PSM_CW01 – Custom Warning
2. PSM_PP01 – Program Priority
3. PSM_AL01 – Alarms

Decision Support:

Frequency: On average 2-4 times per class.

Criticality: High. This is the main function of the PSM.

Risk: Low. System pop-ups are very easy and simple to code and low error rate.

Constraints:

1. Usability:
 1. Only user input required is an acknowledgement of the pop-up.
2. Reliability:
 1. System pop-ups are the most important function of the system and should work correctly 100% of the time.
3. Performance:
 1. Pop-up windows use up very little system resources, this isn't a factor.
4. Supportability:
 1. Since the program will be coded on Java, any system with the Java Virtual Machine should execute the use case correctly.

Modification History:

Owner: CEN4010_team2sp08

Initiation Date: 01/27/2008

Date Last Modified: 02/16/2008

Use Case ID: PSM_012- Edit Schedule

Use Case Level: Functional sub-use case

Details

· Actor: Florida International University Professor

· Pre-conditions:

1. The user has successfully logged into the system.
2. The user has clicked on Edit Schedule.

· Description:

1. Use case begins when the system provides the user with a template for data entry with current values pre-filled.
2. The user can enter or change the following data: class name, start and end date of semester, days of the week, and time.
3. The user shall click the Save button in the edit schedule template for completion.
4. The system shall validate data provided by user.
5. The system shall check for conflicts in the schedule.
6. The system shall save new schedule
7. Use case ends when the system adjusts accepts the data provided by the user.

· Post-conditions:

1. The system shall record the changes made.
2. The user can login during the new scheduled times.
3. The user has access to custom features during scheduled class times and exam week

Alternative Courses of Action

1. In step 2 of the description section if any of the required fields are blank the system shall request the user to make an entry in the appropriate field.
2. In step 3 of the description section the user has the option to cancel the presentation setup.

Exceptions

1. The Edit Schedule button in the home menu is not active.

2. The OK button in the edit schedule menu is not active.

Related Use Cases:

1. Schedule setup
 2. Schedule conflicts
 3. Alarms
 4. End of semester clear
-

Decision Support

Frequency: The professor may implement this use case zero or more times per semester.

Criticality: Medium. This use case allows the user to change original schedule if needed but may not be necessary

Risk: Medium. This use case is related to security tasks due to unauthorized changes in schedule.

Constraints:

1. Usability:
 - a) No previous training time
 - b) Professor Schedule Manager should provide 1 help frame on the topic.
 - c) On average the user should take up to 1 min to complete the edit schedule form.
2. Reliability:
 - a) Mean time to Failure –

b) Availability –

3. Performance:

a) Request should be sent and saved within 5 secs.

b) System should be able to handle 1 edit schedule request at a time per user.

4. Supportability:

a) Since the program will be coded on Java, any system with the Java Virtual Machine should execute the use case correctly.

Modification History

Owner: CEN4010_team2sp08

Initiation date: 01/27/2008

Date last modified: 02/16/2008

Use Case ID: PSM_014- End of Semester Schedule Clear

Details:

Actor: System Itself, FIU Professor

Pre-conditions:

1. The user shall start computer
2. The user shall start program
3. The user shall input Schedule
4. The user shall input last date schedule applies.

Description:

- a. Use case begins when the system provides with a template to input first schedule
- b. The user shall input a last day for his/her schedule to take effect
- c. The system shall compare the data with other professor last days. The later day will get the priority
- d. Use case ends when the system deletes all the information saved during the semester the latest date inputted.

Trigger:

- i. The user inputs the last day of his/her class schedule
- ii. The system responds by comparing to other data inputted if any and by setting the latest day for the semester in order to delete information.

Post-conditions:

1. The system finishes the deletion of the data. The system acts as first time running.

Alternative Courses of Action

1. The user has the option to modify last day of schedule to take effect. The system compares again the latest day to set a date to erase data.

Exceptions:

1. The user input schedule as a presenter not as professor, therefore the system does not save data for user.

Related Use Cases:

1. Misuse case- user inputted wrong last day of schedule.

2. Login
 3. Set-up schedule during initial log in
-

Decision Support

Frequency: Once per semester which equals once per 15 weeks

Criticality: Medium. This case allows system to clean up and for entire to computer to alleviate memory usage.

Risk: Low. This use case only will delete information from program data base, therefore no critical information will be deleted.

Constraints:

1. Usability: On average the user should take less than 1 minute setting up schedule the only field needed for the use case to take effect
 2. Reliability: Mean time to failure- 5% failure is acceptable.
 3. Performance: System should be able to perform deletion without interfering user.
 4. Supportability: System operated under Windows Platform only
 5. Implementation: Use case implemented by system only
-

Modification History

Owner: CEN4010_team2sp08

Initiation date: 01/24/2008

Date last modified: 02/16/2008

Use Case ID: PSM_017 - Password Conflicts

Details:

Actor: System Itself, FIU Professor

Pre-conditions:

1. The user shall start computer
2. The user shall start program
3. The user shall input login
4. The user shall accept input

Description:

1. Use case begins when the system provides with a template to input login
2. The user shall input the password
3. The system shall verify every field of inputted data is correct
4. The system shall give an error to user letting know the password is not correct or does not matches the system.
5. The system shall enable the use of password problem button
6. The system shall open a new page letting user answer a question in order to reset password
7. The user shall input the new created password into the password field of the logins screen
8. Use case ends when the system validates information and let user keep to next step.

Trigger:

- The user inputs incorrect password
- The system responds by verifying the field and giving an error letting the user to open a password conflict screen.

Post-conditions:

The system takes user to new screen in order to answer question to reset password.

Alternative Courses of Action

1. The user input wrong data. The user realizes the error and inputs the password again without having to open the password conflict screen
2. The user exits program. Data inputted gets deleted by default.

Exceptions:

1. The user inputs wrong password more than three times. The system lets user know that his access is temporarily denied. The system lets user try again after 2 minutes

Related Use Cases:

1. Misuse case- user inputs wrong password several times.
2. Editing data
3. Security case- keeping data private

Decision Support

Frequency: 2 times per week

Criticality: High. This use case allows user to utilize system by resetting password and creating a new one.

Risk: Medium. This use case verifies the validity of user if password still conflicting locks system for a period of time

Constraints:

1. Usability: On average the user should take less than 2 minutes answering security question and less than 5 minutes creating a new password

2. Reliability: Mean time to failure-less than 10% is acceptable.
 - 2.The use case depends on user to input incorrect data.
 3. Performance: System should be able to handle 1 request per user.
 4. Supportability: System operated under Windows Platform only
 1. Implementation: Use case depends on user to input incorrect password in order for this use case to be implemented
-

Modification History

Owner: CEN4010_team2sp08

Initiation date: 01/24/2008

Date last modified: 02/16/2008

10.3. Appendix C

Example of well documented test drivers, stubs used during unit and subsystem testing.

```
/*
    * ID: Unit047_DBConnection007 - DBConnection.Java - fetchStartMon(int
    * courseID)
    * Purpose: To verify that DBConnection.fetchStartMon(int courseID)
    * returns the proper data state variable when it is called
    * Test Setup: Connection is mocked
    *             Statement is mocked
    *             DBConnection object is initialized
    *             Mockito.When()thenReturn mocks are properly implemented
    *
    * Test Input: db.fetchStartMon(907);
```

```

        * Expected Output: "Start Monday"
        * Actual Output: "Start Monday"
        * Test Passed
        */
@Test
public void unit047_DBConnection007() throws Exception
{
    Mockito.when(myCon.createStatement()).thenReturn(s);
    Mockito.when(s.executeQuery(Mockito.anyString())).thenReturn(res);
    Mockito.when(s.getResultSet()).thenReturn(res);
    Mockito.when(res.getString(Mockito.anyString())).thenReturn("Start Monday");

    assertEquals("Start Monday", db.fetchStartMon(907));

    Mockito.verify(res).getString(Mockito.anyString());
}

```

```

/*
 * ID: Unit091_DBConnection051 - DBConnection.Java - fetchCourses()
 * Purpose: To verify that DBConnection.getEndDates()
 * returns the proper data state variable when it is called
 * Test Setup: Connection is mocked
 *             Statement is mocked
 *             DBConnection object is initialized
 *
 *             Mockito.when(myCon.createStatement()).thenReturn(s);
 *             Mockito.when(s.executeQuery(Mockito.anyString())).thenReturn(res);
 *             Mockito.when(s.getResultSet()).thenReturn(res);
 *
 *             Mockito.when(res.next()).thenReturn(true,false);
 *             Mockito.when(res.Int(Mockito.anyString())).thenReturn(1);
 *
 *             String result initialized to "1, "
 *
 */

```

```

        * Test Input: db.getEndDates();
        * Expected Output: String containing "1,"
        * Actual Output: String containing "1,"
        * Test Passed
        */
@Test //fetchCourses()
public void unit091_DBConnection51() throws Exception
{
    Mockito.when(myCon.createStatement()).thenReturn(s);
    Mockito.when(s.executeQuery(Mockito.anyString())).thenReturn(res);
    Mockito.when(s.getResultSet()).thenReturn(res);

    Mockito.when(res.next()).thenReturn(true,false);

    Mockito.when(res.getInt(Mockito.anyString())).thenReturn(1);

    String result = "1,";

    assertEquals(result, db.fetchCourses());
}

```

```

/* ID: AppCont042 - appController.java - sleep()
 * Purpose: To test that sleep() catches an interrupted thread exception.
 *
 * Test Setup:
 *         appController instantiated
 *         PowerMockito.mockStatic(Logger.class);
 *
 * Test Input:
 *         appController.sleep(100);
 *
 * Expected Output:
 *         Verify that the Logger mock was called in the catch block.

```

```

*/
@Test
public void Unit090_AppCont042()
{
    //Preconditions
    Thread.currentThread().interrupt();
    PowerMockito.mockStatic(Logger.class);

    //Test input
    appController.sleep(100);
    //Assertions
    PowerMockito.verifyStatic(Logger.class);
}

```

```

/* ID: AppCont050 - appController.java - checkTimes()
* Purpose: To test that checkTimes() parses the time for Saturday
*         correctly when the date and class is on Saturday.
*
* Test Setup:
*         appController instantiated
*         ArrayList<Integer> mockCourseList;
*         mockCourseList.add(1);
*         Set the Saturday end time to 1:15.
*         Set each other class end time to "".
*         Mock the GregorianCalendar class and set the day
*         to Saturday.
*
* Test Input:
*         appController.checkTimes()
*
* Expected Output:
*         The time returned from appController should be 1:15.
*/
@Test

```

```

public void Unit098_AppCont050() throws Exception
{
    //Preconditions
    ArrayList<Integer> mockCourseList = new ArrayList<Integer>();
    mockCourseList.add(1);

    Mockito.when(db.getCourses()).thenReturn(mockCourseList);

    //set internal variables for all days
    Mockito.when(db.fetchEndMon(1)).thenReturn("");
    Mockito.when(db.fetchEndTue(1)).thenReturn("");
    Mockito.when(db.fetchEndWed(1)).thenReturn("");
    Mockito.when(db.fetchEndThu(1)).thenReturn("");
    Mockito.when(db.fetchEndFri(1)).thenReturn("");
    Mockito.when(db.fetchEndSat(1)).thenReturn("01:15");
    GregorianCalendar gc = new GregorianCalendar();
    gc.set(2019, 4, 18); //this sets currentDay to 6

    PowerMockito.whenNew(GregorianCalendar.class).withAnyArguments().thenReturn(gc);

    //Test Input
    appController.checkTimes();

    //Assertions
    int hr = appController.returnHr();
    int min = appController.returnMin();

    assertEquals(1, hr);
    assertEquals(15, min);
}

```

```

/*
 * Test ID: SubSys23 - SubSysTest.java - whileForCourseSelectedRunner
 *
 * Purpose: Verify that when the method whileForCourseSelectedRunner() is called
 *          check that the method app1.CourseSelectedRunner() is
 *          called at least once
 *
 * Test Setup: appController instantiated
 *              Mockito.doReturn(true).when(app1).CourseSelectedRunner();
 *
 * Test Input: app1.whileForCourseSelectedRunner();
 *
 * Test Output: Call to the method should return true and app1.CourseSelectedRunner()
 *              should be called at least once
 */
@Test
public void SubSys23()
{
    appController app1 = Mockito.spy(app);
    Mockito.doReturn(true).when(app1).CourseSelectedRunner();
    app1.whileForCourseSelectedRunner(false);
    Mockito.verify(app1, Mockito.atLeastOnce()).CourseSelectedRunner();
}

/*
 * Test ID: SubSys16 - SubSysTest.java - InitSetupSelectedRunner
 *
 * Purpose: Verify that when the method InitSetupSelectedRunner() is called
 *          this one will return true for the setup of the selected schedule
 *
 * Test Setup: appController instantiated
 *              boolean bool;
 *              assertTrue(bool);|
 *
 * Test Input: bool = app1.InitSetupSelectedRunner();
 *
 * Test Output: Call to the method should return true
 */
@Test
public void SubSys16()
{
    appController app1 = Mockito.spy(app);
    boolean bool = app1.InitSetupSelectedRunner();
    assertTrue(bool);
    Mockito.verify(app1).InitSetupSelectedRunner();
}

```

10.4. Appendix D

The following subsection contains screen shots for code coverage achieved for the specification-based testing and implementation-based for each level of testing e.g., unit, subsystem and system, and for each tool.

Unit Testing





EclEmma 1.5.3 Statement Coverage

Original Coverage:













Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
▼ PSM	 59.7 %	6406	4316	10722
▼ src	 59.7 %	6406	4316	10722
> Default	 45.5 %	1351	1621	2972
> my.PSM.PSM_Interface	 76.9 %	4393	1320	5713
▼ my.PSM.PSM_Storage	 9.3 %	85	828	913
> DBConnection.java	 9.3 %	85	828	913
▼ my.PSM.PSM_Logic	 51.3 %	577	547	1124
> appController.java	 54.2 %	526	444	970
> InterfaceController.java	 37.1 %	33	56	89
> Authenticate.java	 0.0 %	0	38	38
> FutureTimer.java	 78.3 %	18	5	23
> IdleLogoutTestClass.java	 0.0 %	0	4	4

- NOTE: In this version of EclEmma, branch coverage had not yet been implemented.

ECL EMMA 5.3 Updated Branch Coverage

Element	Coverage	Covered Branch...	Missed Branch...	Total Branches
▼ PSM	 4.9 %	8	154	162
▼ src	 4.9 %	8	154	162
▼ my.PSM.PSM_Storage	 100.0 %	8	0	8
> DBConnection.java	 100.0 %	8	0	8

Updated Coverage:

Test Sessions Coverage Boolean Analyzer Pick Test Cases Correlation Problems Console Coverage					
MasterDriver (Apr 19, 2019 2:33:05 PM)					
Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...	
▼ PSM	 84.5 %	10799	1983	12782	
▼ src	 84.5 %	10799	1983	12782	
> my.PSM.PSM_Interface	 76.3 %	4360	1353	5713	
> Default	 91.1 %	4596	448	5044	
▼ my.PSM.PSM_Logic	 88.8 %	988	124	1112	
> InterfaceController.java	 37.1 %	33	56	89	
> Authenticate.java	 0.0 %	0	38	38	
> appController.java	 97.8 %	937	21	958	
> FutureTimer.java	 78.3 %	18	5	23	
> IdleLogoutTestClass.java	 0.0 %	0	4	4	
▼ my.PSM.PSM_Storage	 93.6 %	855	58	913	
> DBConnection.java	 93.6 %	855	58	913	

- NOTE: In this version of Eclemma, branch coverage had not yet been implemented.

CodeCover Statement and Branch Coverage

Original Coverage:

<div> Test Sessions Coverage Boolean Analyzer Pick Test Cases Correlation Problems Console Coverage </div>						
<input type="checkbox"/> Show methods with <div> Statement Coverage <= 90.5 % </div>						
Name	Statement	Branch	Loop	Term	?-Operator	Synchronized
PSM	6.7 %	3.4 %	?	?	?	?
my	6.7 %	3.4 %	?	?	?	?
PSM	6.7 %	3.4 %	?	?	?	?
PSM Storage	6.7 %	3.4 %	?	?	?	?
> DBConnection	6.7 %	3.4 %	?	?	?	?

Updated Coverage:

<div> Test Sessions Coverage Boolean Analyzer Pick Test Cases Correlation Problems Console Coverage </div>						
<input type="checkbox"/> Show methods with <div> Statement Coverage <= 90.5 % </div>						
Name	Statement	Branch	Loop	Term	?-Operator	Synchronized
PSM	95.9 %	89.7 %	?	?	?	?
my	95.9 %	89.7 %	?	?	?	?
PSM	95.9 %	89.7 %	?	?	?	?
PSM Storage	95.9 %	89.7 %	?	?	?	?
> DBConnection	95.9 %	89.7 %	?	?	?	?

Clover

Original Coverage:

Problems Console Coverage Coverage Explorer Test Run Explorer Test Contributions Clover Dashboard Ant

Show: Application classes

Elem	Cov St%	Cov Br%	Cov%
PSM	<div><div></div></div> 37.5%	<div><div></div></div> 11.2%	<div><div></div></div> 32.1%
> Default	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%
> my.PSM.PSM_Interface	<div><div></div></div> 59.7%	<div><div></div></div> 0.0%	<div><div></div></div> 50.1%
> my.PSM.PSM_Logic	<div><div></div></div> 42.0%	<div><div></div></div> 24.2%	<div><div></div></div> 39.6%
> appController.java	<div><div></div></div> 48.2%	<div><div></div></div> 26.8%	<div><div></div></div> 46.4%
> Authenticate.java	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%
> FutureTimer.java	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%
> IdleLogoutTestClass.java	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%
> InterfaceController.java	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%	<div><div></div></div> 5.3%
> my.PSM.PSM_Storage	<div><div></div></div> 6.7%	<div><div></div></div> 0.0%	<div><div></div></div> 8.0%
> DBConnection.java	<div><div></div></div> 6.7%	<div><div></div></div> 0.0%	<div><div></div></div> 8.0%

Updated Coverage:

Problems Console Coverage Coverage Explorer Test Run Explorer Test Contributions Clover Dashboard Ant







Show: Application classes

Elem	Cov St%	Cov Br%	Cov%
PSM	<div><div></div></div> 74.5%	<div><div></div></div> 79.5%	<div><div></div></div> 68.1%
> Default	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%
> my.PSM.PSM_Interface	<div><div></div></div> 63.2%	<div><div></div></div> 0.0%	<div><div></div></div> 53.2%
my.PSM.PSM_Logic	<div><div></div></div> 85.7%	<div><div></div></div> 87.1%	<div><div></div></div> 81.9%
> appController.java	<div><div></div></div> 95.8%	<div><div></div></div> 94.6%	<div><div></div></div> 93.8%
> Authenticate.java	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%
> FutureTimer.java	<div><div></div></div> 83.3%	<div><div></div></div> 50.0%	<div><div></div></div> 77.8%
> IdleLogoutTestClass.java	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%
> InterfaceController.java	<div><div></div></div> 0.0%	<div><div></div></div> 0.0%	<div><div></div></div> 5.3%
my.PSM.PSM_Storage	<div><div></div></div> 97.3%	<div><div></div></div> 100.0%	<div><div></div></div> 97.7%
> DBConnection.java	<div><div></div></div> 97.3%	<div><div></div></div> 100.0%	<div><div></div></div> 97.7%







Subsystem Testing

Original Coverage
















EclEmma 3.1.2 Statement Coverage

Element	Coverage	Covered Lines	Missed Lines	Total Lines
▼ PSMTEST	 55.2 %	1,076	874	1,950
▼ src	 55.2 %	1,076	874	1,950
my.PSM.PSM_Interface	 70.7 %	916	379	1,295
my.PSM.PSM_Logic	 16.2 %	49	254	303
my.PSM.PSM_Storage	 1.6 %	4	239	243
my.PSM.PSM_Test	 98.2 %	107	2	109

EclEmma 3.1.2 Branch Coverage







Element	Coverage	Covered Branches	Missed Branches	Total Branches
▼ PSMTEST	 0.0 %	0	92	92
▼ src	 0.0 %	0	92	92
my.PSM.PSM_Logic	 0.0 %	0	76	76
my.PSM.PSM_Interface	 0.0 %	0	8	8
my.PSM.PSM_Storage	 0.0 %	0	8	8
my.PSM.PSM_Test	 0.0 %	0	0	0

Clover Statement and Branch Coverage

Elem	Cov% ^	Cov St%	Cov Br%	Av Me Cpx	Cpx
▼ PSMTEST	 35.2%	 41.2%	 0.0%	1.3	294.0
my.PSM.PSM_Test	 96.1%	 96.3%	 0.0%	1.0	21.0
my.PSM.PSM_Interface	 53.8%	 63.5%	 0.0%	1.0	132.0
my.PSM.PSM_Logic	 0.9%	 0.5%	 0.0%	2.0	80.0
my.PSM.PSM_Storage	 0.4%	 0.0%	 0.0%	2.1	61.0

Updated Coverage

EclEmma 3.1.2 Statement Coverage

Element	Coverage	Covered Lines	Missed Lines	Total Lines
▼ PSMTEST2	 60.7 %	1,277	827	2,104
▼ src	 60.7 %	1,277	827	2,104
my.PSM.PSM_Interface	 73.2 %	950	347	1,297
my.PSM.PSM_Storage	 1.6 %	4	239	243
my.PSM.PSM_Logic	 42.9 %	159	212	371
my.PSM.PSM_Test	 85.0 %	164	29	193

EclEmma 3.1.2 Branch Coverage

Element	Coverage	Covered Branches	Missed Branches	Total Branches
▼ PSMTEST2	16.7 %	16	80	96
▼ src	16.7 %	16	80	96
▶ my.PSM.PSM_Logic	20.0 %	16	64	80
▶ my.PSM.PSM_Interface	0.0 %	0	8	8
▶ my.PSM.PSM_Storage	0.0 %	0	8	8
▶ my.PSM.PSM_Test		0	0	0

Clover Statement and Branch Coverage

Elem	Cov% ^	Cov St%	Cov Br%	Av Me Cpx	Cpx
▼ PSMTEST2	44.7%	49.2%	17.1%	1.3	343.0
▶ my.PSM.PSM_Interface	59.0%	68.3%	0.0%	1.0	132.0
▶ my.PSM.PSM_Logic	46.0%	47.6%	21.2%	1.4	150.0
▶ my.PSM.PSM_Storage	0.4%	0.0%	0.0%	2.1	61.0
▶ my.PSM.PSM_Test	-	-	-	-	-

System Testing

Original

Package ^	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	64	61% 1149/1882	26% 24/92	1.51
my_PSM.PSM_Interface	53	74% 999/1346	37% 3/8	1.031
my_PSM.PSM_Logic	10	40% 118/295	23% 18/76	2.024
my_PSM.PSM_Storage	1	13% 32/241	37% 3/8	2.75

NEW

Package ^	# Classes	Line Coverage	Branch Coverage	Complexity
All Packages	64	83% 1572/1882	54% 50/92	1.51
my_PSM.PSM_Interface	53	88% 1197/1346	87% 7/8	1.031
my_PSM.PSM_Logic	10	64% 189/295	50% 38/76	2.024
my_PSM.PSM_Storage	1	77% 186/241	62% 5/8	2.75

10.5. Appendix E

Diary of meeting and tasks.

Date Of Meeting	3/25/19
Location	Discord
Topics	Setting up Coverage
Present	Everyone

Date Of Meeting	3/30/19
Location	Discord
Topics	Documentation Deliverable
Present	Everyone

Date Of Meeting	4/2/19
Location	Discord
Topics	PSM Testing
Present	Everyone

Date Of Meeting	4/4/19
Location	Discord
Topics	PSM Testing
Present	Everyone

Date Of Meeting	4/9/19
Location	Discord
Topics	PSM Testing
Present	Everyone

Date Of Meeting	4/11/19
Location	Discord
Topics	PSM Testing
Present	Everyone

Date Of Meeting	4/14/19
-----------------	---------

Location	CASE Lab
Topics	Presentation
Present	Everyone

Date Of Meeting	4/15/19
Location	CASE Lab
Topics	Presentation
Present	Everyone

Date Of Meeting	4/18/19
Location	Discord
Topics	Deliverable 2 Report
Present	Everyone