Specification-Based Test Document

CEN4072 – Fundamentals of Software Testing
Prepared for Dr. Peter Clarke
By Team 1

M. Kian Maroofi
Alexander Jimenez
Matt Taylor
Kristian Perez
Nicholas Delamo

February 29, 2020

Abstract

Payroll Management System (POS) is a web-based system meant to provide employees and employers a mean to manage different aspects of payroll management, such as, time sheet management, salary calculations, etc. This document delineates the corpus of Team 1's (henceforth also referred to as, "we", and "us") Specification-Based Test Document project. We outline the features we test throughout the projects, the systems that these features involve, and the unit tests used to test the subsystems that compose said systems. We include specific test case input and logs of our activity, results, and problems we've encountered.

Table of Contents

1 Introduction

The following chapter introduces the Specification-Based Test Document (STD) with the goal of conducting a specification-based testing on all of the implemented features of the provided system (PMS or Payroll Management System).

The purpose of the PMS is defined below. Following that, the scope of the requirements of the system is defined in Section 1.2. Section 1.3 contains overall testing approach such as unit, subsystems, and system test. Finally Section 1.4 and 1.5 go through terminology (definitions, abbreviations, acronyms) and a brief explanation regarding this document's organization.

1.1 Overview of System

Payroll administration can be very simple, involving the payment of just a handful of employees, or very complicated, involving payroll for employees. In some, very small companies, payroll may have handled by the owner of the company or an employee. However, other companies may have many employees to pay and keep track of necessitating a well-planned, efficient payroll administration system. Payroll Management System is a simple web-based application and it is user- friendly.

The main purpose of Payroll Management System is to become a quick and easy to use web applications for users (Employee's/Employer's) that need to fill in the timesheets and the generating the pay slips. The process consists of calculation of salaries and tax deductions of the employees. The rate of tax deductions is pulled up from the taxfoundation.org where we can find the tax rates in different states. This application (P.M.S) is available on desktop platforms and it can be accessed through many browsers such as Google Chrome, Mozilla Firefox.

1.2 Requirements of the System

The system shall allow all registered employers and employees to login to the PMS (PMS_02_Login, see Appendix B).

The system shall allow all of the logged in users (employer & employee) to logout of their account (PMS_21_logout, see Appendix B).

The system shall allow registered employers to add employees to their company (PMS_13_AddEmployee, see Appendix B).

The system shall allow employers to modify their employees submitted time sheets (PMS_05_ModifyTS, see Appendix B).

The system shall allow employers to approve their employees submitted time sheets (PMS_05_ApproveTS, see Appendix B).

The system shall allow employers to calculate their employees salaries based on their submitted time sheets (PMS_07_CalcSal, see Appendix B).

The system shall allow employees to submit their saved time sheets (No written use case found in original final systems document of the PMS).

The system shall allow employees to enter their time sheets and save them in the data storage (PMS_08_SaveTS, see Appendix B).

The system shall allow users (employee & employer) to set security questions for their accounts (No clearly written use case found in original final systems document of the PMS).

The system shall allow employees to view their work profile including their role, contact details, etc. (No written use case found in original final systems document of the PMS).

1.3 Overall Testing Approach

Overall Testing approach used is based on Unified Software Development Process Model (Clarke). For unit testing, we implemented a new façade (**`model.modelFacade.java)`** to be tested with **`controller.Registration.java`** using a unit test driver. Unit tests are done using JUnit in Eclipse IDE. Subsystem Tests goes through all of the implemented methods in the **`model.modelFacade.java`**. JUnit and Mockito are used within the Eclipse IDE to perform all of the subsystem tests. System Tests, were done by creating different test case scenarios (total of 60, including 3 sunny day, and 3 rainy day test cases for each use case implemented). Selenium IDE and JUnit were used to perform all of the System Tests.

1.4 Terminology

Error! Reference source not found., contains a series of terms and acronyms used through this document. A more elaborate glossary can also be found in Section 9 of this document.

Term	***Meaning***
3TA	Three-Tier Architecture
API	Application Programming Interface
DB	Data Base (Data Storage)
PMS	Payroll Management System
FIU	Florida International University
FSD	Final Systems Document
N/A	Not Applicable
STD	Specification-Based Test Document
UML	Unified Modeling Language
USDP	Unified Software Design Process
V&V	Validation & Verification

1.5 Document Organization

This document is organized into 10 Sections. Section 1will Introduce the system and discusses requirements of the system, overall testing approach, terminology and the document organization. Section 2 goes over specification-based plan, which discusses team member roles, work breakdown, hardware & software requirements, test reference items, list of tested features as well as not-tested features. In section 3, we went over the unit tests. Section 4 and 5, describe the subsystem and system tests in order. Section 6 contains test summary report for all of the failures which occurred during different phases of testing, as well as solution, improvements, and notes on each one of them. Section 7 discusses risks involved with testing procedures. Section 8 is the approval page which contains every team members' signatures. Section 9 contains a full glossary, and finally section 10 contains all of the appendixes referred from other parts of this document.

2 Specification Test Plan

In order to conduct all of the specification-based tastings for the PMS project, we divided the team into different roles, which are mentioned in section 2.1 completely. All of the hardware & software requirements as well as referenced items during the tests are also collected in this section. Before conducting each of the testing tasks (unit, subsystems, and system tests), we implemented a new Façade called **`model.modelFacade.java`** to facilitate the testing process, especially for subsystem and unit tests.

2.1 Organization

Following (Table 2) contains all of the information regarding team members as well as their roles.

Member Name	Roles
M. Kian Maroofi	System Tester, Time Keeper
Alexander Jimenez	Java Developer
Matt Taylor	Unit Tester, Team Leader
Kristian Perez	Subsystem Tester
Nicholas Delamo	Subsystem Tester, Minute Taker

Table 1: The roles assigned to the team members.

2.2 Hardware & Software Requirements

The hardware and software materials needed to complete the project are captured in the following subsections.

2.2.1 Hardware Requirements

The testing environment is a network-enabled computer system with the following hardware requirements:

- Processor: Intel (R) Core (TM) i7-7700 CPU @ 3.60GHz
- Installed Memory (RAM): 16GB DDR4 SDRAM
- Storage: 512GB
- Network Adapter: Inter (R) Ethernet Connection (2) I219-LM

Each member has its own individual station. The details of these stations are not reported in this document.

2.2.2 Software Requirements

The testing environment has the following software applications:

- MySQL 8.0, which is used for a back-end data store server.
- Java JDK 1.8.0_221-b11, with the following external libraries:
- Eclipse EE IDE (Version: 2019-12 (4.14.0))
- Selenium IDE (For performing GUI system tests)
- JUnit 4.0
- Mockito

2.3 Test Reference Items

- Original project deliverables
- Existing source code
- Mockito, JUnit, and Selenium tutorials on STEM/CyLE
- Database Tables Setup (mentioned in each testing chapter)

2.4 Tested Features

- Login
- Logout
- Add Employee
- Submit Timesheet
- Save Timesheet
- Modify Timesheet
- Approve Timesheet
- Security Questions
- Calculate Salary
- View Profile

2.5 Features Not Tested

- Search Employee Timesheet (Not implemented)
- Pay Check
- Work Profile
- Duplicate Submission
- Reset Password (Security)

2.6 Work Breakdown

Refer to Appendix A.

3 Unit Testing

3.1 Unit Test Cases

3.1.1 Test Identification & Objective (Summary)

All implemented methods in the ModelFacade class were subject to testing during the unit testing phase.

The objectives of the following tests are to verify program behavior and also verify that each method behaves gracefully when given unexpected input. Each function is given at least one test with valid input and one test with invalid input (except for those functions that do not receive any input.)

3.1.2 Test Criteria & Procedures

Database tables setup for Unit Test Pre-conditions are as follows:

employees										
emp_id	first_name	last_name	gender	dob	job	phone	email	address	accno	bankname	joindate
1	Adam	Sandler	on	1990-01-10	Movie Star	0	asand@email.com	2121 NW 1st St	1	Bank of America	2020-01-01
2	Dave	Grohl	yes	1986-04-22	Rock Star	0	test@email.com	900 West St	2	CitiBank	2020-03-03

users									
emp_id	user_id	password	sec_que1	ans1	sec_que2	ans2	sec_que3	ans3	createDate
1	adam	adam	Favorite Color?	pink	First pet name?	adam	Favorite movie?	adam	2020-01-12
1	user1	user1	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2020-01-12

paymode	
normal_pay	extra_pay
10	15

employer	
username	password
user1	user1

emp_ts										
ets_id	emp_id	day	wdate	intime	lunch_out	lunch_in	outtime	total_hours	status	date1
1144	1	Monday	2020-01-01	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-01-14
1114	1	Tuesday	2020-01-02	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-01-14
1172	1	Wednesday	2020-01-03	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-01-14
1341	1	Thursday	2020-01-04	10:00:00	12:00:00	13:00:00	22:00:00	11	not approved	2020-01-14
1776	2	Thursday	2020-01-04	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-01-14
1800	1	Friday	2020-01-05	10:00:00	12:00:00	13:00:00	22:00:00	11	not approved	2020-01-14
1337	2	Friday	2020-02-02	10:00:00	12:00:00	13:00:00	22:00:00	11	not approved	2020-01-14

save_ts										
ets_id	emp_id	day	date1	intime	lunch_out	lunch_in	outtime	total_hours	status	createDate
1126	1	Monday	2020-03-04	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-03-07

3.1.3 Test Cases

Test Case ID	UnitTest-PMS-ModelFacadeTests
Purpose	Ensure each method exposed through ModelFacade's API functions as expected
Test Set Up	For each test, the PMS system must be set up and working. The database has been loaded with the data as specified in the test prerequisites.

ID	Input	Output
1	`TimeSheetaddTimeSheet("1", "1", "Monday", "2020-03-02", "09:00:00", "12:00:00", "13:00:00", "17:00:00");`	Time sheet should be added to the save_ts table
2	`TimeSheetaddTimeSheet("1", "1", "2020/22-aa", "2020-03-02", "09:00:00", "12 noon", "13:00:00", "5 pm");`	Method should return a failure due to invalid date and time formats
3	`TimeSheetaddTimeSheet("1", "1", "yeeeee", "2020-03-02", "09:00:00", "12:00:00", "13:00:00", "17:00:00");`	Method should return a failure due to invalid day name
4	`TimeSheetupdateTimeSheet("1144", "10:00:00","12:00:00","13:00:00","22:00:00", "11");`	Time sheet with the ID '1144' should be updated in database table to reflect new info
5	`TimeSheetupdateTimeSheet("1144", "10:00:00","aaaaaaa","6:00 pm","22:00:00", "yes");`	Method should return a failure due to invalid input data
6	`TimeSheetsubmitTimeSheet("1");`	Time sheet in save_ts for employee 1 should be deleted from save_ts and moved to emp_ts
7	`TimeSheetgetEmpTimeSheetNotApproved("1");`	Method should return 2 rows of non-approved time sheets for employee 1
8	`TimeSheetgetEmpTimeSheetNotApproved("aaaaaaaaa");`	Method should return a failure due to invalid input data
9	`TimeSheetgetTimeSheetApproved("1");`	Method should return 3 rows of approved time sheets for employee 2
10	`TimeSheetgetEmpTimeSheetNotApproved("aaaaaaaaa");`	Method should return a failure due to invalid input data

11	`TimeSheetgetTimeSheetNotApproved();`	Method should return 3 rows representing non-approved time sheets across all employees
12	`TimeSheetgetTimeSheetApprovedEmpIds();`	Method should return 2 rows representing employees with approved time sheets
13	`SalarycalculateSalary();`	Method should add 2 newly calculated rows to the salaries table in the database
14	`SalarygetEmpPays();`	Method should return the 2 existing rows of salaries for all employees
15	`SalarygetEmpPay("2");`	Method should return all salaries for employee 2 (in this case, only one row)
16	`SalaryaddPayMode(20, 40);`	Method should add a new row to the paymode table with the input values
17	`EmployeeaddEmployee("4", "Hunter", "Biden", "M", "1986-04-21", "Mailman", "3059032234", "test@email.com", "900 Walker Street", "1234567890", "Bank of America")`	Method should add a new employee with the given data to the employees table
18	`EmployeeaddEmployee("5", "Hunter", "Biden", "M", "aaaaaaaaaaaa", "Mailman", "3059032234", "test@email.com", "900 Walker Street", "1234567890", "Bank of America");`	Method should return a failure due to invalid date of birth
19	`EmployeeaddEmployee("5", "Hunter", "Biden", "aaaaaaa", "1986-04-21", "Mailman", "3059032234", "test@email.com", "900 Walker Street", "1234567890", "Bank of America");`	Method should return a failure due to invalid gender
20	`EmployeeaddEmployee("5", "Hunter", "Biden", "M", "1986-04-21", // invalid email, "Mailman", "3059032234", "aaaaaaaaaaaa", "900 Walker Street", "1234567890", "Bank of America");`	Method should return a failure due to invalid email
21	`EmployeechangePassword("1", "adam", "Favorite Color?", "pink", "First PEt Name?", "adam", "Favorite movie?", "adam", "adam", "swordfish")`	Method should update the employee's password in the employee table and return success.
22	`EmployeechangePassword("1", "adam", "Favorite Color?", "blue", "First PEt`	Method should return a failure due to incorrect security question answers

	Name?", "a", "Favorite movie?", "a", "adam", "swordfish");	
23	EmployeedeleteEmp("2");	Method should delete employee 2 (Dave Grohl) from the employees table
24	EmployeedeleteEmp("42");	Method should return a failure due to a non-existent employee ID
25	EmployeegetEmployee("2");	Method should return the database row containing employee 2
26	EmployeegetEmployee("42");	Method should return a failure due to a non-existent employee ID
27	EmployeegetAllEmployees();	Method should return 2 rows representing both employees in the employee table
28	EmployeeupdateEmployee("2", "Hunter", "Biden", "M", "1986-04-21", "Mailman", "3059032234", "test@email.com", "900 Walker Street", "1234567890", "Bank of America");	Method should update employee 2's data in the employees table to match the input data
29	EmployeeupdateEmployee("42", "Hunter", "Biden", "M", "1986-04-21", "Mailman", "3059032234", "test@email.com", "900 Walker Street", "1234567890", "Bank of America");	Method should return a failure, since there is no employee 42
30	EmployeeupdateEmployee("2", "Hunter", "Biden", "M", "aaaaaaaaaaa", "Mailman", "3059032234", "test@email.com", "900 Walker Street", "1234567890", "Bank of America");	Method should return a failure, since "aaaaaaaaa" is not a valid date
31	EmployeeupdateEmployee("2", "Hunter", "Biden", "M", "1986-04-21", "Mailman", "dhbskdvjbrgnbdv", "test@email.com", "900 Walker Street", "1234567890", "Bank of America");	Method should return a failure due to invalid phone number
32	EmployeeupdateEmployee("2", "Hunter", "Biden", "M", "1986-04-21", "Mailman", "3054035039", "sdjfhbsdljs", "900 Walker Street", "1234567890", "Bank of America");	Method should return a failure due to invalid email
33	Employerauthenticate("user1", "user1")	Method should return success

34	`Employerauthenticate("aaaaaaaaaaaaaaaaa", "bbbbbbbbbbbbbbb");`	Method should return a failure due to invalid credentials
35	`Userauthenticate("1", "adam", "adam");`	Method should return success
36	`Userauthenticate("1", "adam", "swordfish");`	Method should return a failure due to invalid credentials
37	`Security_Questionregisteremployee("3", "joe", "scallop123", "Favorite Color?", "pink", "First PEt Name?", "adam", "Favorite movie?", "adam");`	Method should register a new employee with the given data and security questions

4 Subsystem Testing

4.1 Subsystem Test Cases

4.1.1 Test Identification & Objective (Summary)

Tests were identified by taking units from the unit testing phase and testing them in sequence with multiple units in the same test.

4.1.2 Test Criteria & Procedures

All implemented methods in the ModelFacade class were subject to testing during the subsystem testing phase, with the added criteria that several units were tested in concert.

The objectives of the following tests are to verify program behavior and also verify that each method behaves gracefully when given unexpected input. Each function is given at least one test with valid input and one test with invalid input (except for those functions that do not receive any input.) The intent of the subsystem tests is to view how previously-tested units work in concert.

4.1.3 Test Cases

Test Case ID	SubSystem-PMS-TimeSheetTest
Purpose	Test the interactions between multiple time sheet methods (add time sheet, update time sheet, get approved time sheets, etc)
Test Set Up	Data base found in section 3.1.2

Input	Output
TimeSheetaddTimeSheet("1", "1", "Monday", "2020-03-02", "09:00:00", "12:00:00", "13:00:00", "17:00:00") TimeSheetupdateTimeSheet("1","10:00:00","12:00:00","13:00:00","22:00:00", "11") TimeSheetsubmitTimeSheet("1") TimeSheetgetEmpTimeSheetNotApproved("1")	Add a time sheet to Emp "1", update that time sheet then submit it. Return 3 Not Approved timesheets for Emp "1" (2 from DB and the 1 just added)
TimeSheetaddTimeSheet("4", "4", "Monday", "2020-03-02", "09:00:00", "12:00:00", "13:00:00", "17:00:00") TimeSheetaddTimeSheet("5", "5", "Monday", "2020-03-02", "09:00:00", "12:00:00", "13:00:00", "17:00:00") TimeSheetsubmitTimeSheet("4") TimeSheetsubmitTimeSheet("5") TimeSheetgetTimeSheetNotApproved()	Add two dummy timesheets for Emp "4" and "5". Return 5 Not Approved timesheets for all Emps (3 from Emp 1 and 2 from Emp 4 and 5)
TimeSheetgetTimeSheetApproved("1") TimeSheetGetTimeSheetApprovedEmpIds()	Return 3 Approved time sheets from Emp 1 (found in setup DB) Return 2 (Emp 1 and Emp 2)

Test Case ID	SubSystem-PMS-SalaryTest
Purpose	Test the interactions between multiple salary methods
Test Set Up	Data base found in section 3.1.2

Input	Output
SalarycalculateSalary();	Return 4, 2 from Setup DB, 2 from method
SalarygetEmpPays();	Return 4, 4 salaries in DB
SalarygetEmpPay("2");	Return 2, 1 from setup DB, 1 from calculate salary method

Test Case ID	SubSystem-PMS-EmployeeTest
Purpose	Test the interactions between multiple employee methods
Test Set Up	Data base found in section 3.1.2

Input	Output
EmployeechangePassword("1", "adam", "Favorite Color?", "pink", "First PEt Name?", "adam", "Favorite movie?" , "adam", "adam", "swordfish"); EmployeegetPassword("1", "adam", "Favorite Color?", "pink,"First PEt Name?", "adam", "Favorite movie?" , "adam");	Password of user adam changed to “swordfish”
EmployeeupdateEmployee("2","Hunter","Biden","M","1986-04-21","Mailman","3059032234","test@email.com", "900 Walker Street", "1234567890", "Bank of America"); EmployeedeleteEmp("2"); EmployeeupdateEmployee("2", "Hunter", "Biden", "M", "1986-04-21", "Mailman", "3059032234", "test@email.com", "900 Walker Street", "1234567890", "Bank of America")	Emp 2 is added, then deleted. Then when attempting to update Emp 2, it will fail as Emp 2 was just deleted
EmployeegetEmployee("1");	Return 1 Emp with ID 1
EmployeeaddEmployee("4","Hunter","Biden","M","1986-04-21","Mailman","3059032234","test@email.com", "900 Walker Street", "1234567890", "Bank of America"); EmployeegetAllEmployees();	Add dummy Emp 4, return all employees as 2, Emp 1 and Emp 4

5 System Testing

5.1 System Test Cases

5.1.1 Test Identification & Objective (Summary)

Following table contains all system test case summaries. Note that "XX" in in each identifier ending is indicating the order of sunny and rainy tests which is one of the following: 01, 02, 03.

Test Cases Summary	
Unique Identifier	Purpose of Test
SystemTest-PMS-Login-001-SunnyXX	Investigate the proper execution of login use case.
SystemTest-PMS-Login-001-RainyXX	Investigate the improper execution of login use case.
SystemTest-PMS-Logout-006-SunnyXX	Investigate proper execution of logout use case.
SystemTest-PMS-Logout-006-RainyXX	Investigate improper execution of logout use case.
SystemTest-PMS-AddEmployee-002-SunnyXX	Investigate the proper execution of add employee use case.
SystemTest-PMS-AddEmployee-002-RainyXX	Investigate the improper execution of add employee use case.
SystemTest-PMS-SaveTS-008-SunnyXX	Investigate the proper execution of save timesheet use case.
SystemTest-PMS-SaveTS-008-SunnyXX	Investigate the improper execution of save timesheet use case.
SystemTest-PMS-ModifyTS-004-SunnyXX	Investigate the proper execution of modify timesheet use case.
SystemTest-PMS-ModifyTS-004-RainyXX	Investigate the improper execution of modify timesheet use case.
SystemTest-PMS-SubmitTS-009-SunnyXX	Investigate the proper execution of submit timesheet use case.
SystemTest-PMS-SubmitTS-009-RainyXX	Investigate the improper execution of submit timesheet user case.
SystemTest-PMS-ApproveTS-010-SunnyXX	Investigate the proper execution of approve timesheet use case.
SystemTest-PMS-ApproveTS-010-RainyXX	Investigate the improper execution of approve timesheet use case.
SystemTest-PMS-SecurityQuestion-003-SunnyXX	Investigate the proper execution of security question use case.
SystemTest-PMS-SecurityQuestion-003-RainyXX	Investigate the improper execution of security question use case.
SystemTest-PMS-CalcSal-005-SunnyXX	Investigate the proper execution of calculate salary use case.
SystemTest-PMS-CalcSal-005-RainyXX	Investigate the improper execution of calculate salary use case.
SystemTest-PMS-ViewProfile-007-SunnyXX	Investigate the proper execution of view profile use case.
SystemTest-PMS-ViewProfile-007-RainyXX	Investigate the improper execution of view profile use case.

5.1.2 Test Criteria & Procedures

Database tables setup for System Tests Pre-conditions are as follows:

employer	
username	password
user1	user1
mcdlr	1234$

users									
emp_id	user_id	password	sec_que1	ans1	sec_que2	ans2	sec_que3	ans3	createDate
1	adam	adam	Favorite Color?	pink	First pet name?	adam	Favorite movie?	adam	2020-01-12
99	john	smith	Favorite Color?	*Red*	First Pet Name?	Ana	Favorite movie?	Matrix	2020-01-12
1	user1	user1	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2020-01-12
9	mcdlr	1234$	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	2020-02-12
98	sara	smith	Favorite Color?	*White*	First Pet Name?	Dana	Favorite movie?	Matrix	2020-01-12

employees											
emp_id	first_name	last_name	gender	dob	job	phone	email	address	accno	bankname	joindate
1	Adam	Sandler	on	1990-01-10	Movie Star	0	asand@email.com	2121 NW 1st St	1	Bank of America	2020-01-01
99	John	Smith	on	1996-01-01	SW Tester	8888888888	jsmith@email.com	21 SW 10th St	123	Bank of America	2020-01-01

emp_ts										
ets_id	emp_id	day	wdate	intime	lunch_out	lunch_in	outtime	total_hours	status	date1
1144	1	Monday	2020-01-01	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-01-14
1114	1	Tuesday	2020-01-02	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-01-14
1172	1	Wednesday	2020-01-03	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-01-14
1341	1	Thursday	2020-01-04	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-01-14
1800	1	Friday	2020-01-05	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-01-14
1999	1	Monday	2020-02-02	10:00:00	12:00:00	13:00:00	22:00:00	11	not approved	2020-02-02

paymode	
normal_pay	extra_pay
10	15

save_ts										
ets_id	emp_id	day	date1	intime	lunch_out	lunch_in	outtime	total_hours	status	createDate
1832	99	Monday	2020-02-24	10:00:00	11:00:00	12:00:00	22:00:00	11	not approved	2020-02-24
1833	99	Tuesday	2020-02-25	10:00:00	11:00:00	12:00:00	21:30:00	10.5	not approved	2020-02-25
1834	99	Wednesday	2020-02-26	10:00:00	11:00:00	11:00:00	21:00:00	11	not approved	2020-02-26

5.1.3 Test Cases

Following tables include all of the performed system test cases (total of 60).

Test Case ID	SystemTest-PMS-Login-001-Sunny01
Purpose	Investigate the execution of the login use case for John Smith(an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. John clicks on employee login on the main navigation bar. 2. He enters "1" as Employee ID. 3. Then he enters "john" as User Name. 4. Then he enters "smith" as Password. 5. Finally he clicks on the Login button in the bottom of the form.
Expected Output	The system completes the request without exceptions or errors. John would be finally logged in and being able to see the Payroll Management System Employee Module home page. Also there would be no modifications and/or updates to the database as a result of this action.

Test Case ID	SystemTest-PMS-Login-001-Sunny02
Purpose	Investigate the execution of the login use case for SFTalent Co. (an employer).
Test Set Up	The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. SFTalent Co. clicks on employer login on the main navigation bar. 2. Then he enters "user1" as User Name. 3. Then he enters "user1" as Password. 4. Finally he clicks on the Login button in the bottom of the form.
Expected Output	The system completes the request without exceptions or errors. SFTalent Co. would be finally logged in and being able to see the Payroll Management System Employer Module home page. Also there would be no modifications and/or updates to the database as a result of this action.

Test Case ID	SystemTest-PMS-Login-001-Sunny03
Purpose	Investigate the execution of the login use case for Miami Car Dealership (an employee).
Test Set Up	The PMS system is set up and working. Miami Car Dealership is using Firefox as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. Miami Car Dealership clicks on employer login on the main navigation bar. 2. Then he enters "mcdlr" as User Name. 3. Then he enters "1234$" as Password. 4. Finally he clicks on the Login button in the bottom of the form.
Expected Output	The system completes the request without exceptions or errors. Miami Car Dealership would be finally logged in and being able to see the Payroll Management System Employer Module home page. Also there would be no modifications and/or updates to the database as a result of this action.

Test Case ID	SystemTest-PMS-Login-001-Rainy01
Purpose	Investigate the proper execution of the login use case for John Smith (an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. John clicks on employee login on the main navigation bar. 2. He enters "1" as Employee ID. 3. Then he enters "smith" as Password. 4. Finally he clicks on the Login button in the bottom of the form.
Expected Output	The system cannot complete the request without exceptions or errors. John would see an error message displayed on the screen with the "Error : fail" message shown. This is because he left the User Name field of the login form blank. All three fields are required for a successful login request. Also there would be no modifications and/or updates to the database as a result of this action.

Test Case ID	SystemTest-PMS-Login-001-Rainy02
Purpose	Investigate the proper execution of the login use case for James Brown (an employee).
Test Set Up	The PMS system is set up and working. James Brown (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. James clicks on employee login on the main navigation bar. 2. He enters "624200" as Employee ID. 3. He enters "jbrown" as User Name. 4. Then he enters "Pa$sw0rd" as Password. 5. Finally he clicks on the Login button in the bottom of the form.
Expected Output	The system cannot complete the request without exceptions or errors. James would see an error message displayed on the screen with the "Error : fail" message shown. This is because he has not registered for an employee account yet or in other words, the database does not contain his information or neither credentials. Also there would be no modifications and/or updates to the database as a result of this action.

Test Case ID	SystemTest-PMS-Login-001-Rainy03
Purpose	Investigate the proper execution of the login use case for SFTalent Co. (an employer).
Test Set Up	The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. SFTalent Co. clicks on employer login on the main navigation bar. 2. Then he enters "user1" as User Name. 3. Then he enters "pass" as Password. 4. Finally he clicks on the Login button in the bottom of the form.
Expected Output	The system cannot complete the request without exceptions or errors. John would see an error message displayed on the screen with the "Error : fail" message shown. This is because they did not enter the correct password for the user1. Also there would be no modifications and/or updates to the database as a result of this action.

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-AddEmployee-002-Sunny01</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the add employee case for Sara Smith (an employee) for SFTalent Co. (the employer)</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. SFTalent Co. (an employer) is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. SFTalent Co. hovers on the employee option in the main navigation bar of the employer module.
2. Then they click on the Add Employee option in the drop down menu shown.
3. Then he enters “2” in Employee ID.
4. Then he enters “Sara” in First Name.
5. Then he enters “Smith” in Last Name.
6. He chooses Female in their gender.
7. He enters “1990-01-01” in the Date of Birth (YYYY-MM-DD).
8. He enters “Sales Assistant” in Job Role field.
9. He enters “3053458989” in their Contact.
10. He enters sara.smith@email.com as Email
11. He enters “3456 NW 10th Ave” in the Address.
12. He enters “1234567999” in Account Number.
13. He enters “Bank of America” in bank name.
14. Finally he clicks on the Add button in the bottom of the form.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. An alert message saying “Employee Details Added”. Following row(s) will be added to employees table in database.
<table><tr><th colspan="12">employees</th></tr><tr><td>emp_id</td><td>first_name</td><td>last_name</td><td>gender</td><td>dob</td><td>job</td><td>phone</td><td>email</td><td>address</td><td>accno</td><td>bankname</td><td>joindate</td></tr><tr><td>2</td><td>Sara</td><td>Smith</td><td>on</td><td>...</td><td>…</td><td>305…</td><td>…</td><td>…</td><td>…</td><td>Bank of …</td><td>2020…</td></tr></table></td></tr>
</table>

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-AddEmployee-002-Sunny02</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the add employee case for John Smith (an employee) for SFTalent Co. (the employer)</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. SFTalent Co. (an employer) is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. SFTalent Co. hovers on the employee option in the main navigation bar of the employer module.
2. Then they click on the Add Employee option in the drop down menu shown.
3. Then he enters “2” in Employee ID.
4. Then he enters “John” in First Name.
5. Then he enters “Smith” in Last Name.
6. He chooses Male in their gender.
7. He enters “1980-01-01” in the Date of Birth (YYYY-MM-DD).
8. He enters “Finance Manager” in Job Role field.
9. He enters “3050000000” in their Contact.
10. He enters john.smith@email.com as Email
11. He enters “3456 NW 10th Ave” in the Address.
12. He enters “1234567777” in Account Number.
13. He enters “Bank of America” in bank name.
14. Finally he clicks on the Add button in the bottom of the form.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. An alert message saying “Employee Details Added”. Following row(s) will be added to employees table in database. Note that even though emp_id = 2 was not a unique value and were used for Sara Smith as well, and that did not affected the employee registration process.
<table>
<tr><th colspan="12">employees</th></tr>
<tr><td>emp_id</td><td>first_name</td><td>last_name</td><td>gender</td><td>dob</td><td>job</td><td>phone</td><td>email</td><td>address</td><td>accno</td><td>bankname</td><td>joindate</td></tr>
<tr><td>2</td><td>John</td><td>Smith</td><td>on</td><td>...</td><td>…</td><td>305…</td><td>…</td><td>…</td><td>…</td><td>Bank of …</td><td>2020…</td></tr>
<tr><td>2</td><td>Sara</td><td>Smith</td><td>on</td><td>…</td><td>…</td><td>305…</td><td>….</td><td>…</td><td>…</td><td>Bank of …</td><td>2020…</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-AddEmployee-002-Sunny03</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the add employee case for Albert Lee Smith (an employee) for Miami Car Dealership (the employer)</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. Miami Car Dealership (an employer) is using Firefox as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. Miami Car Dealership. hovers on the employee option in the main navigation bar of the employer module.
2. Then they click on the Add Employee option in the dropdown menu shown.
3. Then he enters “3” in Employee ID.
4. Then he enters “Albert” in First Name.
5. Then he enters “Lee” in Last Name.
6. He enters “1996-11-05” in the Date of Birth (YYYY-MM-DD).
7. He enters “Finance Manager” in Job Role field.
8. He enters “8574567890” in their Contact.
9. He enters alber.lee@gmx.com as Email
10. He enters “009878678” in Account Number.
11. He enters “Wells Fargo” in bank name.
12. Finally he clicks on the Add button in the bottom of the form.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. An alert message saying “Employee Details Added”. Following row(s) will be added to employees table in database. Note that even though emp_id = 2 was not a unique value and were used for Sara Smith as well, and that did not affected the employee registration process. Also gender and address columns will be NULL since there were no input during registration.
<table>
<tr><th colspan="12">employees</th></tr>
<tr><td>emp_id</td><td>first_name</td><td>last_name</td><td>gender</td><td>dob</td><td>job</td><td>phone</td><td>email</td><td>address</td><td>accno</td><td>bankname</td><td>joindate</td></tr>
<tr><td>3</td><td>Albert</td><td>Lee</td><td>NULL</td><td>...</td><td>…</td><td>857…</td><td>…</td><td>NULL</td><td>…</td><td>Wells…</td><td>2020…</td></tr>
</table></td></tr>
</table>

Test Case ID	SystemTest-PMS-AddEmployee-002-Rainy01
Purpose	Investigate the improper execution of the add employee case for John Smith (an employee) for SFTalent Co. (the employer)
Test Set Up	The PMS system is set up and working. SFTalent Co. (an employer) is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. SFTalent Co. hovers on the employee option in the main navigation bar of the employer module. 2. Then they click on the Add Employee option in the drop down menu shown. 3. Then he enters "2" in Employee ID. 4. Then he enters "John" in First Name. 5. Then he enters "Smith" in Last Name. 6. He chooses Female in their gender. 7. He enters "1990-21-10" in the Date of Birth (not YYYY-MM-DD). 8. He enters "Sales Assistant" in Job Role field. 9. He enters "3050000000" in their Contact. 10. He enters john.smith@email.com as Email 11. He enters "3456 NW 10th Ave" in the Address. 12. He enters "1234567999" in Account Number. 13. He enters "Bank of America" in bank name. 14. Finally he clicks on the Add button in the bottom of the form.
Expected Output	The system cannot complete the request without exceptions or errors. SFTalent Co. would see an error message displayed on the screen with the "Error : Employee Registration Failed" message shown. This is because the date format entered in the Date of Birth field was not YYYY-MM-DD which is the proper required format. As a result of this action there would be no changes to database table entries.

Test Case ID	SystemTest-PMS-AddEmployee-002-Rainy02
Purpose	Investigate the improper execution of the add employee case for John Smith (an employee) for SFTalent Co. (the employer)
Test Set Up	The PMS system is set up and working. SFTalent Co. (an employer) is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. SFTalent Co. hovers on the employee option in the main navigation bar of the employer module. 2. Then they click on the Add Employee option in the drop down menu shown. 3. Then he enters "2" in Employee ID. 4. Then he enters "John" in First Name. 5. Then he enters "Smith" in Last Name. 6. He chooses Female in their gender. 7. He enters "1990-01-01" in the Date of Birth (YYYY-MM-DD). 8. He enters "Sales Assistant" in Job Role field. 9. He enters "305-000-0000" in their Contact. 10. He enters john.smith@email.com as Email 11. He enters "3456 NW 10th Ave" in the Address. 12. He enters "1234567999" in Account Number. 13. He enters "Bank of America" in bank name. 14. Finally he clicks on the Add button in the bottom of the form.
Expected Output	The system cannot complete the request without exceptions or errors. SFTalent Co. would see an error message displayed on the screen with the message that says please enter a number for Contact field. The format is supposed to be without the dashes for the phone number in the contact field. As a result of this action there would be no changes to database table entries.

Test Case ID	SystemTest-PMS-AddEmployee-002-Rainy03
Purpose	Investigate the improper execution of the add employee case for John Smith (an employee) for SFTalent Co. (the employer)
Test Set Up	The PMS system is set up and working. SFTalent Co. (an employer) is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. SFTalent Co. hovers on the employee option in the main navigation bar of the employer module. 2. Then they click on the Add Employee option in the drop down menu shown. 3. Then he enters "2" in Employee ID. 4. Then he enters "John" in First Name. 5. Then he enters "Smith" in Last Name. 6. He chooses Female in their gender. 7. He enters "1990-01-01" in the Date of Birth (YYYY-MM-DD). 8. He enters "305-000-0000" in their Contact. 9. He enters john.smith@email.com as Email 10. He enters "3456 NW 10th Ave" in the Address. 11. He enters "1234567999" in Account Number. 12. He enters "Bank of America" in bank name. 13. Finally he clicks on the Add button in the bottom of the form.
Expected Output	The system cannot complete the request without exceptions or errors. SFTalent Co. would see an error message displayed on the screen with the message that says "Fill out this field" for the job role's field. As a result of this action there would be no changes to database table entries.

Test Case ID	SystemTest-PMS-SecurityQuestion-003-Sunny03
Purpose	Investigate the proper execution of the security question security use case for Sara Smith (an employee).
Test Set Up	The PMS system is set up and working. Sara Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that the employee, Sara Smith is aware of their employee id and user name as well as a temporary password, in addition to security questions and answers.
Input	The following sequence is done: 1. Sara Smih first clicks on the employee login button on the main navigation bar on PMS homepage. 2. Then She clicks on the *Forgot Password?* Hyperlink. 3. She inputs "98" in the Employee ID field. 4. She then inputs "sara" in the User Name field. 5. She then chooses "Favorite Color?" as Security Question 1.

	6. She then inputs “White” in the answer for question 1. 7. She then chooses “First Pet name?” as Security Question 2. 8. She He then chooses “Favorite Movie?” as Security Question 3. 9. She then inputs “Matrix” in the answer for question 3.
Expected Output	The system completes the request without exceptions or errors. An alert message saying “Your password= smith”. As a result of this action there would be no updates and/or modifications to the database entries. As a result of this action there would be no changes to the database entries.

Test Case ID	SystemTest-PMS-SecurityQuestion-003-Sunny01
Purpose	Investigate the proper execution of the security question security use case for Adam Sandler (an employee).
Test Set Up	The PMS system is set up and working. Adam Sandler (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that the employee, Adam Sandler, is aware of their employee id and user name as well as a temporary password, in addition to security questions and answers.
Input	The following sequence is done: 1. Adam Sandler first clicks on the employee login button on the main navigation bar on PMS homepage. 2. Then he clicks on the *Forgot Password?* Hyperlink. 3. He inputs “1” in the Employee ID field. 4. He then inputs “adam” in the User Name field. 5. He then chooses “Favorite Color?” as Security Question 1. 6. He then inputs “pink” in the answer for question 1. 7. He then chooses “First Pet name?” as Security Question 2. 8. He then inputs “adam” in the answer for question 2. 9. He then chooses “Favorite Movie?” as Security Question 3. 10. He then inputs “adam” in the answer for question 3.
Expected Output	The system completes the request without exceptions or errors. An alert message saying “Your password= adam”. As a result of this action there would be no updates and/or modifications to the database entries. As a result of this action there would be no changes to the database entries.

Test Case ID	SystemTest-PMS-SecurityQuestion-003-Sunny02
Purpose	Investigate the proper execution of the security question security use case for John Smith (an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that the employee, John Smith, is aware of their employee id and user name as well as a temporary password, in addition to security questions and answers.
Input	The following sequence is done: 1. John Smith first clicks on the employee login button on the main navigation bar on PMS homepage. 2. Then he clicks on the *Forgot Password?* Hyperlink. 3. He inputs "99" in the Employee ID field. 4. He then inputs "john" in the User Name field. 5. He then chooses "Favorite Color?" as Security Question 1. 6. He then inputs "Red" in the answer for question 1. 7. He then chooses "First Pet name?" as Security Question 2. 8. He then inputs "Ana" in the answer for question 2. 9. He then chooses "Favorite Movie?" as Security Question 3. 10. He then inputs "Matrix" in the answer for question 3.
Expected Output	The system completes the request without exceptions or errors. An alert message saying "Your password= smith". As a result of this action there would be no updates and/or modifications to the database entries. As a result of this action there would be no changes to the database entries.

Test Case ID	SystemTest-PMS-SecurityQuestion-003-Rainy01
Purpose	Investigate the proper execution of the security question security use case for Adam Sandler (an employee).
Test Set Up	The PMS system is set up and working. Adam Sandler (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that the employee, Adam Sandler, is aware of their employee id and user name as well as a temporary password, in addition to security questions and answers.
Input	The following sequence is done: 1. Adam Sandler first clicks on the employee login button on the main navigation bar on PMS homepage. 2. Then he clicks on the *Forgot Password?* Hyperlink. 3. He inputs "1" in the Employee ID field. 4. He then inputs "adam" in the User Name field. 5. He then chooses "Favorite Color?" as Security Question 1. 6. He then inputs "pink" in the answer for question 1. 7. He then chooses "First Pet name?" as Security Question 2. 8. He then inputs "idk" in the answer for question 2. 9. He then chooses "Favorite Movie?" as Security Question 3. 10. He then inputs "adam" in the answer for question 3.
Expected Output	The system cannot complete the request without any exceptions or errors. An alert error will be displayed showing the message "user details not found". This happened due to the incorrect answer for second security question. As a result of this action there would be no changes to the database entries.

Test Case ID	SystemTest-PMS-SecurityQuestion-003-Rainy02
Purpose	Investigate the proper execution of the security question security use case for Adam Sandler (an employee).
Test Set Up	The PMS system is set up and working. Adam Sandler (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that the employee, Adam Sandler, is aware of their employee id and user name as well as a temporary password, in addition to security questions and answers.
Input	The following sequence is done: 1. Adam Sandler first clicks on the employee login button on the main navigation bar on PMS homepage. 2. Then he clicks on the *Forgot Password?* Hyperlink. 3. He inputs "1" in the Employee ID field. 4. He then inputs "adam" in the User Name field. 5. He then chooses "Favorite Color?" as Security Question 1. 6. He then inputs "pink" in the answer for question 1. 7. He then chooses "First Pet name?" as Security Question 2. 8. He then inputs "adam" in the answer for question 2. 9. He then chooses "First Pet name?" as Security Question 3. 10. He then inputs "adam" in the answer for question 3.
Expected Output	The system cannot complete the request without any exceptions or errors. An alert error will be displayed showing the message "user details not found". This happened due to the incorrect security question chosen for the third security question. As a result of this action there would be no changes to the database entries.

Test Case ID	SystemTest-PMS-SecurityQuestion-003-Rainy03
Purpose	Investigate the proper execution of the security question security use case for James Brown (an employee).
Test Set Up	The PMS system is set up and working. James Brown (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that the employee, James Brown, is aware of their employee id and user name as well as a temporary password, in addition to security questions and answers. Note that users table does not contain any related information or credentials related to James Brown.
Input	The following sequence is done: 1. James Brown first clicks on the employee login button on the main navigation bar on PMS homepage. 2. Then he clicks on the *Forgot Password?* Hyperlink. 3. He inputs "999" in the Employee ID field. 4. He then inputs "james" in the User Name field. 5. He then chooses "Favorite Color?" as Security Question 1. 6. He then inputs "yellow" in the answer for question 1. 7. He then chooses "First Pet name?" as Security Question 2. 8. He then inputs "max" in the answer for question 2. 9. He then chooses "First Pet name?" as Security Question 3. 10. He then inputs "dog" in the answer for question 3.
Expected Output	The system cannot complete the request without any exceptions or errors. An alert error will be displayed showing the message "user details not found". This happened due to lack of credentials for James Brown in the database entries. As a result of this action there would be no changes to the database entries.

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-ModifyTS-004-Sunny01</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the modify timesheet use case for SFTalent Co. (an employer) and Adam Sandler (an employee).</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. Note that it is assumed that the SFTalent Co. (the employer) is already logged in and viewing the homepage of PMS employer module.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. SFTalent Co. hovers on Time Sheets menu option on the main navigation bar for employer module.
2. He then clicks on View Time Sheets.
3. He then selects “1” as Employee ID.
4. He clicks on Get Details button next to the drop-down list.
5. He then modifies the in time column of the row indicated with TS_ID = 1800, from 10:00:00 to 09:00:00.
6. He also modifies the total hours entry for the same column to 12
7. Finally he clicks on Update Time Sheet button.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. SFTalent Co. would be finally modified the time sheet of Adam Sandler (an employee). An alert message saying “time sheet updated” will be shown. Also as a result of this action, the effected row of the emp_ts table in the database entries will be updated as follows.

<table>
<tr><th colspan="10">emp_ts</th></tr>
<tr><td>ets_id</td><td>emp_id</td><td>day</td><td>wdate</td><td>intime</td><td>lunch_out</td><td>lunch_in</td><td>outtime</td><td>total_hours</td><td>status</td><td>date1</td></tr>
<tr><td>1800</td><td>1</td><td>Friday</td><td>2020-01-05</td><td>09:00:00</td><td>12:00:00</td><td>13:00:00</td><td>22:00:00</td><td>12</td><td>approv.</td><td>2020-01-14</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-ModifyTS-004-Sunny02</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the modify timesheet use case for SFTalent Co. (an employer) and Adam Sandler (an employee).</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. Note that it is assumed that the SFTalent Co. (the employer) is already logged in and viewing the homepage of PMS employer module.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. SFTalent Co. hovers on Time Sheets menu option on the main navigation bar for employer module.
2. He then clicks on View Time Sheets.
3. He then selects “1” as Employee ID.
4. He clicks on Get Details button next to the drop-down list.
5. He then modifies the outtime column of the row indicated with TS_ID = 1172, from 22:00:00 to 18:00:00.
6. Finally he clicks on Update Time Sheet button.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. SFTalent Co. would be finally modified the time sheet of Adam Sandler (an employee). An alert message saying “time sheet updated” will be shown. Also as a result of this action, the effected row of the emp_ts table in the database entries will be updated as follows. Note that total hours column will not be automatically affected as a result of this action.

<table>
<tr><th colspan="11">emp_ts</th></tr>
<tr><td>ets_id</td><td>emp_id</td><td>day</td><td>wdate</td><td>intime</td><td>lunch_out</td><td>lunch_in</td><td>outtime</td><td>total_hours</td><td>status</td><td>date1</td></tr>
<tr><td>1172</td><td>1</td><td>Wednesday</td><td>2020-01-03</td><td>10:00:00</td><td>12:00:00</td><td>13:00:00</td><td>18:00:00</td><td>11</td><td>approv.</td><td>2020-01-14</td></tr>
</table>
</td></tr>
</table>

Test Case ID	SystemTest-PMS-ModifyTS-004-Rainy02
Purpose	Investigate the improper execution of the modify timesheet use case for SFTalent Co. (an employer) and Adam Sandler (an employee).
Test Set Up	The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. Note that it is assumed that the SFTalent Co. (the employer) is already logged in and viewing the homepage of PMS employer module.
Input	The following sequence is done: 1. SFTalent Co. hovers on Time Sheets menu option on the main navigation bar for employer module. 2. He then clicks on View Time Sheets. 3. He then selects "1" as Employee ID. 4. He clicks on Get Details button next to the drop-down list. 5. He then modifies the outtime column of the row indicated with TS_ID = 1800, from 22:00:00 to 9:00 PM. 6. He also clears the value of total hours entry for the same column so it becomes an emoty field. 7. Finally he clicks on Update Time Sheet button.
Expected Output	The system cannot complete the request without any exceptions or errors. SFTalent Co. would be shown an error message saying, "You have an error in your SQL syntax: …". This is because the total hours field must always be filled and cannot be left empty or NULL, even though other in and out time entries might be left empty. As a result of this action there would be no changes to the database entries.

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-CalcSal-005-Sunny01</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the calculate salary use case for SFTalent Co. (an employer) and Adam Sandler (an employee).</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. Note that it is assumed that the SFTalent Co. (the employer) is already logged in and viewing the homepage of PMS employer module.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. SFTalent Co. hovers on Salary menu option on the main navigation bar for employer module.
2. He then clicks on Calculate Pay.
3. He then clicks on Calculate button.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. SFTalent Co. finally calculated all salaries for the current time sheets for all of their employees including Adam Sandler. An alert message showing “Pay Calculations completed” will be shown. As a result of this action a row containing the following data entries will be added to the salaries table in the database.

<table>
<tr><th colspan="6">salaries</th></tr>
<tr><td>emp_id</td><td>total_hours</td><td>tax</td><td>gross_sal</td><td>net_sal</td><td>date1</td></tr>
<tr><td>1</td><td>53</td><td>159</td><td>530</td><td>371</td><td>2020-02-17</td></tr>
</table>
</td></tr>
</table>

Test Case ID	SystemTest-PMS-ModifyTS-004-Rainy01
Purpose	Investigate the improper execution of the modify timesheet use case for SFTalent Co. (an employer) and Adam Sandler (an employee).
Test Set Up	The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. Note that it is assumed that the SFTalent Co. (the employer) is already logged in and viewing the homepage of PMS employer module.
Input	The following sequence is done: 1. SFTalent Co. hovers on Time Sheets menu option on the main navigation bar for employer module. 2. He then clicks on View Time Sheets. 3. He then selects "1" as Employee ID. 4. He clicks on Get Details button next to the drop-down list. 5. He then modifies the outtime column of the row indicated with TS_ID = 1800, from 22:00:00 to 9:00 PM. 6. He also modifies the total hours entry for the same column to 11 7. Finally he clicks on Update Time Sheet button.
Expected Output	The system cannot complete the request without any exceptions or errors. SFTalent Co. would be shown an error message saying "Incorrect time value : 9:00 PM". This is due to the improper format of the time used. It should be 00:00:00 format with 24 hour system not the AM/PM format. As a result of this action there would be no changes to the database entries.

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-CalcSal-005-Sunny02</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the calculate salary use case for SFTalent Co. (an employer) and Adam Sandler (an employee).</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS Employer Module. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. Note that it is assumed that the SFTalent Co. (the employer) is already logged in and viewing the homepage of PMS employer module. Note that in this test case paymode table in the database does not have any entries yet.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. SFTalent Co. hovers on Salary menu option on the main navigation bar for employer module.
2. He then clicks on Pay Mode.
3. He then inputs “15” in Actual Pay field.
4. He then inputs “20” in Extra Time Pay field.
5. He again hovers on Salary option in menu bar.
6. He clicks on Calculate Pay this time.
7. He finally clicks on Calculate button.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. SFTalent Co. finally calculated all salaries for the current time sheets for all of their employees including Adam Sandler. An alert message showing “Pay Calculations completed” will be shown. As a result of this action a row containing the following data entries will be added to the salaries table in the database as well as paymode table.

<table>
<tr><th colspan="6">salaries</th></tr>
<tr><td>emp_id</td><td>total_hours</td><td>tax</td><td>gross_sal</td><td>net_sal</td><td>date1</td></tr>
<tr><td>1</td><td>53</td><td>238.5</td><td>795</td><td>556.5</td><td>2020-02-17</td></tr>
</table>

<table>
<tr><th colspan="2">paymode</th></tr>
<tr><td>normal_pay</td><td>extra_pay</td></tr>
<tr><td>15</td><td>20</td></tr>
</table>
</td></tr>
</table>

Test Case ID	SystemTest-PMS-Logout-006-Sunny03
Purpose	Investigate the proper execution of the logout use case for Miami Car Dealership (an employer).
Test Set Up	The PMS system is set up and working. Miami Car Dealership is using Firefox as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. Miami Car Dealership clicks on Logout button on the main navigation bar for PMS: Employer Module.
Expected Output	The system completes the request without exceptions or errors. Miami Car Dealership would be finally logged out and redirected to the PMS homepage. As a result of this action there would be no changes and/or modifications to the database.

Test Case ID	SystemTest-PMS-Logout-006-Rainy02
Purpose	Investigate the improper execution of the logout use case for John Smith (an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that John Smith (the employee) is already logged into the employee module for PMS. Note that server connection is broken after its established due to the server being down or bad network connectivity.
Input	The following sequence is done: 1. John clicks on Logout button on the main navigation bar for PMS: Employee Module.
Expected Output	The system cannot complete the request without exceptions or errors. John Smith would not be logged out. An alert error showing a message saying “Page load failed with error:…” may appear on the screen. As a result of this action there would be no changes and/or modifications to the database.

Test Case ID	SystemTest-PMS-Logout-006-Sunny01
Purpose	Investigate the proper execution of the logout use case for SFTalent Co. (an employer).
Test Set Up	The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done:

	1. SFTalent Co. clicks on Logout button on the main navigation bar for PMS: Employer Module.
Expected Output	The system completes the request without exceptions or errors. SFTalent Co. would be finally logged out and redirected to the PMS homepage. As a result of this action there would be no changes and/or modifications to the database.

Test Case ID	SystemTest-PMS-Logout-006-Rainy03
Purpose	Investigate the improper execution of the logout use case for Miami Car Dealership (an employer).
Test Set Up	The PMS system is set up and working. Miami Car Dealership is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. Miami Car Dealership clicks on Logout button on the main navigation bar for PMS: Employer Module.
Expected Output	The system cannot complete the request without exceptions or errors. Miami Car Dealership would not be logged out. An alert error showing a message saying "Page load failed with error:…" may appear on the screen. As a result of this action there would be no changes and/or modifications to the database.

Test Case ID	SystemTest-PMS-Logout-006-Sunny02
Purpose	Investigate the proper execution of the logout use case for John Smith(an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. John clicks on Logout button on the main navigation bar for PMS: Employee Module.
Expected Output	The system completes the request without exceptions or errors. John would be finally logged out and redirected to the PMS homepage. As a result of this action there would be no changes and/or modifications to the database.

Test Case ID	SystemTest-PMS-Logout-006-Rainy01
Purpose	Investigate the improper execution of the logout use case for SFTalent Co. (an employer).
Test Set Up	The PMS system is set up and working. SFTalent Co. is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. SFTalent Co. clicks on Logout button on the main navigation bar for PMS: Employer Module.
Expected Output	The system cannot complete the request without exceptions or errors. SFTalent Co. would not be logged out. An alert error showing a message saying "Page load failed with error:…" may appear on the screen. As a result of this action there would be no changes and/or modifications to the database.

Test Case ID	SystemTest-PMS-ViewProfile-007-Sunny02
Purpose	Investigate the proper execution of the view profile use case for Adam Sandler (an employee).
Test Set Up	The PMS system is set up and working. Adam Sandler (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. Adam clicks on View Pay Slips button on main navigation bar for PMS: Employee module.
Expected Output	The system completes the request without exceptions or errors. Adam would be finally looking at his employee profile details including his name, ID, Job Title, phone, as well as the recent pay slips. As a result of this action there would be no changes and/or modifications to the database.

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-SaveTS-008-Sunny01</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the save timesheet use case for John Smith(an employee).</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. John hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module.
2. Then he clicks on Add Time Sheets button in the displayed drop-down menu.
3. He picks “Monday” from the first row of the Day column picker.
4. Then he enters “2020-02-24” in the Date column. (YYYY-MM-DD)
5. He enters “10:00:00” in In time field.
6. He enters “11:00:00” in Lunch out field.
7. He enters “12:00:00” in Lunch in field.
8. He enters “22:00:00” in Check Out field.
9. He finally clicks save.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. Proper output would be a success message and saved values in the time sheet table, However due to faulty implementation of PMS redirect page for Time Sheet Control, an error display will be shown containing the “HTTP Status 500 – Internal Server Error”. But this does not affect the updates and modification to the database, hence the table save_ts in PMS Database will be updated with the following row:
<table>
<tr><th colspan="11">save_ts</th></tr>
<tr><td>ets_id</td><td>emp_id</td><td>day</td><td>date1</td><td>intime</td><td>lunch_out</td><td>lunch_in</td><td>outtime</td><td>total_hours</td><td>status</td><td>createDate</td></tr>
<tr><td>1832</td><td>99</td><td>Monday</td><td>2020-02-24</td><td>10:00:00</td><td>11:00:00</td><td>12:00:00</td><td>22:00:00</td><td>11</td><td>not approved</td><td>2020-02-24</td></tr>
</table>
</td></tr>
</table>

Test Case ID	SystemTest-PMS-ViewProfile-007-Sunny01
Purpose	Investigate the proper execution of the view profile use case for John Smith(an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. John clicks on View Pay Slips button on main navigation bar for PMS: Employee module.
Expected Output	The system completes the request without exceptions or errors. John would be finally looking at his employee profile details including his name, ID, Job Title, phone, as well as the recent pay slips. As a result of this action there would be no changes and/or modifications to the database.

Test Case ID	SystemTest-PMS-SaveTS-008-Sunny02
Purpose	Investigate the proper execution of the save timesheet use case for John Smith(an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. John hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module. 2. Then he clicks on Add Time Sheets button in the displayed drop-down menu. 3. He picks “Tuesday” from the second row of the Day column picker. 4. Then he enters “2020-02-25” in the Date column. (YYYY-MM-DD) 5. He enters “10:00:00” in In time field. 6. He enters “11:00:00” in Lunch out field. 7. He enters “12:00:00” in Lunch in field. 8. He enters “21:30:00” in Check Out field. 9. He finally clicks save.
Expected Output	The system completes the request without exceptions or errors. Proper output would be a success message and saved values in the time sheet table, however due to faulty implementation of PMS redirect page for Time Sheet Control, an error display will be shown containing the “HTTP Status 500 – Internal Server Error”. But this does not affect the updates and modification to the database, hence the table *save_ts* in PMS Database will be updated with the following row:

save_ts										
ets_id	emp_id	day	date1	intime	lunch_out	lunch_in	outtime	total_hours	status	createDate

	1833	99	Tuesday	2020-02-25	10:00:00	11:00:00	12:00:00	21:30:00	10.5	not approved	2020-02-25

Test Case ID	SystemTest-PMS-SaveTS-008-Sunny03
Purpose	Investigate the proper execution of the save timesheet use case for John Smith(an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. John hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module. 2. Then he clicks on Add Time Sheets button in the displayed drop-down menu. 3. He picks “Wednesday” from the third row of the Day column picker. 4. Then he enters “2020-02-26” in the Date column. (YYYY-MM-DD) 5. He enters “10:00:00” in In time field. 6. He enters “11:00:00” in Lunch out field. 7. He enters “11:00:00” in Lunch in field. 8. He enters “21:00:00” in Check Out field. 9. He finally clicks save.
Expected Output	The system completes the request without exceptions or errors. Proper output would be a success message and saved values in the time sheet table, however due to faulty implementation of PMS redirect page for Time Sheet Control, an error display will be shown containing the “HTTP Status 500 – Internal Server Error”. But this does not affect the updates and modification to the database, hence the table *save_ts* in PMS Database will be updated with the following row:

save_ts										
ets_id	emp_id	day	date1	intime	lunch_out	lunch_in	outtime	total_hours	status	createDate
1834	99	Wednesday	2020-02-26	10:00:00	11:00:00	11:00:00	21:00:00	11	not approved	2020-02-26

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-SaveTS-008-Rainy001</td></tr>
<tr><td>Purpose</td><td>Investigate the improper execution of the save timesheet use case for John Smith(an employee).</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database Tables users & employees contain the following. It is also assumed that John Smith (the employee) is already logged into the employee module for PMS

<table>
<tr><th colspan="11">employees</th></tr>
<tr><td>emp_id</td><td>first_name</td><td>last_name</td><td>gender</td><td>dob</td><td>job</td><td>phone</td><td>email</td><td>address</td><td>accno</td><td>bankname</td><td>joindate</td></tr>
<tr><td>99</td><td>John</td><td>Smith</td><td>on</td><td>19</td><td>…</td><td>0</td><td>a…</td><td>2121 …</td><td>1</td><td>Bank of A.</td><td>2020/…</td></tr>
</table>
<table>
<tr><th colspan="10">users</th></tr>
<tr><td>emp_id</td><td>user_id</td><td>password</td><td>sec_que1</td><td>ans1</td><td>sec_que2</td><td>ans2</td><td>sec_que3</td><td>ans3</td><td>createDate</td></tr>
<tr><td>99</td><td>john</td><td>smith</td><td>Favorite Color?</td><td>Red</td><td>First Pet Name?</td><td>Ana</td><td>Favorite movie?</td><td>Matrix</td><td>2020-01-12</td></tr>
</table>
</td></tr>
<tr><td>Input</td><td>The following sequence is done:

1. John hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module.
2. Then he clicks on Add Time Sheets button in the displayed drop-down menu.
3. He picks “Wednesday” from the third row of the Day column picker.
4. Then he enters “2020-26-02” in the Date column. (not YYYY-MM-DD)
5. He enters “10:00:00” in In time field.
6. He enters “11:00:00” in Lunch out field.
7. He enters “11:00:00” in Lunch in field.
8. He enters “21:00:00” in Check Out field.
9. He finally clicks save.</td></tr>
<tr><td>Expected Output</td><td>The system cannot complete the request without exceptions or errors. An error display will be shown containing the “HTTP Status 500 – Internal Server Error”. Due to the incorrect input format for the date column there would be no updates or modifications to the database.</td></tr>
</table>

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-SaveTS-008-Rainy002</td></tr>
<tr><td>Purpose</td><td>Investigate the improper execution of the save timesheet use case for John Smith(an employee).</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database Tables users & employees contain the following. It is also assumed that John Smith (the employee) is already logged into the employee module for PMS

<table>
<tr><th colspan="12">employees</th></tr>
<tr><td>emp_id</td><td>first_name</td><td>last_name</td><td>gender</td><td>dob</td><td>job</td><td>phone</td><td>email</td><td>address</td><td>accno</td><td>bankname</td><td>joindate</td></tr>
<tr><td>99</td><td>John</td><td>Smith</td><td>on</td><td>19</td><td>…</td><td>0</td><td>a…</td><td>2121 …</td><td>1</td><td>Bank of A.</td><td>2020/…</td></tr>
</table>
<table>
<tr><th colspan="10">users</th></tr>
<tr><td>emp_id</td><td>user_id</td><td>password</td><td>sec_que1</td><td>ans1</td><td>sec_que2</td><td>ans2</td><td>sec_que3</td><td>ans3</td><td>createDate</td></tr>
<tr><td>99</td><td>john</td><td>smith</td><td>Favorite Color?</td><td>Red</td><td>First Pet Name?</td><td>Ana</td><td>Favorite movie?</td><td>Matrix</td><td>2020-01-12</td></tr>
</table>
</td></tr>
<tr><td>Input</td><td>The following sequence is done:

1. John hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module.
2. Then he clicks on Add Time Sheets button in the displayed drop-down menu.
3. He picks “Wednesday” from the third row of the Day column picker.
4. Then he enters “2020-02-26” in the Date column. (not YYYY-MM-DD)
5. He enters “11:00:00” in Lunch out field.
6. He enters “11:00:00” in Lunch in field.
7. He enters “21:00:00” in Check Out field.
8. He finally clicks save.</td></tr>
<tr><td>Expected Output</td><td>The system cannot complete the request without exceptions or errors. An error display will be shown containing the “HTTP Status 500 – Internal Server Error”. Due to the empty value for In time field, there would be no changes, neither updates to the database.</td></tr>
</table>

Test Case ID	SystemTest-PMS-SaveTS-008-Rainy003
Purpose	Investigate the improper execution of the save timesheet use case for John Smith(an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. John hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module. 2. Then he clicks on Add Time Sheets button in the displayed drop-down menu. 3. He picks “Wednesday” from the third row of the Day column picker. 4. Then he enters “2020-02-26” in the Date column. (not YYYY-MM-DD) 5. He enters “10:00 AM” in In time field. 6. He enters “11:00 AM” in Lunch out field. 7. He enters “11:00 AM” in Lunch in field. 8. He enters “21:00 PM” in Check Out field. 9. He finally clicks save.
Expected Output	The system cannot complete the request without exceptions or errors. An error display will be shown containing the “HTTP Status 500 – Internal Server Error”. Time format to be saved in the database is HH:MM:SS with the 24 hour format instead of the AM/PM. So there would be no updates or modifications to the database.

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-SubmitTS-009-Sunny01</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the submit timesheet use case for John Smith(an employee).</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that John Smith (the employee) is already logged into the employee module for PMS.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. John hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module.
2. Then he clicks on Add Time Sheets button in the displayed drop-down menu.
3. He finally clicks submit button on the bottom of the form.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. An alert message saying “time sheet submitted” is displayed. As a result of this action, save_ts table in the database will clear all of the values related to employee’s specific saved time sheet, and emp_ts table will be updated with the following rows:

<table>
<tr><th colspan="11">emp_ts</th></tr>
<tr><td>ets_id</td><td>emp_id</td><td>day</td><td>wdate</td><td>intime</td><td>lunch_out</td><td>lunch_in</td><td>outtime</td><td>total_hours</td><td>status</td><td>createDate</td></tr>
<tr><td>1832</td><td>99</td><td>Monday</td><td>2020-02-24</td><td>10:00:00</td><td>11:00:00</td><td>12:00:00</td><td>22:00:00</td><td>11</td><td>not approved</td><td>2020-02-24</td></tr>
<tr><td>1833</td><td>99</td><td>Tuesday</td><td>2020-02-25</td><td>10:00:00</td><td>11:00:00</td><td>12:00:00</td><td>21:30:00</td><td>10.5</td><td>not approved</td><td>2020-02-25</td></tr>
<tr><td>1834</td><td>99</td><td>Wednesday</td><td>2020-02-26</td><td>10:00:00</td><td>11:00:00</td><td>11:00:00</td><td>21:00:00</td><td>11</td><td>not approved</td><td>2020-02-26</td></tr>
</table></td></tr>
</table>

Test Case ID	SystemTest-PMS-SubmitTS-009-Sunny02
Purpose	Investigate the proper execution of the submit timesheet use case for Adam Sandler(an employee).
Test Set Up	The PMS system is set up and working. Adam Sandler (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that Adam Sandler (the employee) is already logged into the employee module for PMS.
Input	The following sequence is done: 1. Adam hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module. 2. Then he clicks on Add Time Sheets button in the displayed drop-down menu. 3. He finally clicks submit button on the bottom of the form.
Expected Output	The system completes the request without exceptions or errors. An alert message saying "time sheet submitted" is displayed. As a result of this action, since the *save_ts* table in the database did not have any rows related to Adam's saved time sheets, there would be no updates to the emp_ts neither save_ts tables in the database.

Test Case ID	SystemTest-PMS-SubmitTS-009-Sunny03
Purpose	Investigate the proper execution of the submit timesheet use case for Sara Smith(an employee).
Test Set Up	The PMS system is set up and working. Sara Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that Sara Smith (the employee) is already logged into the employee module for PMS.
Input	The following sequence is done: 1. Sara hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module. 2. Then she clicks on Add Time Sheets button in the displayed drop-down menu. 3. She finally clicks submit button on the bottom of the form.
Expected Output	The system completes the request without exceptions or errors. An alert message saying "time sheet submitted" is displayed. As a result of this action, since the *save_ts* table in the database did not have any rows related to Sara's saved time sheets, there would be no updates to the emp_ts neither save_ts tables in the database.

Test Case ID	SystemTest-PMS-SubmitTS-009-Rainy01
Purpose	Investigate the improper execution of the submit timesheet use case for Adam Sandler(an employee).
Test Set Up	The PMS system is set up and working. Adam Sandler (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that Adam Sandler (the employee) is already logged into the employee module for PMS. Furthermore, database has connectivity issues in this case.
Input	The following sequence is done: 1. Adam hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module. 2. Then he clicks on Add Time Sheets button in the displayed drop-down menu. 3. He finally clicks submit button on the bottom of the form.
Expected Output	The system cannot complete the request without exceptions or errors. An error message would be shown indicating no connection status to the database. Hence, there would be no updates or modifications to the PMS database.

Test Case ID	SystemTest-PMS-SubmitTS-009-Rainy02
Purpose	Investigate the improper execution of the submit timesheet use case for John Smith(an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that John Smith (the employee) is already logged into the employee module for PMS. Furthermore, database has connectivity issues in this case.
Input	The following sequence is done: 1. John hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module. 2. Then he clicks on Add Time Sheets button in the displayed drop-down menu. 3. He finally clicks submit button on the bottom of the form.
Expected Output	The system cannot complete the request without exceptions or errors. An error message would be shown indicating no connection status to the database. Hence, there would be no updates or modifications to the PMS database.

Test Case ID	SystemTest-PMS-ViewProfile-007-Rainy01
Purpose	Investigate the improper execution of the view profile use case for Adam Sandler (an employee).
Test Set Up	The PMS system is set up and working. Adam Sandler (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. Furthermore, database has connection problems in this case.
Input	The following sequence is done: 1. Adam clicks on View Pay Slips button on main navigation bar for PMS: Employee module.
Expected Output	The system cannot complete the request without exceptions or errors. An error indicating the no connection status to the database would be shown. As a result of this action there would be no changes neither updates to the database.

Test Case ID	SystemTest-PMS-SubmitTS-009-Rainy03
Purpose	Investigate the improper execution of the submit timesheet use case for Sara Smith(an employee).
Test Set Up	The PMS system is set up and working. Sara Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that Sara Smith (the employee) is already logged into the employee module for PMS. Furthermore, database has connectivity issues in this case.
Input	The following sequence is done: 1. Sara hovers on Time Sheets menu option from the main navigation bar of PMS: Employee module. 2. Then she clicks on Add Time Sheets button in the displayed drop-down menu. 3. She finally clicks submit button on the bottom of the form.
Expected Output	The system cannot complete the request without exceptions or errors. An error message would be shown indicating no connection status to the database. Hence, there would be no updates or modifications to the PMS database.

Test Case ID	SystemTest-PMS-ViewProfile-007-Rainy02
Purpose	Investigate the improper execution of the view profile use case for John Smith (an employee).
Test Set Up	The PMS system is set up and working. John Smith (an employee) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. Furthermore, database has connection problems in this case.
Input	The following sequence is done: 1. John clicks on View Pay Slips button on main navigation bar for PMS: Employee module.
Expected Output	The system cannot complete the request without exceptions or errors. An error indicating the no connection status to the database would be shown. As a result of this action there would be no changes neither updates to the database.

Test Case ID	SystemTest-PMS-ViewProfile-007-Rainy03
Purpose	Investigate the improper execution of the view profile use case for Sara Smith (an employee).
Test Set Up	The PMS system is set up and working. Sara Smith (an employee) is using Chrome as their browser and she is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. Furthermore, database has connection problems in this case.
Input	The following sequence is done: 1. Sara clicks on View Pay Slips button on main navigation bar for PMS: Employee module.
Expected Output	The system cannot complete the request without exceptions or errors. An error indicating the no connection status to the database would be shown. As a result of this action there would be no changes neither updates to the database.

Test Case ID	SystemTest-PMS-ViewProfile-007-Sunny03
Purpose	Investigate the proper execution of the view profile use case for Sara Smith (an employee).
Test Set Up	The PMS system is set up and working. Sara Smith (an employee) is using Chrome as their browser and she is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document.
Input	The following sequence is done: 1. Sara clicks on View Pay Slips button on main navigation bar for PMS: Employee module.
Expected Output	The system completes the request without exceptions or errors. Sara would be finally looking at her employee profile details including his name, ID, Job Title, phone, as well as the recent pay slips. As a result of this action there would be no changes and/or modifications to the database.

Test Case ID	SystemTest-PMS-ApproveTS-010-Sunny01
Purpose	Investigate the proper execution of the approve time sheet use case for SFTalent Co. (an employer) and Adam Sandler (an employee).
Test Set Up	The PMS system is set up and working. SFTalent Co. (the employer) is using Chrome as their browser and she is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that the employer is already logged in their account.
Input	The following sequence is done: 1. SFTalent Co. (the employer) hovers on Time Sheets button in the PMS: Employer Module's Main navigation bar. 2. Then he clicks on Approve Time Sheet button in the opened drop-down menu. 3. He checks the approval column for the row with TS_ID = 1999. 4. He finally clicks on approve button.
Expected Output	The system completes the request without exceptions or errors. SFTalent Co. (the employer) have successfully approved all of the selected rows from the approval table. The approved rows must disappear from the displaying table view. The following row(s) of the emp_ts table in the database would be updated with the following:

emp_ts										
ets_id	emp_id	day	wdate	intime	lunch_out	lunch_in	outtime	total_hours	status	date1
1999	1	Monday	2020-02-02	10:00:00	12:00:00	13:00:00	22:00:00	11	approved	2020-02-02

<table>
<tr><td>Test Case ID</td><td>SystemTest-PMS-ApproveTS-010-Sunny02</td></tr>
<tr><td>Purpose</td><td>Investigate the proper execution of the approve time sheet use case for SFTalent Co. (an employer) and John Smith (an employee).</td></tr>
<tr><td>Test Set Up</td><td>The PMS system is set up and working. SFTalent Co. (the employer) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that the employer is already logged in their account.</td></tr>
<tr><td>Input</td><td>The following sequence is done:
1. SFTalent Co. (the employer) hovers on Time Sheets button in the PMS: Employer Module’s Main navigation bar.
2. Then he clicks on Approve Time Sheet button in the opened drop-down menu.
3. He checks the approval column for the row with TS_ID = 1834.
4. He checks the approval column for the row with TS_ID = 1833.
5. He finally clicks on approve button.</td></tr>
<tr><td>Expected Output</td><td>The system completes the request without exceptions or errors. SFTalent Co. (the employer) have successfully approved all of the selected rows from the approval table. The approved rows must disappear from the displaying table view. The following row(s) of the emp_ts table in the database would be updated with the following:
<table>
<tr><th colspan="11">emp_ts</th></tr>
<tr><td>ets_id</td><td>emp_id</td><td>day</td><td>wdate</td><td>intime</td><td>lunch_out</td><td>lunch_in</td><td>outtime</td><td>total_hours</td><td>status</td><td>date1</td></tr>
<tr><td>1833</td><td>99</td><td>Tuesday</td><td>2020-02-25</td><td>10:00:00</td><td>11:00:00</td><td>12:00:00</td><td>21:30:00</td><td>10.5</td><td>approved</td><td>2020-02-25</td></tr>
<tr><td>1834</td><td>99</td><td>Wednesday</td><td>2020-02-26</td><td>10:00:00</td><td>11:00:00</td><td>11:00:00</td><td>21:00:00</td><td>11</td><td>approved</td><td>2020-02-26</td></tr>
</table></td></tr>
</table>

Test Case ID	SystemTest-PMS-ApproveTS-010-Sunny03
Purpose	Investigate the proper execution of the approve time sheet use case for SFTalent Co. (an employer) and Sara Smith (an employee).
Test Set Up	The PMS system is set up and working. SFTalent Co. (the employer) is using Chrome as their browser and he is currently on the home page for PMS. Database contents and tables setup are as described in introduction of section 5.1 (System Test Cases) of this document. It is also assumed that the employer is already logged in their account.
Input	The following sequence is done: 1. SFTalent Co. (the employer) hovers on Time Sheets button in the PMS: Employer Module's Main navigation bar. 2. Then he clicks on Approve Time Sheet button in the opened drop-down menu. 3. He finally clicks on approve button.
Expected Output	The system completes the request without exceptions or errors. SFTalent Co. (the employer) have successfully approved all of the selected rows from the approval table. The approved rows must disappear from the displaying table view. There would be no updates or modification to database entries since no approval check box was selected.

6 Test Summary Report

Test Case ID	Pass/Fail	Fix
UnitTest-PMS-ModelFacadeTests-1	Pass	N/A
UnitTest-PMS-ModelFacadeTests-2	Pass	N/A
UnitTest-PMS-ModelFacadeTests-3	Fail	Implement data validation for days of the week (Monday, Tuesday, etc.)
UnitTest-PMS-ModelFacadeTests-4	Pass	N/A
UnitTest-PMS-ModelFacadeTests-5	Pass	N/A
UnitTest-PMS-ModelFacadeTests-6	Pass	N/A
UnitTest-PMS-ModelFacadeTests-7	Pass	N/A
UnitTest-PMS-ModelFacadeTests-8	Pass	N/A
UnitTest-PMS-ModelFacadeTests-9	Pass	N/A
UnitTest-PMS-ModelFacadeTests-10	Pass	N/A
UnitTest-PMS-ModelFacadeTests-11	Pass	N/A
UnitTest-PMS-ModelFacadeTests-12	Pass	N/A
UnitTest-PMS-ModelFacadeTests-13	Pass	N/A
UnitTest-PMS-ModelFacadeTests-14	Pass	N/A
UnitTest-PMS-ModelFacadeTests-15	Pass	N/A
UnitTest-PMS-ModelFacadeTests-16	Pass	N/A
UnitTest-PMS-ModelFacadeTests-17	Pass	N/A
UnitTest-PMS-ModelFacadeTests-18	Pass	N/A
UnitTest-PMS-ModelFacadeTests-19	Fail	Implement data validation for gender (“M”, “F”, “O”)
UnitTest-PMS-ModelFacadeTests-20	Fail	Implement basic format validation for emails
UnitTest-PMS-ModelFacadeTests-21	Pass	N/A
UnitTest-PMS-ModelFacadeTests-22	Pass	N/A
UnitTest-PMS-ModelFacadeTests-23	Pass	N/A

UnitTest-PMS-ModelFacadeTests-24	Pass	N/A
UnitTest-PMS-ModelFacadeTests-25	Pass	N/A
UnitTest-PMS-ModelFacadeTests-26	Pass	N/A
UnitTest-PMS-ModelFacadeTests-27	Pass	N/A
UnitTest-PMS-ModelFacadeTests-28	Pass	N/A
UnitTest-PMS-ModelFacadeTests-29	Pass	N/A
UnitTest-PMS-ModelFacadeTests-30	Pass	N/A
UnitTest-PMS-ModelFacadeTests-31	Pass	N/A
UnitTest-PMS-ModelFacadeTests-32	Fail	Implement basic format validation for emails
UnitTest-PMS-ModelFacadeTests-33	Pass	N/A
UnitTest-PMS-ModelFacadeTests-34	Pass	N/A
UnitTest-PMS-ModelFacadeTests-35	Pass	N/A
UnitTest-PMS-ModelFacadeTests-36	Pass	N/A
UnitTest-PMS-ModelFacadeTests-37	Pass	N/A
SubSystem-PMS-TimeSheetTest	Pass	Should be stateless testing.
SubSystem-PMS-SalaryTest	Pass	Should be stateless testing.
SubSystem-PMS-EmployeeTest	Pass	Should be stateless testing.
SystemTest-PMS-Login-001-Sunny01	Pass	N/A
SystemTest-PMS-Login-001-Sunny02	Pass	N/A
SystemTest-PMS-Login-001-Sunny03	Pass	N/A
SystemTest-PMS-Login-001-Rainy01	Pass	N/A
SystemTest-PMS-Login-001-Rainy02	Pass	N/A
SystemTest-PMS-Login-001-Rainy03	Pass	N/A
SystemTest-PMS-Logout-006-Sunny01	Pass	N/A
SystemTest-PMS-Logout-006-Sunny02	Pass	N/A
SystemTest-PMS-Logout-006-Sunny03	Pass	N/A
SystemTest-PMS-Logout-006-Rainy01	Pass	N/A
SystemTest-PMS-Logout-006-Rainy02	Pass	N/A

SystemTest-PMS-Logout-006-Rainy03	Pass	N/A
SystemTest-PMS-AddEmployee-002-Sunny01	Pass	An improvement could be displaying a confirmation message.
SystemTest-PMS-AddEmployee-002-Sunny02	Pass	An improvement could be displaying a confirmation message.
SystemTest-PMS-AddEmployee-002-Sunny03	Pass	An improvement could be displaying a confirmation message.
SystemTest-PMS-AddEmployee-002-Rainy01	Pass	An improvement could be displaying a confirmation message.
SystemTest-PMS-AddEmployee-002-Rainy02	Pass	An improvement could be displaying a confirmation message.
SystemTest-PMS-AddEmployee-002-Rainy03	Pass	An improvement could be displaying a confirmation message.
SystemTest-PMS-SaveTS-008-Sunny01	Fail	The timesheet page does not retrieve contents of *save_ts* for employee to show, however these values exist in database.
SystemTest-PMS-SaveTS-008-Sunny02	Fail	The timesheet page does not retrieve contents of *save_ts* for employee to show, however these values exist in database.
SystemTest-PMS-SaveTS-008-Sunny03	Fail	The timesheet page does not retrieve contents of *save_ts* for employee to show, however these values exist in database.
SystemTest-PMS-SaveTS-008-Rainy01	Fail	The timesheet page does not retrieve contents of *save_ts* for employee to show, however these values exist in database.
SystemTest-PMS-SaveTS-008-Rainy02	Fail	The timesheet page does not retrieve contents of *save_ts* for employee to show, however these values exist in database.

SystemTest-PMS-SaveTS-008-Rainy03	Fail	The timesheet page does not retrieve contents of *save_ts* for employee to show, however these values exist in database.
SystemTest-PMS-ModifyTS-004-Sunny01	Pass	N/A
SystemTest-PMS-ModifyTS-004-Sunny02	Pass	N/A
SystemTest-PMS-ModifyTS-004-Sunny03	Pass	N/A
SystemTest-PMS-ModifyTS-004-Rainy01	Pass	N/A
SystemTest-PMS-ModifyTS-004-Rainy02	Pass	N/A
SystemTest-PMS-ModifyTS-004-Rainy03	Pass	N/A
SystemTest-PMS-SubmitTS-009-Sunny01	Pass	N/A
SystemTest-PMS-SubmitTS-009-Sunny01	Pass	N/A
SystemTest-PMS-SubmitTS-009-Sunny01	Pass	N/A
SystemTest-PMS-SubmitTS-009-Rainy01	Pass	N/A
SystemTest-PMS-SubmitTS-009-Rainy02	Pass	N/A
SystemTest-PMS-SubmitTS-009-Rainy03	Pass	N/A
SystemTest-PMS-ApproveTS-010-Sunny01	Pass	N/A
SystemTest-PMS-ApproveTS-010-Sunny02	Pass	N/A
SystemTest-PMS-ApproveTS-010-Sunny03	Pass	N/A
SystemTest-PMS-ApproveTS-010-Rainy01	Pass	N/A
SystemTest-PMS-ApproveTS-010-Rainy02	Pass	N/A
SystemTest-PMS-ApproveTS-010-Rainy03	Pass	N/A
SystemTest-PMS-SecurityQuestion-003-Sunny01	Fail	The lists which contain different security questions in the front-end implemented page, does not contain 3 questions.
SystemTest-PMS-SecurityQuestion-003-Sunny01	Fail	The lists which contain different security questions in the front-end implemented page, does not contain 3 questions.

SystemTest-PMS-SecurityQuestion-003-Sunny01	Fail	The lists which contain different security questions in the front-end implemented page, does not contain 3 questions.
SystemTest-PMS-SecurityQuestion-003-Rainy01	Fail	The lists which contain different security questions in the front-end implemented page, does not contain 3 questions.
SystemTest-PMS-SecurityQuestion-003-Rainy02	Fail	The lists which contain different security questions in the front-end implemented page, does not contain 3 questions.
SystemTest-PMS-SecurityQuestion-003-Rainy03	Fail	The lists which contain different security questions in the front-end implemented page, does not contain 3 questions.
SystemTest-PMS-CalcSal-005-Sunny01	Pass	Could have utilized the *paymode* table entries. Also there is no empty check for *paymode* table which can have negative impact on the use case.
SystemTest-PMS-CalcSal-005-Sunny02	Pass	Could have utilized the *paymode* table entries. Also there is no empty check for *paymode* table which can have negative impact on the use case.
SystemTest-PMS-CalcSal-005-Sunny03	Pass	Could have utilized the *paymode* table entries. Also there is no empty check for *paymode* table which can have negative impact on the use case.
SystemTest-PMS-CalcSal-005-Rainy01	Pass	Could have utilized the *paymode* table entries. Also there is no empty check for *paymode* table which can have

		negative impact on the use case.
SystemTest-PMS-CalcSal-005-Rainy02	Pass	Could have utilized the *paymode* table entries. Also there is no empty check for *paymode* table which can have negative impact on the use case.
SystemTest-PMS-CalcSal-005-Rainy03	Pass	Could have utilized the *paymode* table entries. Also there is no empty check for *paymode* table which can have negative impact on the use case.
SystemTest-PMS-ViewProfile-007-Sunny01	Pass	Could have implemented a separate page for profile views instead of using the pay slips view.
SystemTest-PMS-ViewProfile-007-Sunny02	Pass	Could have implemented a separate page for profile views instead of using the pay slips view.
SystemTest-PMS-ViewProfile-007-Sunny03	Pass	Could have implemented a separate page for profile views instead of using the pay slips view.
SystemTest-PMS-ViewProfile-007-Rainy01	Pass	Could have implemented a separate page for profile views instead of using the pay slips view.
SystemTest-PMS-ViewProfile-007-Rainy02	Pass	Could have implemented a separate page for profile views instead of using the pay slips view.
SystemTest-PMS-ViewProfile-007-Rainy03	Pass	Could have implemented a separate page for profile views instead of using the pay slips view.

7 Risks & Contingencies

The main risk we incur when undergoing these tests is the accidental modification of live data in a production instance of this software. To remedy this, we use dedicated testing databases with dummy data to simulate a live testing environment. This solution allowed us to test with realistic conditions, simulating unpredictable user input, without risking interfering with the data of a production instance.

The tests themselves may also be incomplete or prone to human error, as with any human-generated product. To remedy this, we plan to continuously review our test cases, and create more as we see fit, to ensure optimal code coverage and as few faults as possible.

8 Approvals

Approval Page of Specification-Based Test Document of

Payroll Management System

Testing Team Member Signatures

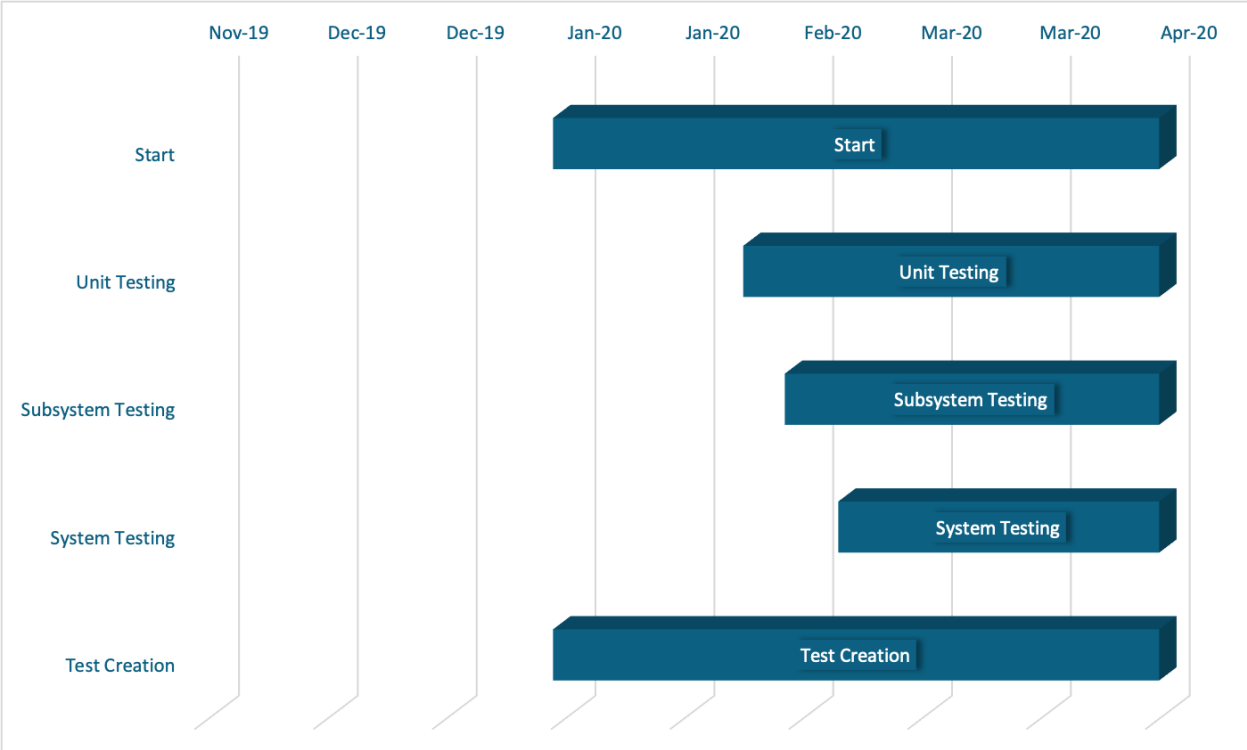
M. Kian Maroofi	02/29/2020
Member Signature	Date
Matt Taylor	02/29/2020
Member Signature	Date
Alexander Jimenez	02/29/2020
Member Signature	Date
Kristian	02/29/2020
Member Signature	Date
Nicholas Delamo	02/29/2020
Member Signature	Date

9 Glossary

- **Scenario**, a scene that illustrates some interactions of the proposed system.
- **Gantt Chart,** a bar chart where the x-axis is time and the y-axis is the different tasks, and the duration of each task is represented by the length of a bar.
- **Unified Software Development Model**, …
- **PMS,** Payroll Management System
- **STD,** Specification-Based Test Document
- **Role,** a set of technical and managerial tasks that are expected from a participant or a team.
- **Activity,** a set of tasks performed towards a specific purpose.
- **Task,** an atomic unit of work that can be managed and that consumes resources.
- **Milestone,** end-point of a software process activity.
- **Deliverable,** a work product for the client.
- **Notation,** a graphical or textual set of rules representing a model.
- **Method,** a repeatable technique for solving a specific problem.
- **Methodology,** a collection of methods for solving a class of problems.
- **Use Case,** a sequence of events describing all possible actions between actors and the system for a given piece of functionality.
- **Actors,** the roles interacting with the system such as end-users and other computer systems.

10 Appendix

10.1 Appendix A – Test Schedule



10.2 Appendix B -- Implemented Use Cases

10.2.1 **Login**

Use Case ID: PMS_02_Login
Use Case Level:
Details: Tom enters his/her username “Maria001” and password “Password1234” to log in to the system.
Actor: the actor in this case is Employer(Maria).
Pre-conditions: Maria should be authenticated employer of the system.
Description:
Trigger: The system responds by ...

a. The use case begins when the Maria enters username “Maria001” and password “Password1234” in the login page.
b. Maria then clicks on “Sign in” button in the login page.
c. PMS system authenticates username “Maria001” and password “Password1234”

Relevant requirements:
Post-conditions: 1) Maria is taken to PMS Homepage.

10.2.2 **Logout**

Use Case ID: PMS_21_logout**.**
Actors: Maria, Employee
Pre-conditions:

1) Maria has an account in payroll management system.
2) Maria tries to login with username “maria007” and password “pmsmaria123”.

Description: This will help the user to logout.
Trigger:
Maria clicks the “logout” button in the home page. The system responds by ...

1) Maria is logged out from the system.
2) Maria is shown a message saying “Successfully logged out”.

Post-conditions: 1) Maria is redirected to login page.

10.2.3 **Add Employee**

Use Case ID: PMS_13_AddEmployee.
Actor: the actor in this case is Amit who is the Employee.
Pre-conditions:

Amit is logged in using valid employee_id “ashen007”.
Amit is in the manage tab of the webpage.

Description:
Trigger: Amit clicks on the manage tab. He has two options to manage Employee and Employee. 1) Amit clicks on the “Employee” option

2) The PM system will then give options to Amit like to add a Employee.
3) Amit clicks on add Employee.

4) Then Amit fills in the info like:

d. DeptName: Deployment. DeptLocation: Room155. DeptManager: Popya.

5) Then Amit clicks on save button.

Post-conditions: The PM system will create a new Employee with name "Popya" to the department "Deployment"

10.2.4 **Modify TS**

Use Case ID : PMS_05_ModifyTS

Actors: Maria (Employer**).**

Pre-conditions: The user Maria logs in with her credentials.

Description: The employer Maria will modify Tom's timesheet which had already been submitted. Maria clicks "SearchEmployee" menu item in PMS.

The system responds by

1)Maria searches Tom's information by Tom's ID (001).

2) The system shows the unapproved timesheet for the Tom's ID i.e. 001 that Maria is searching for.

3) Now Maria will select the timesheet she is looking for and will click on the edit button.

4) Now Maria will edit the timesheet and click the approve button when she verifies that the information provided is valid.

Post-conditions:

l) The timesheet is forwarded towards calculating the Paycheck.

10.2.5 **Approve TS**

Use Case ID : PMS_05_ApproveTS

Actors: Maria (Employer).

Pre-conditions:

l) The user Maria logs in with her credentials.

Description:

The employer Maria will approve the employee Tom's time sheet which had already been submitted.

Trigger: Maria clicks "SearchEmployee" menu item in PMS. The system responds by

1) Maria searches Tom's information by Tom's ID (001).

2) The system shows the unapproved timesheet for the Tom's ID i.e. 001 that Maria is searching for.

3) Now Maria will select the timesheet she is looking for and will click on the view button. Now Maria will view the timesheet and click the approve button when she verifies that the information provided is valid.

Post-conditions:

l) The timesheet is forwarded towards calculating the Paycheck.

10.2.6 **Calculate Salary**

Use Case ID : PMS_07_CalcSal

Actors: Maria (Employer).
Pre-conditions:
1) The user Maria logs in with her credentials.
Description:
1) The employer Maria will click on the salary in the menu.
Trigger: Maria clicks "CalcSalary" menu item in PMS. The system responds by
1) Maria sees the Salary page opened.
2) Maria searches Tom's information by Tom's [D (001).
3) The system shows the salary in the menu.
4) Now Maria will select the Calcsalary button of the Tom.
5) Now Maria will View the Calcsalary of the Tom.
Post-conditions:
l) The timesheet is moved back to salary window.

10.2.7 **Save TS**
Use Case ID : PMS_08_SaveTS
Actors: Tom (Employee).
Pre-conditions:
l. The user Tom logs in with her credentials.
2. The employee Tom's timesheet will saved.
Trigger: Tom clicks "Timesheet" menu item in PMS. The system responds by
1. Tom fills the Timesheet for particular duration of work. Timesheet.
2. The system shows values inserted in
3. Tom saves the Timesheet values inserted using Save Timesheet button.
Post-conditions:
l. The timesheet is forwarded towards calculating the Paycheck.

10.3 Appendix C – Test Drivers & Stubs

10.3.1 Unit Test Drivers & Stubs

```
package unitTests;

import static org.junit.Assert.*;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import model.ModelFacade;
import utilities.DBConnection;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class ModelFacadeTest {

	// Defined here to avoid having to define them twice for setup and tear-down
	private String qry1 = "drop database if exists PMS;",
	               qry2 = "create database PMS;",
	               qry3 = "use PMS;",
	               qry4 = "create table employees(" +
	                      "  emp_id VARCHAR(500)," +
	                      "  first_name VARCHAR(500)," +
	                      "  last_name VARCHAR(500)," +
	                      "  gender VARCHAR(500)," +
	                      "  dob DATE," +
	                      "  job VARCHAR(500)," +
	                      "  phone VARCHAR(500)," +
	                      "  email_id VARCHAR(500)," +
	                      "  address VARCHAR(500)," +
	                      "  accno VARCHAR(500)," +
	                      "  bankname VARCHAR(500)," +
	                      "  joindate DATE" +
	                      ");",
	               qry5 = "create table employer(" +
	                      "  username VARCHAR(500)," +
	                      "  password VARCHAR(500)" +
	                      ");",
	               qry6 = "create table users(" +
	                      "  emp_id VARCHAR(500)," +
	                      "  user_id VARCHAR(500)," +
	                      "  password VARCHAR(500)," +
	                      "  sec_que1 VARCHAR(500)," +
	                      "  ans1 VARCHAR(500)," +
	                      "  sec_que2 VARCHAR(500)," +
	                      "  ans2 VARCHAR(500)," +
```

```
                        "  sec_que3 VARCHAR(500)," +
                        "  ans3 VARCHAR(500)," +
                        "  createDate DATE" +
                        ");",
                    qry7 = "create table paymode(" +
                        "  normal_pay DOUBLE," +
                        "  extra_pay DOUBLE" +
                        ");",
                    qry8 = "create table emp_ts(" +
                        "  ets_id VARCHAR(500)," +
                        "  emp_id VARCHAR(500)," +
                        "  day VARCHAR(500)," +
                        "  wdate DATE," +
                        "  intime TIME," +
                        "  lunch_out TIME," +
                        "  lunch_in TIME," +
                        "  outtime TIME," +
                        "  total_hours DOUBLE," +
                        "  status VARCHAR(500)," +
                        "  date1 DATE" +
                        ");",
                    qry9 = "create table save_ts(" +
                        "  ets_id VARCHAR(500)," +
                        "  emp_id VARCHAR(500)," +
                        "  day VARCHAR(500)," +
                        "  date1 DATE," +
                        "  intime TIME," +
                        "  lunch_out TIME," +
                        "  lunch_in TIME," +
                        "  outtime TIME," +
                        "  total_hours DOUBLE," +
                        "  status VARCHAR(500)," +
                        "  createDate DATE" +
                        ");",
                    qry10 = "create table salaries(" +
                        "  emp_id VARCHAR(500)," +
                        "  total_hours DOUBLE," +
                        "  tax DOUBLE," +
                        "  gross_sal DOUBLE," +
                        "  net_sal DOUBLE," +
                        "  date1 DATE" +
                        ");",
                    qry11 = "INSERT INTO `employer` (username, password) VALUES
('user1', 'user1');",
                    qry12 = "INSERT INTO `users` (emp_id, user_id, password,
createDate) VALUES ('1', 'user1', " +
                        "'user1', '2020-01-12');",
                    qry13 = "INSERT INTO `employees` VALUES
('1','Adam','Sandler','on','1901-01-01','Movie " +
                        "Star','0','adam.sandler@email.com','2121 SW 12TH
ST','1','Bank of America','2020-" +
                        "01-14')," +
                        "('2', 'Dave', 'Grohl', 'yes', '1986-04-22', 'Rock Star',
'0', 'test@email.com', " +
```

```
                        "'900 West St', '2', 'CitiBank', '2020-03-03');",
                  qry14 = "INSERT INTO `users` VALUES ('1','adam','adam','Favorite
Color?','pink','First PEt " +
                        "Name?','adam','Favorite movie?','adam','2020-01-14');",
                  qry15 = "INSERT INTO `emp_ts` VALUES ('1144','1','Monday','2020-
01-" +

      "01','10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-01-14')," 
+
                              "('1114','1','Tuesday','2020-01-" +

      "02','10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-01-14')," 
+
                              "('1172','1','Wednesday','2020-01-" +

      "03','10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-01-14')," 
+
                              "('1341','1','Thursday','2020-01-" +

      "04','10:00:00','12:00:00','13:00:00','22:00:00',11,'not approved','2020-01-
14')," +
                              "('1776','2','Thursday','2020-01-" +

      "04','10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-01-14')," 
+
                              "('1337','2','Friday','2020-01-" +

      "04','10:00:00','12:00:00','13:00:00','22:00:00',11,'not approved','2020-01-
14')," +
                              "('1800','1','Friday','2020-01-" +

      "05','10:00:00','12:00:00','13:00:00','22:00:00',11,'not approved','2020-01-
14');",
                  qry16 = "INSERT INTO `save_ts` VALUES ('1126','1','Monday','2020-
03-04'," +

      "'10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-03-07');",
                  qry17 = "INSERT INTO `paymode` VALUES (10,15);",
                  qry18 = "INSERT INTO `salaries` VALUES ('1',55,165,550,385,'2020-
01-14')," +
                              "('2',22,30,220,285,'2020-01-14');";

      @Before
      public void setUp() throws Exception {
            // For setup, we add rows to tables to use to test methods that involve
updating, deleting,
            // or requesting data from those tables. We do this directly through
DB.connection to avoid using
            // the same code that we're testing
            Connection con = DBConnection.createConnection();
            Statement st = con.createStatement();

            // Set up database using the data from pmsdb.sql
            st.addBatch(qry1);
```

```
		st.addBatch(qry2);
		st.addBatch(qry3);
		st.addBatch(qry4);
		st.addBatch(qry5);
		st.addBatch(qry6);
		st.addBatch(qry7);
		st.addBatch(qry8);
		st.addBatch(qry9);
		st.addBatch(qry10);
		st.addBatch(qry11);
		st.addBatch(qry12);
		st.addBatch(qry13);
		st.addBatch(qry14);
		st.addBatch(qry15);
		st.addBatch(qry16);
		st.addBatch(qry17);
		st.addBatch(qry18);

		int[] res = st.executeBatch();

		// Make sure all responses are good
		for (int i : res) {
			if (i == Statement.EXECUTE_FAILED)
				throw new Exception("Error while setting up database for
tests, please check statements.");
		}
	}

	@After
	public void tearDown() throws Exception {
		// Same as setup for now to ensure consistency
		Connection con = DBConnection.createConnection();
		Statement st = con.createStatement();

		st.addBatch(qry1);
		st.addBatch(qry2);
		st.addBatch(qry3);
		st.addBatch(qry4);
		st.addBatch(qry5);
		st.addBatch(qry6);
		st.addBatch(qry7);
		st.addBatch(qry8);
		st.addBatch(qry9);
		st.addBatch(qry10);
		st.addBatch(qry11);
		st.addBatch(qry12);
		st.addBatch(qry13);
		st.addBatch(qry14);
		st.addBatch(qry15);
		st.addBatch(qry16);
		st.addBatch(qry17);
		st.addBatch(qry18);

		int[] res = st.executeBatch();
```

```
            // Make sure all responses are good
            for (int i : res) {
                  if (i == Statement.EXECUTE_FAILED)
                        throw new Exception("Error while setting up database for
tests, please check statements.");
            }
      }

      @Test
      public void testTimeSheetaddTimeSheet() {
            // Valid data, expect success
          String result = ModelFacade.TimeSheetaddTimeSheet("1", "1", "Monday",
                  "2020-03-02", "09:00:00", "12:00:00", "13:00:00", "17:00:00");

            assertEquals("success", result);

          Connection con = DBConnection.createConnection();

          try {
            Statement st = con.createStatement();
                ResultSet rs = st.executeQuery("select * from save_ts");

                int numRows = 0;
                while (rs.next()) {
                  numRows++;
                }

                // Should be 2 rows in save_ts, 1 of which was newly added
                assertEquals(2, numRows);

            } catch (SQLException e) {
                  fail("SQLException encountered: " + e.toString());
            }

            // Invalid data, expect failure
          result = ModelFacade.TimeSheetaddTimeSheet("1", "1", "2020/22-aa",
                  "2020-03-02", "09:00:00", "12 noon", "13:00:00", "5 pm");

            assertFalse(result.equals("success"));
      }

      @Test
      public void testTimeSheetupdateTimeSheet() {
            // Valid data, expect success
          String result = ModelFacade.TimeSheetupdateTimeSheet("1144",
                  "10:00:00","12:00:00","13:00:00","22:00:00", "11");

            assertEquals("success", result);

            // Invalid data, expect failure
            result = ModelFacade.TimeSheetupdateTimeSheet("1144",
                  "10:00:00","aaaaaaa","6:00 pm","22:00:00", "yes");
```

```
            assertFalse(result.equals("success"));
      }

      @Test
      public void testTimeSheetsubmitTimeSheet() {
            // Valid data, expect success
            // This should move the time sheet with ID 1126 from save_ts to emp_ts
          ModelFacade.TimeSheetsubmitTimeSheet("1");

          Connection con = DBConnection.createConnection();

          try {
             Statement st = con.createStatement();
                 ResultSet rs = st.executeQuery("select * from save_ts where ets_id =
'1126'");
                 // Equivalent to asserting that rs must be empty
                    assertFalse(rs.next());

                    rs=st.executeQuery("select * from emp_ts where ets_id = '1126'");
                    assertTrue(rs.next());
             } catch (SQLException e) {
                    fail("SQLException encountered: " + e.toString());
             }
      }

      @Test
      public void testTimeSheetgetEmpTimeSheetNotApproved() {
            try {
                    // Valid data, expect success
                ResultSet rs = ModelFacade.TimeSheetgetEmpTimeSheetNotApproved("1");

                // Count the number of rows in the response
                int numRows = 0;
                while (rs.next())
                   numRows++;

                // Expecting 2 rows of not-approved time sheets for employee 1
                   assertEquals(2, numRows);

                   // Invalid data, expect fail
                rs = ModelFacade.TimeSheetgetEmpTimeSheetNotApproved("aaaaaaaaa");

                   assertFalse(rs.next());
             } catch (SQLException e) {
                    fail("SQLException encountered: " + e.toString());
             }
      }

      @Test
      public void testTimeSheetgetTimeSheetApproved() {
            try {
                    // Valid data, expect success
                ResultSet rs = ModelFacade.TimeSheetgetTimeSheetApproved("1");
```

```
            // Count the number of rows in the response
            int numRows = 0;
            while (rs.next())
               numRows++;

            // Expecting 3 rows of approved time sheets for employee 1
               assertEquals(3, numRows);

               // Invalid data, expect fail
            rs = ModelFacade.TimeSheetgetEmpTimeSheetNotApproved("aaaaaaaaa");

               assertFalse(rs.next());
        } catch (SQLException e) {
               fail("SQLException encountered: " + e.toString());
        }
    }

    @Test
    public void testTimeSheetgetTimeSheetNotApproved() {
        try {
               // Valid data, expect success
            ResultSet rs = ModelFacade.TimeSheetgetTimeSheetNotApproved();

            // Count the number of rows in the response
            int numRows = 0;
            while (rs.next())
               numRows++;

            // Expecting 3 rows of not-approved time sheets across all employees
               assertEquals(3, numRows);
        } catch (SQLException e) {
               fail("SQLException encountered: " + e.toString());
        }
    }

    @Test
    public void testTimeSheetgetTimeSheetApprovedEmpIds() {
        try {
               // Valid data, expect success
            ResultSet rs = ModelFacade.TimeSheetgetTimeSheetApprovedEmpIds();

            // Count the number of rows in the response
            int numRows = 0;
            while (rs.next())
               numRows++;

            // Expecting 2 distinct employees with approved time sheets
               assertEquals(2, numRows);
        } catch (SQLException e) {
               fail("SQLException encountered: " + e.toString());
        }
    }

    @Test
```

```
public void testSalarycalculateSalary() {
       // Valid data, expect success
       // This should populate the salaries table with employee salaries
       ModelFacade.SalarycalculateSalary();

    Connection con = DBConnection.createConnection();

    try {
       Statement st = con.createStatement();
           ResultSet rs = st.executeQuery("select * from salaries");

           int numRows = 0;
           while (rs.next()) {
              numRows++;
           }

           // Should be 4 rows in salaries, 2 of which were newly calculated
           assertEquals(4, numRows);

       } catch (SQLException e) {
              fail("SQLException encountered: " + e.toString());
       }
}

@Test
public void testSalarygetEmpPays() {
       try {
              // Valid data, expect success
           ResultSet rs = ModelFacade.SalarygetEmpPays();

           // Count the number of rows in the response
           int numRows = 0;
           while (rs.next())
              numRows++;

           // Expecting 2 rows of salaries from all employees
              assertEquals(2, numRows);
       } catch (SQLException e) {
              fail("SQLException encountered: " + e.toString());
       }
}

@Test
public void testSalarygetEmpPay() {
       try {
              // Valid data, expect success
           ResultSet rs = ModelFacade.SalarygetEmpPay("2");

           // Count the number of rows in the response
           int numRows = 0;
           while (rs.next())
              numRows++;

           // Expecting 1 rows of salaries from employee 2
```

```
                assertEquals(1, numRows);
            } catch (SQLException e) {
                fail("SQLException encountered: " + e.toString());
            }
        }

        @Test
        public void testSalaryaddPayMode() {
            // Valid data, expect success
            ModelFacade.SalaryaddPayMode(20, 40);

            Connection con = DBConnection.createConnection();

            try {
               Statement st = con.createStatement();
                   ResultSet rs = st.executeQuery("select * from paymode");

                   int numRows = 0;
                   while (rs.next()) {
                      numRows++;
                   }

                   // Should be 2 rows in paymode, 1 of which was newly added
                   assertEquals(2, numRows);

               } catch (SQLException e) {
                      fail("SQLException encountered: " + e.toString());
               }
        }

        @Test
        public void testEmployeeaddEmployee() {
               // Valid employee data
               String result = ModelFacade.EmployeeaddEmployee("4", "Hunter", "Biden",
"M", "1986-04-21",
                              "Mailman", "3059032234", "test@email.com", "900 Walker
Street",
                              "1234567890", "Bank of America");
               assertEquals("success", result);

            Connection con = DBConnection.createConnection();

            try {
               Statement st = con.createStatement();
                   ResultSet rs = st.executeQuery("select * from employees");

                   int numRows = 0;
                   while (rs.next()) {
                      numRows++;
                   }

                   // Should be 3 rows in employees, 1 of which was newly added
                   assertEquals(3, numRows);
```

```
            } catch (SQLException e) {
                  fail("SQLException encountered: " + e.toString());
            }

            // Invalid employee DoB, should fail
            result = ModelFacade.EmployeeaddEmployee("5", "Hunter", "Biden", "M",
"aaaaaaaaaaaa", // invalid date,
                        "Mailman", "3059032234", "test@email.com", "900 Walker
Street",
                        "1234567890", "Bank of America");
            assertEquals("fail", result);
      }

      @Test
      public void testEmployeechangePassword() {
            // Valid employee data
            String result = ModelFacade.EmployeechangePassword("1", "adam",
"Favorite Color?", "pink",
                        "First PEt Name?", "adam", "Favorite movie?" , "adam",
"adam", "swordfish");
            assertEquals("success", result);

          Connection con = DBConnection.createConnection();

          try {
             Statement st = con.createStatement();
                 ResultSet rs = st.executeQuery("select password from users where
user_id = 'adam'");
                 rs.next(); // Move the cursor to the first row

                 String pass = rs.getString(1);
                 // Password was changed to swordfish, so this should be true
                 assertEquals(pass, "swordfish");
             } catch (SQLException e) {
                    fail("SQLException encountered: " + e.toString());
             }

             // Invalid security question answers, should fail
           result = ModelFacade.EmployeechangePassword("1", "adam", "Favorite
Color?", "blue",
                        "First PEt Name?", "a", "Favorite movie?" , "a", "adam",
"swordfish");
            assertFalse(result.equals("success"));
      }

      @Test
      public void testEmployeedeleteEmp() {
            // Valid employee data
            String result = ModelFacade.EmployeedeleteEmp("2");
            assertEquals("success", result);

          Connection con = DBConnection.createConnection();

          try {
```

```
            Statement st = con.createStatement();
                ResultSet rs = st.executeQuery("select * from employees");

                int numRows = 0;
                while (rs.next()) {
                   numRows++;
                }

                // Should be 1 row in employees, since we just deleted 1
                assertEquals(1, numRows);

            } catch (SQLException e) {
                   fail("SQLException encountered: " + e.toString());
            }

            // Invalid employee ID, should fail
            result = ModelFacade.EmployeedeleteEmp("42");
            assertEquals("fail", result);
      }

      @Test
      public void testEmployeegetPassword() {
            // Valid employee data
            String result = ModelFacade.EmployeegetPassword("1", "adam", "Favorite
Color?", "pink",
                        "First PEt Name?", "adam", "Favorite movie?" , "adam");
            assertTrue(result.contains("success"));

            // Invalid security question answers, should fail
            result = ModelFacade.EmployeegetPassword("1", "adam", "Favorite Color?",
"yellow",
                        "First PEt Name?", "fido", "Favorite movie?" , "aaaa");
            assertFalse(result.contains("success"));
      }

      @Test
      public void testEmployeegetEmployee() {
          try {
            // Valid employee data
                ResultSet rs = ModelFacade.EmployeegetEmployee("2");

                int numRows = 0;
                while (rs.next()) {
                   numRows++;
                }

                // Should be 1 employee with the ID '2'
                assertEquals(1, numRows);

                // Invalid employee ID, should fail
                   rs = ModelFacade.EmployeegetEmployee("42");
                   assertFalse(rs.next());
            } catch (SQLException e) {
                   fail("SQLException encountered: " + e.toString());
```

```
            }
      }

      @Test
      public void testEmployeegetAllEmployees() {
          try {
             // Valid employee data
                 ResultSet rs = ModelFacade.EmployeegetAllEmployees();

                 int numRows = 0;
                 while (rs.next()) {
                    numRows++;
                 }

                 // Should be 2 employees total
                 assertEquals(2, numRows);
             } catch (SQLException e) {
                    fail("SQLException encountered: " + e.toString());
             }
      }

      @Test
      public void testEmployeeupdateEmployee() {
             // Valid employee data
             String result = ModelFacade.EmployeeupdateEmployee("2", "Hunter",
"Biden", "M", "1986-04-21",
                           "Mailman", "3059032234", "test@email.com", "900 Walker
Street",
                           "1234567890", "Bank of America");
             assertEquals("success", result);

          Connection con = DBConnection.createConnection();

          try {
             Statement st = con.createStatement();
                 ResultSet rs = st.executeQuery("select first_name from employees
where emp_id = '2'");
                 rs.next(); // Move the cursor to the first row

                 String name = rs.getString(1);
                 // Verify name was changed to "Hunter" in database
                 assertEquals(name, "Hunter");
             } catch (SQLException e) {
                    fail("SQLException encountered: " + e.toString());
             }

             // No employee with ID 42, this should fail
          result = ModelFacade.EmployeeupdateEmployee("42", "Hunter", "Biden", "M",
"1986-04-21",
                           "Mailman", "3059032234", "test@email.com", "900 Walker
Street",
                           "1234567890", "Bank of America");
             assertFalse(result.equals("success"));
      }
```

```
	@Test
	public void testEmployerauthenticate() {
		// Valid data, expect success
	    String result = ModelFacade.Employerauthenticate("user1", "user1");

		assertEquals("success", result);

		// Invalid data, expect failure
		result = ModelFacade.Employerauthenticate("aaaaaaaaaaaaaaaaa",
"bbbbbbbbbbbbbbb");

		assertFalse(result.equals("success"));
	}

	@Test
	public void testUserauthenticate() {
		// Valid data, expect success
	    String result = ModelFacade.Userauthenticate("1", "adam", "adam");

		assertEquals("success", result);

		// Invalid data, expect failure
		result = ModelFacade.Userauthenticate("1", "adam", "swordfish");

		assertFalse(result.equals("success"));
	}

	@Test
	public void testSecurity_Questionregisteremployee() {
		// Valid employee data
		String result = ModelFacade.Security_Questionregisteremployee("3",
"joe", "scallop123", "Favorite Color?", "pink",
				"First PEt Name?", "adam", "Favorite movie?" , "adam");
		assertEquals("success", result);

	    Connection con = DBConnection.createConnection();

	    try {
		   Statement st = con.createStatement();
		       ResultSet rs = st.executeQuery("select password from users where
user_id = 'joe'");
			rs.next(); // Move the cursor to the first row

			String pass = rs.getString(1);
			// Password was set to scallop123, so this should be true
			assertEquals(pass, "scallop123");
		   } catch (SQLException e) {
			  fail("SQLException encountered: " + e.toString());
		   }
	}
}
```

10.3.2 Subsystem Test Drivers & Stubs

```
package subSystemTests;

import static org.junit.Assert.*;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import model.ModelFacade;
import utilities.DBConnection;

public class SubSystemTests {
	// Defined here to avoid having to define them twice for setup and tear-down
		private String qry1 = "drop database if exists PMS;",
				qry2 = "create database PMS;",
				qry3 = "use PMS;",
				qry4 = "create table employees(" +
					"  emp_id VARCHAR(500)," +
					"  first_name VARCHAR(500)," +
					"  last_name VARCHAR(500)," +
					"  gender VARCHAR(500)," +
					"  dob DATE," +
					"  job VARCHAR(500)," +
					"  phone VARCHAR(500)," +
					"  email_id VARCHAR(500)," +
					"  address VARCHAR(500)," +
					"  accno VARCHAR(500)," +
					"  bankname VARCHAR(500)," +
					"  joindate DATE" +
					");",
				qry5 = "create table employer(" +
					"  username VARCHAR(500)," +
					"  password VARCHAR(500)" +
					");",
				qry6 = "create table users(" +
					"  emp_id VARCHAR(500)," +
					"  user_id VARCHAR(500)," +
					"  password VARCHAR(500)," +
					"  sec_que1 VARCHAR(500)," +
```

```
                                "  ans1 VARCHAR(500)," +
                                "  sec_que2 VARCHAR(500)," +
                                "  ans2 VARCHAR(500)," +
                                "  sec_que3 VARCHAR(500)," +
                                "  ans3 VARCHAR(500)," +
                                "  createDate DATE" +
                                ");",
                        qry7 = "create table paymode(" +
                                "  normal_pay DOUBLE," +
                                "  extra_pay DOUBLE" +
                                ");",
                        qry8 = "create table emp_ts(" +
                                "  ets_id VARCHAR(500)," +
                                "  emp_id VARCHAR(500)," +
                                "  day VARCHAR(500)," +
                                "  wdate DATE," +
                                "  intime TIME," +
                                "  lunch_out TIME," +
                                "  lunch_in TIME," +
                                "  outtime TIME," +
                                "  total_hours DOUBLE," +
                                "  status VARCHAR(500)," +
                                "  date1 DATE" +
                                ");",
                        qry9 = "create table save_ts(" +
                                "  ets_id VARCHAR(500)," +
                                "  emp_id VARCHAR(500)," +
                                "  day VARCHAR(500)," +
                                "  date1 DATE," +
                                "  intime TIME," +
                                "  lunch_out TIME," +
                                "  lunch_in TIME," +
                                "  outtime TIME," +
                                "  total_hours DOUBLE," +
                                "  status VARCHAR(500)," +
                                "  createDate DATE" +
                                ");",
                        qry10 = "create table salaries(" +
                                "  emp_id VARCHAR(500)," +
                                "  total_hours DOUBLE," +
                                "  tax DOUBLE," +
                                "  gross_sal DOUBLE," +
                                "  net_sal DOUBLE," +
                                "  date1 DATE" +
                                ");",
                        qry11 = "INSERT INTO `employer` (username, password)
VALUES ('user1', 'user1');",
                        qry12 = "INSERT INTO `users` (emp_id, user_id, password,
createDate) VALUES ('1', 'user1', " +
                                "'user1', '2020-01-12');",
```

```
                        qry13 = "INSERT INTO `employees` VALUES
('1','Adam','Sandler','on','1901-01-01','Movie " +
                                "Star','0','adam.sandler@email.com','2121 SW 12TH
ST','1','Bank of America','2020-" +
                                "01-14')," +
                                "('2', 'Dave', 'Grohl', 'yes', '1986-04-22', 'Rock
Star', '0', 'test@email.com', " +
                                "'900 West St', '2', 'CitiBank', '2020-03-03');",
                        qry14 = "INSERT INTO `users` VALUES
('1','adam','adam','Favorite Color?','pink','First PEt " +
                                "Name?','adam','Favorite movie?','adam','2020-01-
14');",
                        qry15 = "INSERT INTO `emp_ts` VALUES
('1144','1','Monday','2020-01-" +

      "01','10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-01-14')," 
+
                                        "('1114','1','Tuesday','2020-01-" +

      "02','10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-01-14')," 
+
                                        "('1172','1','Wednesday','2020-01-" +

      "03','10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-01-14')," 
+
                                        "('1341','1','Thursday','2020-01-" +

      "04','10:00:00','12:00:00','13:00:00','22:00:00',11,'not approved','2020-01-
14')," +
                                        "('1776','2','Thursday','2020-01-" +

      "04','10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-01-14')," 
+
                                        "('1337','2','Friday','2020-01-" +

      "04','10:00:00','12:00:00','13:00:00','22:00:00',11,'not approved','2020-01-
14')," +
                                        "('1800','1','Friday','2020-01-" +

      "05','10:00:00','12:00:00','13:00:00','22:00:00',11,'not approved','2020-01-
14');",
                        qry16 = "INSERT INTO `save_ts` VALUES
('1126','1','Monday','2020-03-04'," +

      "'10:00:00','12:00:00','13:00:00','22:00:00',11,'approved','2020-03-07');",
                        qry17 = "INSERT INTO `paymode` VALUES (10,15);",
                        qry18 = "INSERT INTO `salaries` VALUES
('1',55,165,550,385,'2020-01-14')," +
                                        "('2',22,30,220,285,'2020-01-14');";
```

```
	@Before
	public void setUp() throws Exception {
		// For setup, we add rows to tables to use to test methods that
involve updating, deleting,
		// or requesting data from those tables. We do this directly
through DB.connection to avoid using
		// the same code that we're testing
		Connection con = DBConnection.createConnection();
		Statement st = con.createStatement();

		// Set up database using the data from pmsdb.sql
		st.addBatch(qry1);
		st.addBatch(qry2);
		st.addBatch(qry3);
		st.addBatch(qry4);
		st.addBatch(qry5);
		st.addBatch(qry6);
		st.addBatch(qry7);
		st.addBatch(qry8);
		st.addBatch(qry9);
		st.addBatch(qry10);
		st.addBatch(qry11);
		st.addBatch(qry12);
		st.addBatch(qry13);
		st.addBatch(qry14);
		st.addBatch(qry15);
		st.addBatch(qry16);
		st.addBatch(qry17);
		st.addBatch(qry18);

		int[] res = st.executeBatch();

		// Make sure all responses are good
		for (int i : res) {
			if (i == Statement.EXECUTE_FAILED)
				throw new Exception("Error while setting up database
for tests, please check statements.");
		}
	}

	@After
	public void tearDown() throws Exception {
		// Same as setup for now to ensure consistency
		Connection con = DBConnection.createConnection();
		Statement st = con.createStatement();

		st.addBatch(qry1);
		st.addBatch(qry2);
		st.addBatch(qry3);
		st.addBatch(qry4);
```

```
                    st.addBatch(qry5);
                    st.addBatch(qry6);
                    st.addBatch(qry7);
                    st.addBatch(qry8);
                    st.addBatch(qry9);
                    st.addBatch(qry10);
                    st.addBatch(qry11);
                    st.addBatch(qry12);
                    st.addBatch(qry13);
                    st.addBatch(qry14);
                    st.addBatch(qry15);
                    st.addBatch(qry16);
                    st.addBatch(qry17);
                    st.addBatch(qry18);

                    int[] res = st.executeBatch();

                    // Make sure all responses are good
                    for (int i : res) {
                           if (i == Statement.EXECUTE_FAILED)
                                  throw new Exception("Error while setting up database
for tests, please check statements.");
                    }
             }

      @Test
      public void testTimeSheet()
      {
             String result = ModelFacade.TimeSheetaddTimeSheet("1", "1", "Monday",
//Add a TS
                    "2020-03-02", "09:00:00", "12:00:00", "13:00:00", "17:00:00");
          assertEquals("success", result);   //Test added time sheet

             result = ModelFacade.TimeSheetupdateTimeSheet("1",
                           "10:00:00","12:00:00","13:00:00","22:00:00", "11");
      //Update the TS
          assertEquals("success", result);   ///Test update
          ModelFacade.TimeSheetsubmitTimeSheet("1");             //Submit TS
             try
             {
                    ResultSet rs =
ModelFacade.TimeSheetgetEmpTimeSheetNotApproved("1"); //Get the TS' for that emp (Not
approved)

                    int numRows = 0;
                    while (rs.next())
                        {
                           numRows++;
                        }
```

```
                  assertEquals(3, numRows); //3 unapproved TS for EMP 1

                  ModelFacade.TimeSheetaddTimeSheet("4", "4", "Monday",
            //Add a TS
                              "2020-03-02", "09:00:00", "12:00:00", "13:00:00",
"17:00:00");
                  ModelFacade.TimeSheetaddTimeSheet("5", "5", "Monday",
            //Add a TS
                              "2020-03-02", "09:00:00", "12:00:00", "13:00:00",
"17:00:00");
                  ModelFacade.TimeSheetsubmitTimeSheet("4");
                  ModelFacade.TimeSheetsubmitTimeSheet("5");
                  //Added new TS to increase size of Not Approved TS, making total
of 5

                  rs = ModelFacade.TimeSheetgetTimeSheetNotApproved();
      //Get all unapproved TS
                  rs.next();
                  numRows = 0;
                  while (rs.next())
                  {
                        numRows++;
                  }
                  assertEquals(5, numRows);

                  rs = ModelFacade.TimeSheetgetTimeSheetApproved("1");
      //Get the TS' for that emp (Approved)
                  rs.next();
                  numRows = 0;
                  while (rs.next())
                  {
                        numRows++;
                  }
                  assertEquals(3, numRows);


                  rs = ModelFacade.TimeSheetgetTimeSheetApprovedEmpIds();
            //Get all EMPs with approved TS
                  numRows = 0;
                while (rs.next())
                {
                  numRows++;
                }
                  assertEquals(2, numRows);
            }
            catch (SQLException e)
            {
                  fail("SQLException encountered: " + e.toString());
            }
```

```
}

@Test
public void testSalary()
{
       ModelFacade.SalarycalculateSalary();

    Connection con = DBConnection.createConnection();

    try
    {
       Statement st = con.createStatement();
           ResultSet rs = st.executeQuery("select * from salaries");

           int numRows = 0;
           while (rs.next())
           {
              numRows++;
           }

           // Should be 4 rows in salaries, 2 of which were newly calculated
           assertEquals(4, numRows);

           rs = ModelFacade.SalarygetEmpPays();
           // Count the number of rows in the response
           numRows = 0;
           while (rs.next())
           {
              numRows++;
           }

           // Expecting 4 rows of salaries from all employees
              assertEquals(4, numRows);

           rs = ModelFacade.SalarygetEmpPay("2");
           // Count the number of rows in the response
           numRows = 0;
           while (rs.next())
           {
              numRows++;
           }

           // Expecting 2 rows of salaries from employee 2
              assertEquals(2, numRows);

       }

    catch (SQLException e)
```

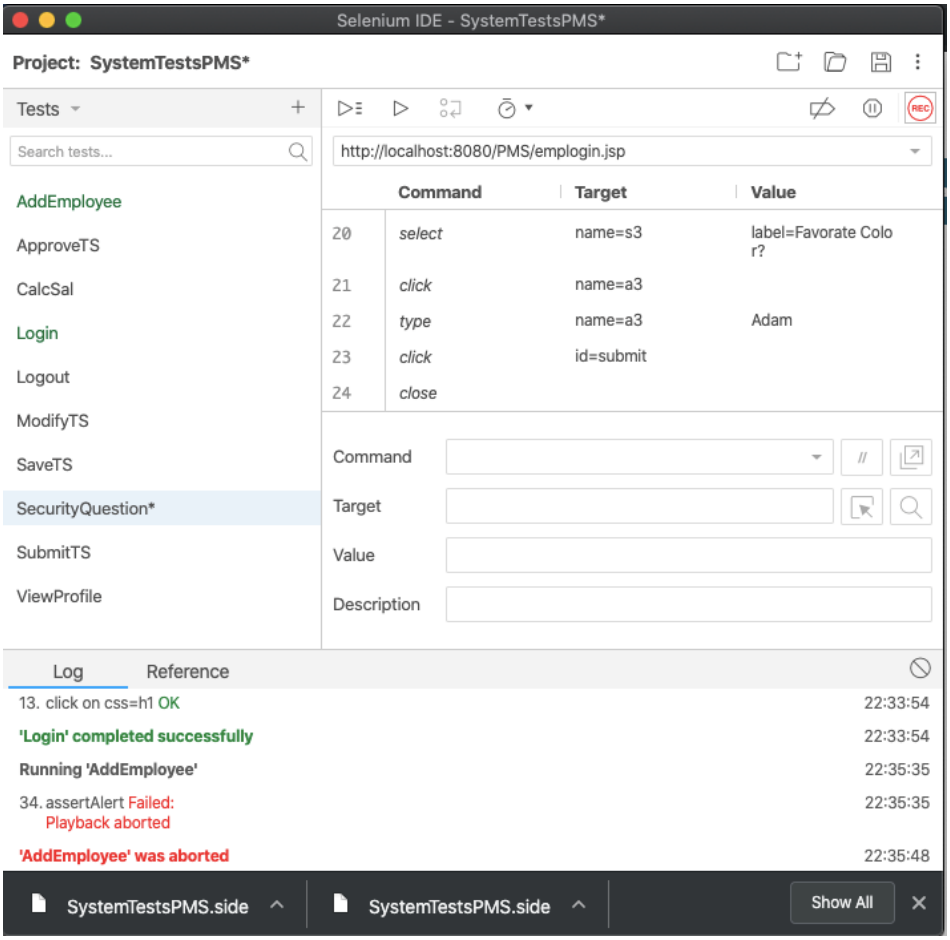
```
		{
			fail("SQLException encountered: " + e.toString());
		}

		String result = ModelFacade.SalaryaddPayMode(20, 40);
		assertEquals("success", result);
	}

	@Test
	public void testEmployee()
	{
		String result = ModelFacade.EmployeeaddEmployee("4", "Hunter", "Biden",
"M", "1986-04-21", //Add EMP "4"
				"Mailman", "3059032234", "test@email.com", "900 Walker
Street",
				"1234567890", "Bank of America");
		assertEquals("success", result);

		result = ModelFacade.EmployeechangePassword("1", "adam", "Favorite
Color?", "pink",    //Change pass to swordfish
				"First PEt Name?", "adam", "Favorite movie?" , "adam",
"adam", "swordfish");
		assertEquals("success", result);

		result = ModelFacade.EmployeeupdateEmployee("2", "Hunter", "Biden", "M",
"1986-04-21",       //Add EMP "2"
				"Mailman", "3059032234", "test@email.com", "900 Walker
Street",
				"1234567890", "Bank of America");
		assertEquals("success", result);
		result = ModelFacade.EmployeedeleteEmp("2");
					//Del EMP "2"
		assertEquals("success", result);
		result = ModelFacade.EmployeeupdateEmployee("2", "Hunter", "Biden", "M",
"1986-04-21",       //get EMP "2" (FAIL)
				"Mailman", "3059032234", "test@email.com", "900 Walker
Street",
				"1234567890", "Bank of America");
		assertEquals("fail", result);

		result = ModelFacade.EmployeegetPassword("1", "adam", "Favorite Color?",
"pink",             //Get "swordfish" pass
				"First PEt Name?", "adam", "Favorite movie?" , "adam");
		assertEquals("success,swordfish", result);

		try
		{
			ResultSet rs = ModelFacade.EmployeegetEmployee("4");
				//Get EMP ID '1'
			int numRows = 0;
```

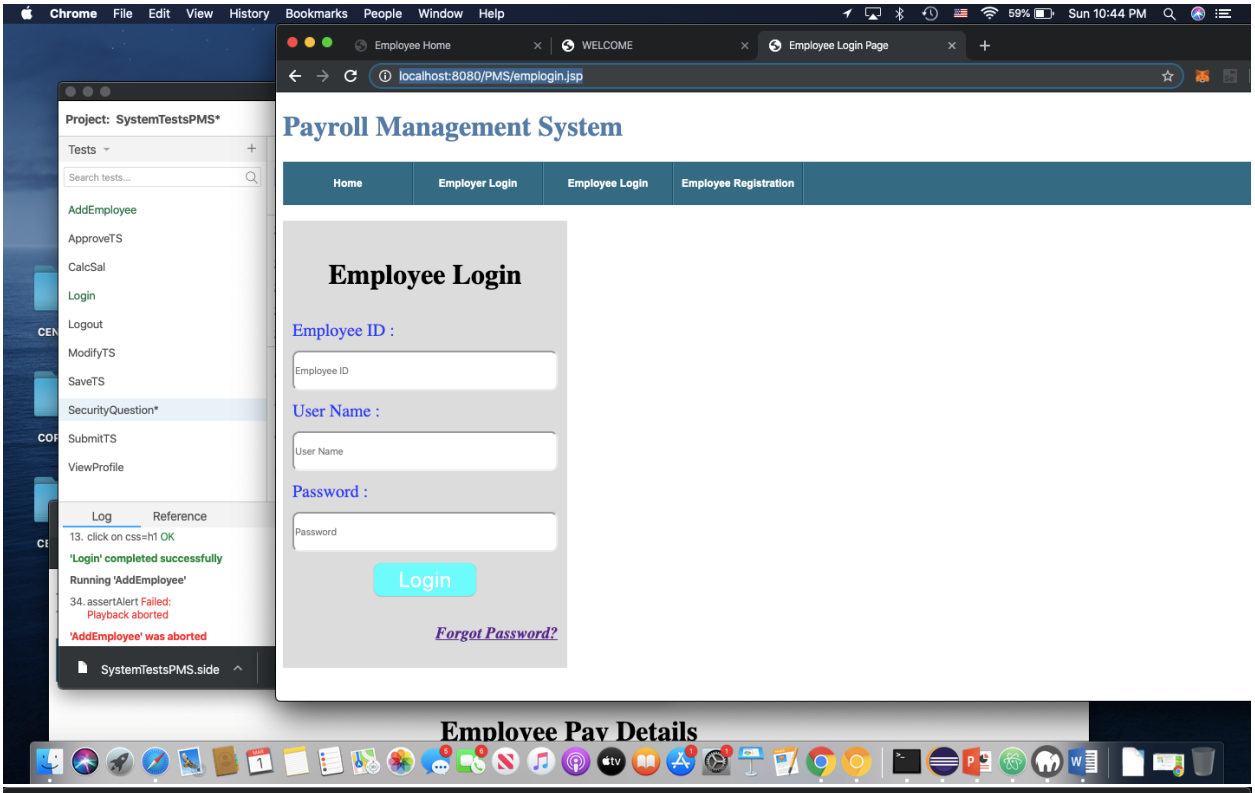
```
                while (rs.next()) {
                   numRows++;
                }

                // Should be 1 employee with the ID '1'
                assertEquals(1, numRows);

                rs = ModelFacade.EmployeegetAllEmployees();
                                        // Should be 2 employees total

                                                              // ID 1 and ID 4
                numRows = 0;
                while (rs.next())
                {
                   numRows++;
                }


                assertEquals(2, numRows);
             }
             catch (SQLException e)
          {
                   fail("SQLException encountered: " + e.toString());
             }

       }

       @Test
       public void testAuthenticate()
       {
                   // Valid data, expect success
                String result = ModelFacade.Employerauthenticate("user1", "user1");

                   assertEquals("success", result);

                   // Valid data, expect success
                result = ModelFacade.Userauthenticate("1", "adam", "adam");

                   assertEquals("success", result);
                   // Valid employee data
                   result = ModelFacade.Security_Questionregisteremployee("3",
"joe", "scallop123", "Favorite Color?", "pink",
                                  "First PEt Name?", "adam", "Favorite movie?" ,
"adam");
                   assertEquals("success", result);

                Connection con = DBConnection.createConnection();

                try {
                   Statement st = con.createStatement();
```

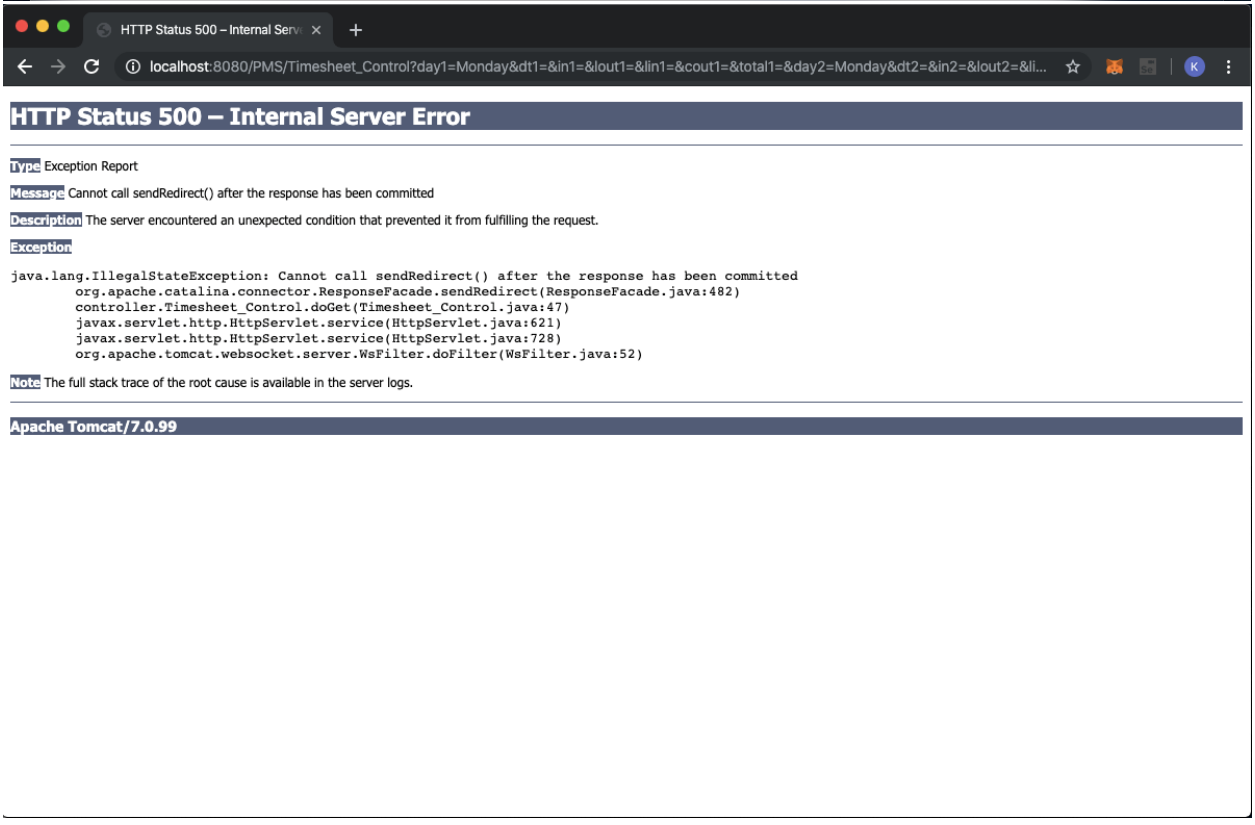
```
                        ResultSet rs = st.executeQuery("select password from users
where user_id = 'joe'");
                        rs.next(); // Move the cursor to the first row

                        String pass = rs.getString(1);
                        // Password was set to scallop123, so this should be true
                        assertEquals(pass, "scallop123");
                    } catch (SQLException e) {
                           fail("SQLException encountered: " + e.toString());
                    }


      }

}
```

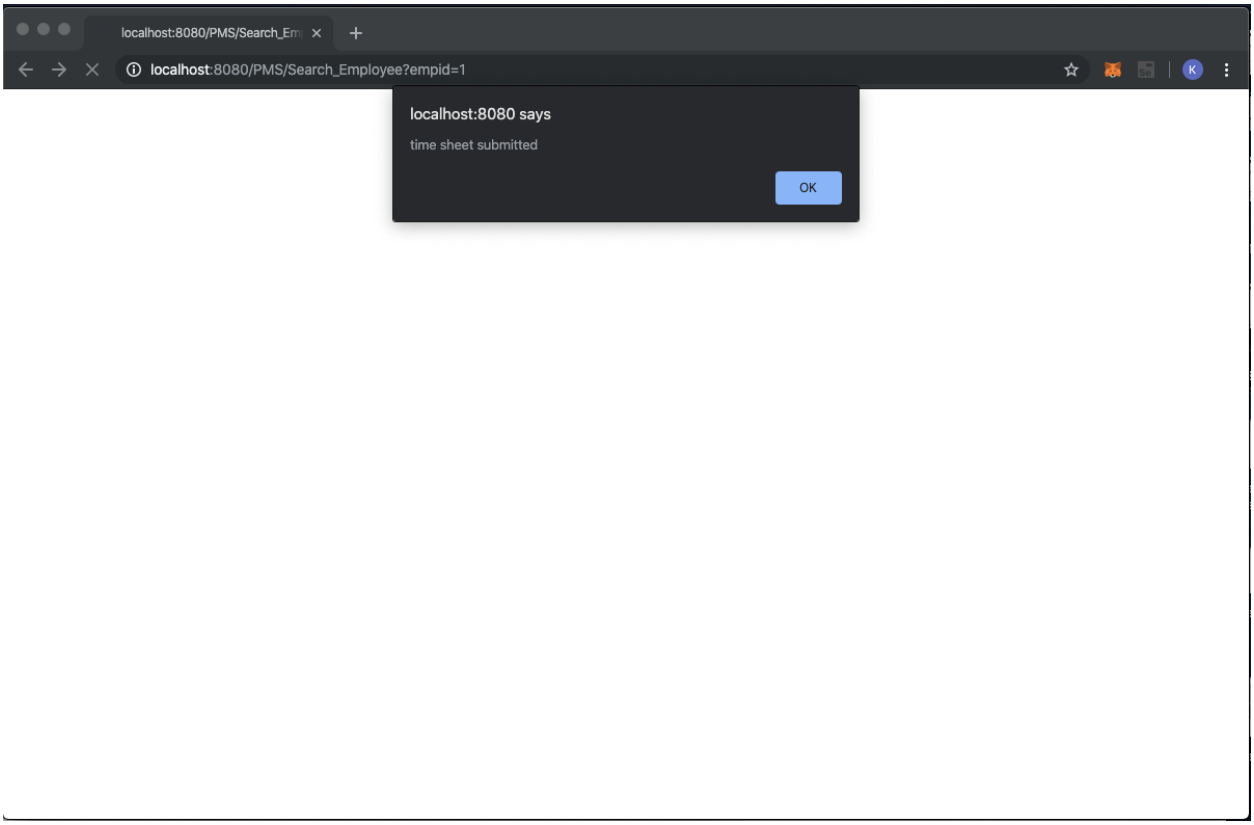
10.4 Appendix D – GUI Tests

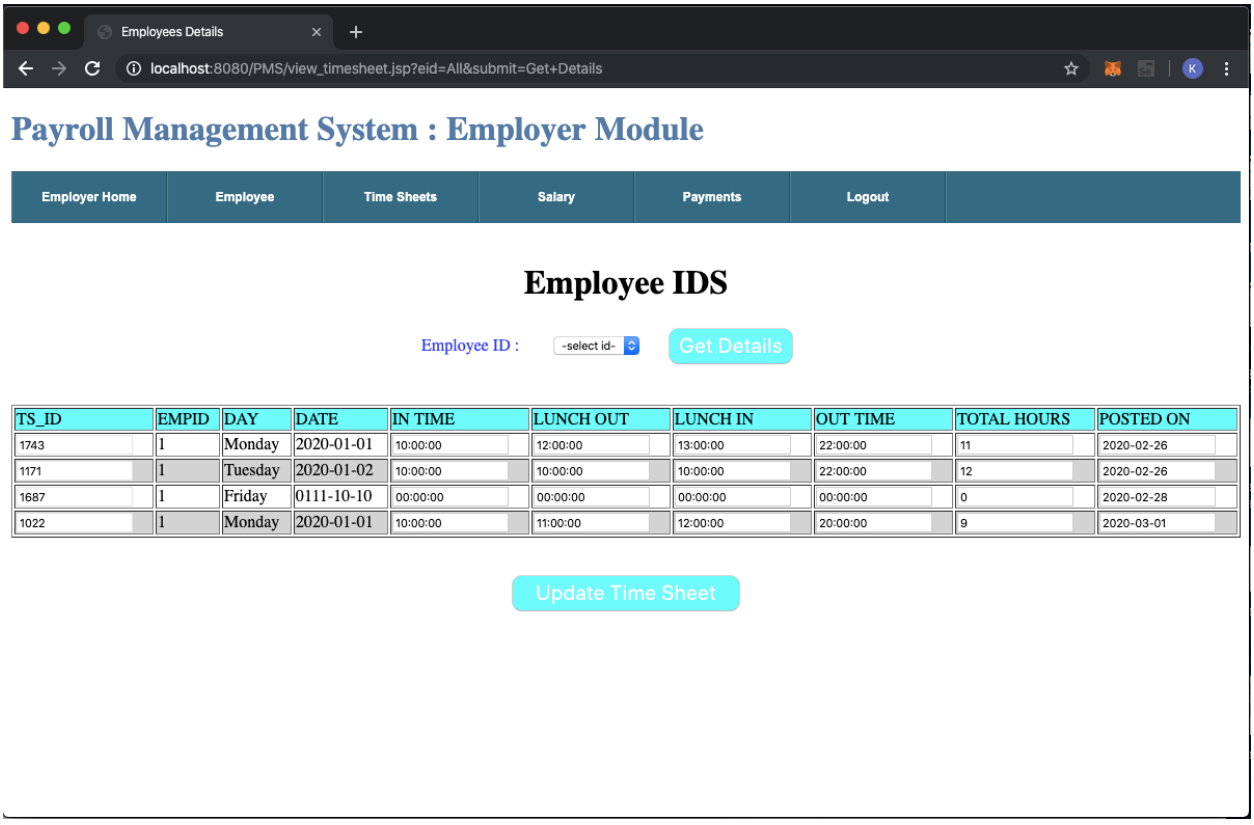
Following pictures show the system testing environment (Selenium IDE as a Google Chrome extension). They also indicate some tests outputs.

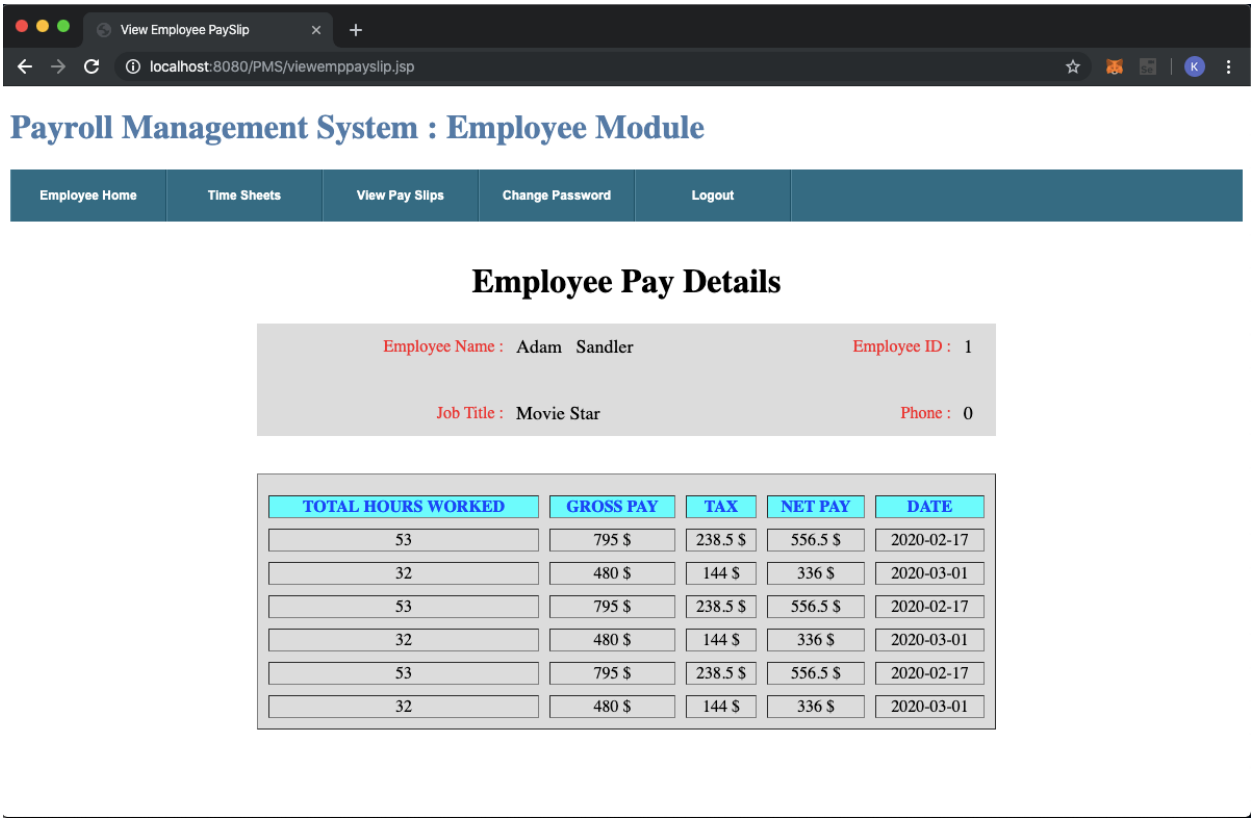



Chrome File Edit View History Bookmarks People Window Help
Sun 10:44 PM
Employee Home
WELCOME
Employee Login Page
localhost:8080/PMS/emplogin.jsp
Payroll Management System
Home
Employer Login
Employee Login
Employee Registration
Employee Login
Employee ID :
Employee ID
User Name :
User Name
Password :
Password
Login
Forgot Password?
Project: SystemTestsPMS*
Tests
Search tests...
AddEmployee
ApproveTS
CalcSal
Login
Logout
ModifyTS
SaveTS
SecurityQuestion*
SubmitTS
ViewProfile
Log
Reference
13. click on css=h1 OK
'Login' completed successfully
Running 'AddEmployee'
34. assertAlert Failed:
Playback aborted
'AddEmployee' was aborted
SystemTestsPMS.side
Employee Pay Details


HTTP Status 500 – Internal Serv
localhost:8080/PMS/Timesheet_Control?day1=Monday&dt1=&in1=&lout1=&lin1=&cout1=&total1=&day2=Monday&dt2=&in2=&lout2=&li...
HTTP Status 500 – Internal Server Error
Type Exception Report
Message Cannot call sendRedirect() after the response has been committed
Description The server encountered an unexpected condition that prevented it from fulfilling the request.
Exception
java.lang.IllegalStateException: Cannot call sendRedirect() after the response has been committed
org.apache.catalina.connector.ResponseFacade.sendRedirect(ResponseFacade.java:482)
controller.Timesheet_Control.doGet(Timesheet_Control.java:47)
javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
Note The full stack trace of the root cause is available in the server logs.
Apache Tomcat/7.0.99


localhost:8080/PMS/Search_Em
localhost:8080/PMS/Search_Employee?empid=1
localhost:8080 says
time sheet submitted
OK


Employees Details
localhost:8080/PMS/view_timesheet.jsp?eid=All&submit=Get+Details
Payroll Management System : Employer Module
Employer Home
Employee
Time Sheets
Salary
Payments
Logout
Employee IDS
Employee ID :
-select id-
Get Details
TS_ID EMPID DAY DATE IN TIME LUNCH OUT LUNCH IN OUT TIME TOTAL HOURS POSTED ON
1743 1 Monday 2020-01-01 10:00:00 12:00:00 13:00:00 22:00:00 11 2020-02-26
1171 1 Tuesday 2020-01-02 10:00:00 10:00:00 10:00:00 22:00:00 12 2020-02-26
1687 1 Friday 0111-10-10 00:00:00 00:00:00 00:00:00 00:00:00 0 2020-02-28
1022 1 Monday 2020-01-01 10:00:00 11:00:00 12:00:00 20:00:00 9 2020-03-01
Update Time Sheet


View Employee PaySlip
localhost:8080/PMS/viewemppayslip.jsp
Payroll Management System : Employee Module
Employee Home
Time Sheets
View Pay Slips
Change Password
Logout
Employee Pay Details
Employee Name : Adam Sandler
Employee ID : 1
Job Title : Movie Star
Phone : 0
TOTAL HOURS WORKED
GROSS PAY
TAX
NET PAY
DATE
53 795 $ 238.5 $ 556.5 $ 2020-02-17
32 480 $ 144 $ 336 $ 2020-03-01
53 795 $ 238.5 $ 556.5 $ 2020-02-17
32 480 $ 144 $ 336 $ 2020-03-01
53 795 $ 238.5 $ 556.5 $ 2020-02-17
32 480 $ 144 $ 336 $ 2020-03-01

10.5 Appendix E – Diary

10.5.1 February 1, 2020

When and Where	Role
Date: 2/3/2020	**Team Leader:** Matt Taylor
Start: 1:30 pm	**Timekeeper:** M. Kian Maroofi
End: 3:00 pm	**Minute Taker:** Nicholas Delamo
Room: CASE lab	**Attending:** Alexander Jimenez, Nicholas Delamo, Kristian Perez, Matt Taylor, M. Kian Maroofi

1. Status

Initial meetings. Team members are introduced to each other and the initial tasks where distributed. An initial overview of the system, tools, and resources for testing were done.

2. Discussion

Tasks were distributed based on each member's preference. We have covered all of the testing approaches we will be going through during the first deliverable. USDP (Clarke V&V model) were also discussed and analyzed in order to provide more reliable and accurate testing process for Payroll Management System. Payroll Management System or PMS were installed and run from the provided source code in class on each team member individual laptop/PC. All of the required testing tools, resources, document, etc. were analyzed and provided to all of the team members. Eclipse EE IDE were installed on each members device to be used for testing environment. Junit 4.0, MySQL, and Apache were installed on individuals' devices as well. No trouble found conducting the test using different operating systems such as Windows or macOS.

3. Wrap Up

- M. Kian Maroofi will do System Tests as well as being the timekeeper.
- Alexander Jimenez will work on ModelFacade.java for unit & subsystem tests.
- Matt Taylor will conduct Unit Tests as well as being the team leader.
- Kristian Perez will also cover Subsystem Tests.
- Nicholas Delamo will cover the testing schedule and be in charge of minute taking.

10.5.2 February 20, 2020

When and Where	Role
Date: 2/20/2020	**Team Leader:** Matt Taylor
Start: 7:30 pm	**Timekeeper:** M. Kian Maroofi
End: 8:00 pm	**Minute Taker:** Nicholas Delamo
Room: CASE lab	**Attending:** Alexander Jimenez, Nicholas Delamo, Kristian Perez, Matt Taylor, M. Kian Maroofi

1. Status

Second meeting, just checking on everyone's work process so far, making sure everything is on the right track as well as getting ready for the first deliverable and presentation for this project.

2. Discussion

System Test cases were done by M. Kian Maroofi successfully (51 test cases) using Selenium IDE as well as Junit. Matt Taylor did all of the unit test cases (21 test cases), and, Alexander Jimenez also completed the modelFacade.java successfully. Kristian Perez is working on Subsystem tests, he is still conducting subsystem tests, however, they will be done soon.

3. Wrap Up

- M. Kian Maroofi will prepare the Specification-Based Test Document (STD) for the first deliverable by putting together system test cases (section 5 of STD) as well as sections 1, 2 and appendixes.
- Matt Taylor will prepare the PowerPoint for the upcoming presentation based on the given presentation outline by Dr. Clarke.
- Nicholas Delamo will push the GNATT chart to the GitHub repository.
- Kristian Perez will finish up the remaining Subsystem tests and put them on the document.

10.5.3 March 3, 2020

When and Where	Role
Date: 3/3/2020	**Team Leader:** Matt Taylor
Start: 1:30 pm	**Timekeeper:** M. Kian Maroofi
End: 3:30 pm	**Minute Taker:** Nicholas Delamo
Room: CASE lab	**Attending:** Nicholas Delamo, Kristian Perez, Matt Taylor, M. Kian Maroofi

1. Status

Third meeting we went through the PowerPoint presentation for the first deliverable by a practice/rehearsal. Final tough ups on the STD document conducted as well.

2. Discussion

PowerPoint presentation rehearsed by attending team members. STD document completed the missing appendixes and other tables regarding some test results. GNATT chart has been added as well.

3. Wrap Up

- PowerPoint presentation for the first deliverable is rehearsed and we are ready to present out specification-based testing process for PMS system in class.
- All testing sections including unit, subsystem, and system are done and completed by assigned members for each.
- GitHub repository is up-to-date.