Group 5: Iteration 3
Membership Management Software
Test Report

Members:

Marco Puzzo - 500761565
Janakkumar Bhanushali - 500593453
Devon Li Vong Shing - 500705669
Jacob Wagner - 500754931
Nicolas Barrios-Ruiz - 500773454

Unit Test Document for Membership Management Software

This document contains the test report regarding our Membership management software project. The report consists of the test results we observed from various amounts of unit testing done on key components and functionalities of the application.

Selection of Units: When choosing units for testing, we selected all the classes that we felt were critical to the functionality of the application and were responsible for most of the operations specified by our revised product backlog.

Test Scripts: For a large portion of our tests we utilized junit in order to create test cases for our applications modules. These test cases were used to ensure that the correct output from our methods were given under many scenarios.

Testing Enhancement: As we continued testing our modules, some old test cases originally specified by our test plan were removed and some new ones were added. This is due to our design of the software changing sometimes significantly during our implementation.

Testing Tool: JUnit
For the unit testing of our project we used an open source software called JUnit. JUnit is a unit testing framework used specifically for test-driven development in the java language. We choose to use JUnit as it is simple to use and is supported by eclipse.To use JUnit you must create a class that extends “TestCase”, which is provided by JUnit. This class can be used to test a component of the software, but requires theses methods to function:

(a) setUp(): In this we instantiate various objects needed to perform the testing.
(b) tearDown(): In this we deallocate all or some of the memory which was used up by objects created in setUp() method. This is called at the end of all tests.
(c) suite(): This method is used to create a test suite, which specifies as to which tests will be performed.
(d) Various methods of the name testXXX(): These methods contain the actual code for testing. In any such method, we do whatever operations we want to do, and then call the method assertTrue()/assertFalse(), with a boolean as the argument, which specifies as to what condition we wish to hold true/false, for being convinced that the tested method performs correctly.

Test Performed: The classes that we looked at during our unit testing were (a) TreasurerUI, (b) CoachUI, (c) CustomerUI, (d) IOWork, (e) LoginInterface, and (f) User.

(a) Testing the methods of TreasurerUI to create, manipulate and delete income statements as well as user accounts:

Operation performed	Condition Tested	Actual Result
Access the treasurer’s user interface’s “Accounts” tab	The treasurer’s “Accounts” tab is selected and is displayed on the window	Test passed
Access the treasurer’s user interface’s “Management” tab	The treasurer’s “Management” tab is selected and is displayed on the window	Test passed
Create a Coach in the list of users with: First name: John Last name: Smith Username: JohnIsGreat Password: 123 Salary: 5	Check that there exists a user in userList where userFlag = 1 and username = John	Test passed
Remove the coach from the list of users with: Username = john	Check that no user with the username John and userFlag 1 exists in the arraylist “userLists”	Test passed
Create a Customer in the list of users with: First name: Janak Last name: Jacob Username: janob Password: 456 Address: new deli Phone number: 5555555555	Check that there exists a user in userList where userFlag = 2 and username = Janak	Test passed
Create a copy “janak” user First name: Janak Last name: Jacob Username: janob Password: 456 Address: new deli Phone number: 5555555555	Error message is displayed in pop up window warning user that the user specified already exists	Test passed
Remove a Customer from the list	Check that no user with the username Janak and userFlag 2 exists in the arraylist “userLists”	Test passed
Create a new income statement in the treasurer interface with: revenue : 500 Expenses: 300 Account payables: 100 Taxes: 30	Pop up appears, notifying the user that the income statement has been saved and what directory it has been saved to. Check that the directory has 1 file.	Test passed
Modify the saved income statement and change its expenses to 400 then save it to overwrite the old file	Open the “old”income statement using the management tab and observe the changed value	Test passed
Delete the saved income statement	Number of income statements in the default search folder should	Test passed

	be zero.	
Invalid income statement selection	Pop up is triggered warning user of exception error.	Test passed
Invalid information input into “Management” text fields	Pop up is triggered warning user of invalid values within the textfields	Test passed

(b) Testing the methods of CoachUI to send notifications to all customers and messages to individual customers:

Operation performed	Condition Tested	Actual Result
Send notification “next practice is tuesday” to all customers in the club	All users with userFlag 1 have their notifications string concatenated with the coach’s message “next practice is tuesday”	Test passed
Send message “next practice is tuesday” to a individual customer of the club with username “janak”	The notification string of the user with username “janak” has the string “next practice is tuesday” concatenated to it.	Test passed
Send a blank notification or message to one or all customers	Error message is displayed in pop up window	Test passed

(c) Testing the methods of CustomerUI to pay for their membership

Operation performed	Condition Tested	Actual Result
Accessing the Payment tab for Credit	The Customer’s “Credit” tab is selected and is displayed on the window	Test Passed

Accessing the Payment tab for Debit	The Customer’s “Debit” tab is selected and is displayed on the window	Test Passed
Confirm the Payment using credit	User’s hasPaid boolean variable set to true	Test passed
Confirm the Payment using debit	User’s hasPaid boolean variable set to true	Test passed
Leave input text fields blank and confirm payment as customer	Error message displayed on window	Test passed
Pay for subscription when hasPaid is already true	Message displayed under text fields reads “payment for this month has already been made”	Test passed

(d) Testing the methods of IOWork to serialize and deserialize to maintain persistent user objects as well as sort users by attended classes and missed payments

Operation performed	Condition Tested	Actual Result
Deserialize a User	Program is opened, The .ser files for the users are deserialized and arrayList “userList” is instantiated with new user objects.	Test passed
Serialize a User	Program is terminated, and created Users are saved into .ser files within “users” directory	Test passed
Sort arrayList of users by their attendance of classes	arrayList Users is sorted with the first element being the user with the highest value for attendedClasses	Test passed
Sort arrayList of users by their number of missed payments	arrayList Users is sorted with the first element being the user with the highest value for missedPayments	Test passed

(e) Testing the methods of LoginInterface to open different user interfaces depending on the login information entered into the text fields

Operation performed	Condition Tested	Actual Result
Login as Customer with valid username and password	Login window closes and main Customer interface window is displayed.	Test Passed

Login as Treasurer with valid username and password	Login window closes and main Treasurer interface window is displayed.	Test Passed
Login as a Coach with valid username and password	Login window closes and main Coach interface window is displayed.	Test Passed
Login using invalid credentials or blank text fields	Login window remains open and no new window is displayed.	Test passed

(f) Testing the methods of User to receive messages / notifications and pay

Operation performed	**Condition Tested**	**Actual Result**
Concatenate incoming message from coach	Notifications string size is greater than 0	Test passed
Pay for subscription	Check that the user’s boolean variable hasPaid is true	Test passed

Results: Using the java programs created with JUnit we were able to test our classes and their methods. We were able to see how many tests were executed and whether they gave the expected output specified in each test case.

Testing for User.java:
4 tests: success: 4, failure: 0, error: 0