David Westgate

Term Project: *Task Management Application*

Test Plan Document

Table of Contents

1 Introduction

This document outlines the testing procedure that will be used on the Task Management Application, a Java program. First a testing description will be introduced which will define the aspects of the program to be tested, as well as to give a tentative testing schedule. Afterwards, a Unit testing approach will be outlined by defining each testable unit and exploring the tests to be performed on each, as well as test cases.
Lastly, the integration testing stage will be explored to describe the tests required when combining the units.

1.1 Purpose and Scope

Application testing is a vital pre-release requirement for any software project. No developer is capable of writing bug free code given the nearly infinite nature of user input to most modern and complex applications. Because there will exist a fair amount of user interaction and input in the Task Management Application, thorough testing must occur to ensure as many as possible scenarios of interactions are bug free, and do not produce errors. This document aims to comprehensibly outline the testing process I will use on the implemented project.

1.2 Target Audience

This document is primarily intended for myself, who is expected to do the entirety of this projects testing. I would find having this document on hand to be very useful during the testing process. By outlining the entire testing process, I will be able to carry out the testing quickly and efficiently. If the scope of this project was to extend beyond the classroom, others who were to perform testing may reference this document to see which tests were performed and how, to build on my tests as well as see what tests have not been done.

1.3 Terms and Definitions

For simplicity, some unfamiliar terms and acronyms may be used:

- TMA: Task Management Application
- IDE: Integrated Development Environment
- Junit: A unit testing framework for java programs.

2 Test Plan Description

In this section, the test plan description shall give a general overview of what testing this project intends to accomplish. Specifically here, the scope of the testing process will be described and how certain tests will cover specific areas of the project.
Likewise, a testing schedule will be established for testing efficiency. Lastly, release criteria for the application will be evaluated. This acknowledges the realistic limits of testing of testing and how the apply to this project.

2.1 Scope of Testing

As described by the design document, the TMA is first broken down into two main modules. The first is the user login/registration interface, and the second is the task managing component. Of each module here, I will derive four units. These units shall be, user registration, user login, task creating, and task transition. The primary tests to perform for the first 3 units will be validating/simulating text input. I shall make sure the text input is accepted and relatively bug free for the given criteria and reject for improper criteria. A number of possible transition can be made for a given task, so these must be tested as much as possible in combinations.

2.2 Testing Schedule

The final project is due 02 June 2016, so a testing schedule will help keep development focused.

- Friday, 27 May: Complete first iteration of User login/registration module
- Saturday 28 May: Test and take notes on first two units
- Sunday 29 May: Reiterate first module. Begin development of second module
- Monday 30 May: Complete development of second module. Test/Report on next two units
- Wednesday 01 June: Reiterate second module. Integrate all four units/two modules. Test and report on integration.

- Thursday 02 June: Make final iteration on integrated project. Last minute changes. Project report and submit.

2.3 Release Criteria

This project must meet the absolute minimum as required by the project description. Though the project description does not discuss acceptable fault tolerances, this could be interpreted as simply the failure to include anything not specifically required. For example, being unable to have a properly functioning password encryption, inability to edit tasks, and a lack of aesthetical design could be considered accept faults. Minimally, I must meet the simple criteria of a user login and registration system, adding and deleting tasks, and task transitions within the three lists.

3 Unit Testing

This section will outline each testable unit of the TMA, their purpose, how I will go about testing them, and what test cases are relevant. As stated above, there will be a user Registration unit, A user login unit, a task creation unit, and a task transition unit. The intended functionality of each unit is generally self explanatory. It is noteworthy that the first two units, are fundamental entities of the account management module, and the latter two are of the task management module. As a java project, JUnit, will be deployed extensively for unit testing.

3.1 User Registration

The user registration unit is the first things a new user will deal with before anything else. The requirements of this unit are very straightforward. It must allow a user to register with a unique username and password. Therefor, a user may not register and already registered username, and any newly unique username must be accepted. Unlike in some systems, this application does not require user-name restrictions, or matching password confirmations, so the test cases will not cover those concepts. Rather, the test cases will assert a valid registration for any new username registration attempt, and assert a failed registration attempt for the same username twice in a row.

3.1.1 Valid User Registration

The valid user registration should be asserted by the following inputs:

- “A new unique username”, “any password”, TRUE

This is a very simple test case. This assertion should hold under any circumstance when there is a new and unique username being user as input. Performing this assertion a number of times should hold that any username could be validated.

3.1.2 Invalid User Registration

Asserting the case of an invalid user registration attempt is also very straightforward and should be performed directly after a valid user registration assertion.

- "A username that was already validly registered", "any password", FALSE

If assertion holds for every username that was already registered, then the entire user registration module can be considered working and without unacceptable faults.

3.2 User Login

A user will encounter the user login unit for every session of using this application. The user will expect that by entering the same set of credentials, they will gain access. I will need to assert that for successful login attempts, this program enters the state reflecting such attempts. It will also need to be asserted that for any failing login attempt, the program does not enter that state. It is noteworthy that while usernames may not need to be case sensitive, passwords should be.

3.3 Making tasks

When a user creates tasks, he or she will be prompted to enter a block of text to describe the task. There exist no other particular restrictions to how a user could enter a task. Testing this unit will involve running test cases that simulate task creating, by calling the function within the Click Listener of the make task button. It is expected that any possible new task will be successfully created. I shall assert this in JUnit.

3.4 Task Transitions

Task Transitions are the movements of a user created task from one stage to another. A specific transition function will take each task status container as a parameter, and will remove the task from the first and place it in the second. There are 4 possible transitions among the three status containers. I shall test each transition with both a new task, and a task that has already been transitioned before. A simple assertion that the task no longer

exists in the list it was from, and now exists in the list it went to, will verify a successful transition.

4 Integration Testing

After each unit is tested individually, and found to be in good working order, It is best to performs tests on the integrated units. Doing these tests in this sequence shall help me better understand where the bugs might be found, if any tests were to fail. I shall first test the integration of each pair of units into their respective modules, and then the integration of the modules into the whole application.

4.1 Integrating Units One and Two

Successful integration of the user registration and login unit should provide a complete working module. If the following cases are all asserted, this has been successful

- "Register user a", "Login user a", TRUE
- "Register user a", Login user b", FALSE;
- "Register user a, Register user b", "Login User a", TRUE.

These assertions essentially show that any user who has registered, should be able to log in. Likewise, any user who has not registered should not be able to log in. It should not matter if other users have registered in between the time a particular user has registered and attempted to log in.

4.2 Integrating Units Three and Four

Aside from a simple delete function, the integration of the latter two units will provide a working task management process. Some of the following sequences are test cases to help ensure successful integration. Functions that should resolve as "True" are valid transitions. My program will not allow an invalid transition, such as between lists 1 and 3, so I cannot test that.

- "Add a task", "Transition to list 2", "Transition to list 3" TRUE
- "Add a task" TRUE
- "Add a task", "Transition to list 2", "Transition to list 1" TRUE

- “Add a task.”, “Transition to list 2” “Transition to list 3”, “Transition to list 2”, “transition to list 1” , TRUE

4.3 Integrating Modules One and Two

Upon successful test of all else, testing the integration of both modules will provide a satisfactory application. These tests may not be as easily performed by Junit frameworks. The following tests will show the quality of integration

- Register my account. Log in, make a task in each category and quit. Log back in and make sure my account is persistent.
- Make changes in my account, and quit. Log into another account. Quit and Log back into my account. Make sure my account is accurately persistent.
- Delete a task, log out and log back in. Make sure task is deleted