SE 3XA3: Test Plan
PGH Software Solutions

Team #, Team Name
Pratyush Bhandari, bhandarp
Gazenfar Syed, syedg1
Hamid Ghasemi, ghasemih

October 25, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	**Version**	**Notes**
Oct 25	1.0	Completing parts 1 and 2 of document
Date 2	1.1	Notes

1 General Information

1.1 Purpose

The purpose of testing our project is to allow us to build more confidence in knowing that our product works as it is intended to.

1.2 Scope

This test plan will provide us with a method to exhaustively test our MovieGuide re-implementation. Our re-implementation must be able to display, sort and update a list of movies from an API. As well as display summaries, trailers and ratings for each respective movie. The objective of our test plan is to prove that the MovieGuide re-implementation can successfully perform these main tasks.

1.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
PoC	Proof of Concept
GUI	Graphical User Interface
SRS	Software Requirements Specification

1.4 Overview of Document

This document aims to formalize an exhaustive test plan for the MovieGuide re-implementation. It aims to arrange testing activities and delegate testing responsibilities among team members. As well as setting the initial groundwork for testing to begin, the document will also cover the specifics of testing certain characteristics of the code and will define what testing methods are to be used to perform each test.

Table 3: **Table of Definitions**

Term	**Definition**
Automated Testing	A series of tests that are run automatically usually by a testing framework
Unit Testing	A Method of testing centred on testing functions or methods
Integration Testing	A method of testing that tests the entire system at once
Static Testing	A method of testing that is done without executing the code
Dynamic Testing	A method of testing that is done while the code is executed

2 Plan

2.1 Software Description

The software that is being developed will allow users to navigate through a list of movies loaded from an API. The user can then click on any movie to view a summary, a rating and watch a trailer. In technical terms, when a movie is loaded from the API, it is parsed into a java Movie object with summary, rating, trailer and title data. This data is then accordingly shown to the user through the GUI buttons.

2.2 Test Team

The team members involved in testing the application are Pratyush Bhandari, Gazenfar Syed and Hamid Ghasemi. Testing responsibilities will be split equally among team members.

2.3 Automated Testing Approach

Our approach to automated testing is bottom-up testing, which in essence tests each component at a lower hierarchy before going on to test the components higher in the hierarchy. As such, there will be a focus on unit testing all of the methods within classes first. Once there is certainty about the

reliability of methods in the classes, the focus will shift to integration testing, which will test the functionality of each class. Finally there will be full application testing, which will involve testing the GUI of the application to perform actions that may involve the use of multiple classes.

2.4 Testing Tools

The primary tool to be used for running all of the tests for the MovieGuide re-implementation will be JUnit; this framework will allow us to use Automated Testing to test our program.

2.5 Testing Schedule

https://gitlab.cas.mcmaster.ca/syedg1/MovieGuide_PGH/blob/master/BlankProjectTemplate/ProjectSchedule/3XA3%20Gantt.pdf

3 System Test Description

3.1 Tests for Functional Requirements

3.1.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

Input:

Output:

How test will be performed:

3.1.2 Area of Testing2

...

3.2 Tests for Nonfunctional Requirements

3.2.1 Area of Testing1

Title for Test

1. test-id1

   Type:

   Initial State:

   Input/Condition:

   Output/Result:

   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.

   Initial State:

   Input:

   Output:

   How test will be performed:

3.2.2 Area of Testing2

...

3.3 Traceability Between Test Cases and Requirements

4 Tests for Proof of Concept

4.1 Area of Testing1

Title for Test

1. test-id1

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

2. test-id2

   Type: Functional, Dynamic, Manual, Static etc.
   Initial State:
   Input:
   Output:
   How test will be performed:

4.2 Area of Testing2

...

5 Comparison to Existing Implementation

6 Unit Testing Plan

6.1 Unit testing of internal functions

6.2 Unit testing of output files

7 Appendix

This is where you can place additional information.

7.1 Symbolic Parameters

The definition of the test cases will call for SYMBOLIC_CONSTANTS. Their values are defined in this section for easy maintenance.

7.2 Usability Survey Questions?

This is a section that would be appropriate for some teams.