SE 3XA3: Test Report
Movie Guide

Team 35, PGH Software Solutions
Hamid Ghasemi and ghasemih
Pratyush Bhandari, bhandarp
Gazenfar Syed, syedg1

December 5, 2018

Contents

List of Tables

List of Figures

Table 1: **Revision History**

Date	Version	Notes
Dec3	1.0	Completed parts 1, 2, and 3
Dec4	1.1	Completed parts 4 and 7

1 Functional Requirements Evaluation

The aim of these tests is to make sure that user is able to use the software according to the given requirements. These tests will include sorting testing, searching teasting, accessing the summary, rate and date released, and trailer testing.

Test Name: FRSR
Results: The user is able to sort movies based on rate, popularity, and released date.

Test Name: FRSE
Results: The user is able to search a movie by using movie name.

Test Name: FRAC
Results: The user is able to access movie's summary, rate and released date.

Test Name: FRTR
Results: The user is able to watch a movie trailer by simply selecting movie's trailer.

2 Nonfunctional Requirements Evaluation

2.1 Usability

2.1.1 GUI Testing

The Graphical User Interface (GUI) was tested by 7 students from Mac who are not in Software Engineering but they were interested to see our apps (From other programs) to reflect the technological experience of the potential users for this program. All the participants observed the timing it took them to perform the requested tasks, and they see the difficulty of the software, and how long would take them to get their hands around the app. At the end of the section they all gave us feedbacks about our apps and evaluate the difficulty and performance of our application.

Test Name: SS-1
Results: All participants were able to successfully complete installing the program to their personal phone (they all had android phone) within 2-3 minutes.

Test Name: SS-2
Results: All participants were able to successfully complete the task of searching a favorite movie by inputing the movie's name .

Test Name: SS-3
Results: All participants were able to successfully the task of sorting movies based on rate, released date and popularity.

Test Name: SS-4
Results: All participants mentioned that all the tasks which they have done above were really easy and so simple at the same time. Their feedback was that our app is easy to install, easy to use, understandable and in overal simple. They were all satisfied but some of them suggested that the software can be improved. For instance one of participants said that software can have an option to sort the movies which will come out within 6 months. This way people can realize which movies are coming out. Moreover, they enjoyed using the app since it is touchscreen so it makes a connection with the user.

2.1.2 Media Output Testing

The program was installed into phone with an android system operator. The results of attempting to launch the program were noted.

Test Name: SS-5
Results: The program installed on phones that has android OS and there wasn't any problem.

2.2 Performance

2.2.1 Screen Speed Performance

The preformance was calculated based on how long the app takes to perform user requests and if the app run smoothly.

Test Name: SS-6
Results: The time to perform user requests was less than 2 second from all participants.

2.3 Output

2.3.1 Media Output Testing

The time between finding a movie and retrieving the data from dataset was calculated. In addition to that the aim of this test was to check if the output is the expected output. And each time different movie requested by participants to ensure that data, and output file is consistent. Plus, the calculation between sorting movies were noted down in this section.

Test Name: SS-7
Results: Movies were seatrched by participants retrieved within approximately 3 seconds.

Test Name: SS-8
Results: Sorting movies based on rate, popularity, and date released took about 5 seconds and it outputed list of movies based on those categories.

3 Comparison to Existing Implementation

In our app, we used retrofit which is one of the library of the android studio. It is a powerful framework for authenticating and interacting with api and sending network requests. Retrofit made our app easier to code since it can be used to create json object and then we used that in our implementation, using it in our java implementation. This is one of the main difference between our code and existing implementation.

4 Unit Testing

4.1 Load Movies API Unit Test

4.1.1 Control

Retrieve data from the API

4.1.2 Input

Query used to retrieve data from the database

4.1.3 Output

A list containing Movie objects. Each Movie object will contain parsed data from the API.

4.1.4 Procedure

A get request was made to the API using a query passed in as input. The data is parsed into Movie objects and a list of Movie objects is created. Finally, the list of Movie objects was iterated through and movie names were printed out to verify that data for that movie has been passed.

4.2 Sorting Movies Unit Test

4.2.1 Control

Returns a list of movies sorted based on input by making a query to the API.

4.2.2 Input

Select to sort by popularity, rating, or release date.

4.2.3 Output

The list of movies will be returned in sorted order based on input.

4.2.4 Procedure

A sorting option will be selected. The software will display the movies in sorted order. The information of the movies will be manually checked to determine whether the sorting has been done correctly.

4.3 Movie Trailers Unit Test

4.3.1 Control

Utilizes HTML5 to load the trailers into the program.

4.3.2 Input

The play button for the video is clicked.

4.3.3 Output

The program plays the selected video.

4.3.4 Procedure

A movie was selected from the displayed list of movies. One of the movie trailers was played. The response of the program was checked to determine whether the movie trailers functionality is functioning as expected.

5 Changes Due to Testing

6 Automated Testing

7 Trace to Requirements

7.0.1 Load Movies Unit Test (Tests FRAC and FRSE)

This test covers the following requirements (Requirements numbering corresponds to the requirements document): FR1, FR2, FR3

It covers FR1 because when it loads the movies, it parses the data of each movie and prints the movie name to verify that the software is able to provide a synopsis about the movies.

It covers FR2 because it loads movies using the search query passed. The movie names are printed out to show that only movies matching the search query were obtained from the database.

It covers FR3 because no movie names will be printed upon running the test if there are no search hits. This means that no movies will be displayed, which communicates that there were no movies found in the database matching the search query.

7.0.2 Sorting Movies Unit Test (Test FRSR)

This test covers the following requirements (Requirements numbering corresponds to the requirements document): FR5

It covers FR5 because it tests whether the movies have been sorted correctly.

7.0.3 Movie Trailers Unit Test (Test FRTR)

This test covers the following requirements (Requirements numbering corresponds to the requirements document): FR1, FR4, FR6

It covers FR1 because it tests whether the movie data, which includes movie trailers, is loaded successfully into the program.

It covers FR4 because it tests whether the video playback works as expected by playing the videos.

It covers FR6 because it requires the videos to be successfully retrieved from the internet, so that they can be displayed to the user.

8 Trace to Modules

9 Code Coverage Metrics