COMP 214 Testing Report

Cong Bao, Hongchi Chen, Jinke He, Xiang Li, Zekun Wang, Yuan Zhu
Artificial Intelligence Group Project
Team 1

INTRODUCTION

In general, there were three parts of testing, which were Component Testing, Integration Testing (including case based Testing) and Performance Testing. The Component Testing was the testing of each component (subsystem), which was done by the developers of the subsystems. The Integration Testing was the testing of the system that was integrated from subsystems. Case based Testing was used in the testing of the whole system. Finally, there was a Performance Testing that tested the performance of the website.

COMPONENT TESTING

By definition, Component Testing is the testing of individual program components. Basically, the project can be divided into four subsystems, which are server, database, recommendation algorithm and UI. Each subsystem can be further divided into smaller components, for example, UI was made up of several web pages. This testing should be done before the integration testing.

In our project, each subsystem was designed and implemented by one or two members. The members who implemented the subsystem were also the ones who took charge of the Component Testing of the subsystem. Most of the component testing can be done during the implementation phase but this is not the case for some components that have functionality which can only be tested with other components.

For the recommendation algorithm, it only provided one functionality, which was generating a list of recommended movies given the user's previous ratings. According to the design, the recommendation program should receive ratings from the server and return the results back to the server. To test it, however, the developers manually created several lists of user ratings and inputted them into the recommendation program separately. The results were checked according to the interface design of this subsystem.

According to the testing design, the coupling within the algorithm also needed to be checked. The way to check it was to check whether different parts of the algorithm has been divided into different functions according to their logical functionality. For example, the function of computing the Cosine Similarity between two n-dimensional vector should be made into a separate function. Moreover, there should not be too many global variables because they may increase the coupling of the algorithm. According to the result, the whole algorithm had been divided into separate functions according to their logical functionality and no global variable was used so it passed the testing.

For the database, the component testing was divided into three parts. Firstly, the developers tested whether all the data from data set has been loaded into the database. This was done by checking whether the total number of rating records in the database was equal to that in the data set. After making sure that the database and the data set had the same number of ratings, the developers randomly picked up 30 ratings from the data set and checked whether they also existed in the database. The second part was to test modifying the database. The operations included creating a new user account in user table, editing the profile of a user, retrieving user information, creating a new rating record in rating table and create a new movie record in movie table. These operations were tested manually but according to the design, all the operations would finally be performed automatically by the Java program. Therefore, the final step was to test whether the database could be connected to and manipulated by a Java program. To test it, the developers created a new Java program and connected it to the MySQL database and performed the same operations again using Java code.

There was not too much Component Testing in the server because server itself did not have too much functionality. Since the project used Tomcat Apache as the web server, the developers tested whether it could run well by visiting the specific port and the expected web page was shown. After the developers bound a domain with the server, there was also a testing of whether the server could be accessed by the domain.

For the user interface, before it was uploaded to the server, the testing was mainly static testing, which focused on the layout, style, picture, font, etc. Dynamic Testing of the user interface was done after the UI was uploaded to the server and connected to the database.

INCREMENTAL INTEGRATION TESTING

By definition, Integration Testing is to test the complete system based on the specification. However, since the project contained four subsystems, it was hard to integrate all of them by one step. To reduce this problem, the team used Incremental Integration Testing, which was to take a testing whenever a new subsystem was added to the whole system and finally test the complete system after all the subsystems were integrated.

The first Integration Testing was to test the integration of the UI and database. Since the project was a web Java project based on tomcat, after the integration of UI and database, it could be run locally like a real website. Therefore, the integration testing then was to test the whole website and all the functionality apart from the recommendation functionality and the remote access functionality. The testing of the website included all the operations that a user was able to do and it was a case based testing, which would be described in detail in the next section.

The Second Integration Testing was done after the Recommendation Algorithm was added to the whole system. The results showed that there were some problems with the scheduling, that was when the website should ask the recommendation program to generate new recommendation lists for the users. The main problem was that the server which the Recommendation System was built on was in a wrong time zone. It turned out that the server had a default time zone which was not the same with which the developers were in. After setting the server of Recommendation System into the right time zone, the recommendation functionality performed well in local and succeeded to generate recommendation lists for the testing users.

Finally, there was an Integration Testing for the complete system which had been uploaded to the user. This testing is a fully back-box testing and was done remotely. Testers did all the test cases again but this time the recommendation functionality was also included. Starting from a non-registered user, the testers registered a new account and rated several movies and finally got a recommendation list. To make sure that the website worked well in all platforms, the website was tested in Mac, Linux and Windows operating systems, using different browsers such as IE, Chrome, Firefox, Edge, etc. The result showed that the website worked well in all the browsers except IE browsers that were earlier than IE 9.0. By checking the console of the browser, a problem was found that there were warnings about accessing external resources that were not from https websites. The solution was to check every link to the external resources and changed http into https at the head of them.

CASE BASED TESTING

Case based Testing was used in website testing. The test plan and result are shown below.

Name of case	Description	Input	Action	Expected Output	Actual Output	Succeed/Fail
SignUpSucceed	Fail to sign up	A random user name and a weak password '123'	Sign up	Failed	Failed (password too short)	Succeed
SignUpFail	Succeed to sign up	A random user name and a strong password 'HJK1996!'	Sign up	Succeeded	Succeeded (a new account was created in database)	Succeed
LoginSucceed	Succeed to login	Correct user name and password	Sign in	Succeeded	Succeeded (user can view his profile)	Succeed
LoginFail	Fail to login	A user name and a wrong password	Sign in	Failed	Failed (wrong password)	Succeed
RateMovie	Rate movies that were chosen randomly	Stars between 0 and 5	Submit rating	Ratings loaded into database	Ratings could be seen in the database	Succeed
AddMovieToFavorite	Add movies to one's favorite list	Movies that were chosen randomly	Click "Add to favorite" Button	Added into the favorite list	The chosen movies could be seen in the favorite list and the changes were uploaded to the databse	Succeed
Logout	User logout	None	Click "Logout" Button	User logout	User logout (cannot view his profile any more)	Succeed
SearchForMovie	User search for a movie by a title	Movie titles	Search	Lists of movies	Lists of movies (empty list for a non-existing movie)	Succeed
EditProfile	User edit his profile	New email addresses and passwords	Submit changes	Information changed	The changes can be seen in the profile page and they were uploaded to the database	Succeed

ViewRecommendation	System generates the recommendation list and User can view it the next day	No inputs needed if the user has already rated at least one movie	View Recommendation List	Lists of recommended movies	Lists of recommended movies (each list contains 50 recommended movies) and they were uploaded to the database as well	Succeed

PERFORMANCE TESTING

The first part of the Performance Testing was to test the response time of the website. According to Firefox Network Monitor, the response time to load the home page was 0.04 seconds. It took 0.64 seconds to load a movie preview page.

The second part was to test the waiting time of searching for a movie. According to Firefox Network Monitor, searching for a movie with its title like "Iron Man" took around 0.44 seconds. However, searching for content that was not a title of a movie might take much longer. For example, it took 20 seconds to search for the letter "c".

The final part was to test how much time it would take to generate a recommendation list. According to the result, the time it took to generate a recommendation list for a user depended on the number of movies that the user had rated. The team picked up 30 users, five of them rated 5 movies, five of them rated 10 movies, five of them rated 15 movies, five of them rated 20 movies, five of them rated 25 movies and the rest five of them rated 30 movies. After generating recommendation lists for all of them and recording the average times it took, a statistical table was produced as below.