COMP214 Requirement Specification Report

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

1 PROJECT DESCRIPTION

1.1 INTRODUCTION

Data are playing important roles in people's daily life. It is becoming a trend that people using data to help with decision making. Recommender system, which is a system that providing a list of recommendations to users, is a typical case of data utilizing. Nevertheless, the algorithms used in common recommender systems are usually based on categories and tags, which require considerable costs in categorization and tag management [1]. The movie recommender system, which will be developed in our project, will use the Collaborative Filtering (CF) algorithm instead to provide recommendations based on users' ratings. More specifically, rather than using an average value of ratings as a reference, CF try to find users who share similar interests by analysing the previous ratings of a user [2]. Based on the ratings made by the users, the system will predict the ratings of a user on the movies he/she has not seen yet. After that, a list of recommended movies will be generated according to the predicted ratings [3].

1.2 TARGET CUSTOMER

There are three kinds of customers that will make use of the movie recommender system.

- People who love to watch movies but have no idea what movies he/she may like.
- People who enjoy sharing their preference of movies with others.

1.3 OBJECTIVES

The main purpose of our project is to develop a movie recommender system based on CF algorithm. A new algorithm to predict movie ratings will be introduced by combining the two algorithms. Additionally, to support the recommender system, a database system and a web server system will be established as well. Eventually, customers will be able to interact with the movie recommender system through a web site. After register to be a member, user will be provided with a personal recommended movies list at regular interval according to their previous ratings. Some additional functions, such as searching for a movie, home page random movie recommendation will also added to keep integrity of system.

2 STATEMENT OF DELIVERABLES

2.1 ANTICIPATED DOCUMENTATIONS

There are several documentations anticipated to be produced in the project, which are shown in the list below.

- System Specification
- Architecture Documentation
- Software Specification
- User Documentation
- Java Doc

2.2 ANTICIPATED SOFTWARE

2.2.1 Main features

The project is aimed to produce a movie recommendation system based on CF algorithm. A list of its main features is shown below.

Essential features:

- User can rate a movie.
- The system will give user a recommended movies list at a particular time point according to the previous ratings of user.

Desirable features:

- User can click random movie on home-page to rate.
- User can search for movies by keywords.

Detailed features and functionalities will be given in the functional and non-functional requirements.

2.2.2 Boundary diagram

The boundary diagram is shown in Fig. 1.

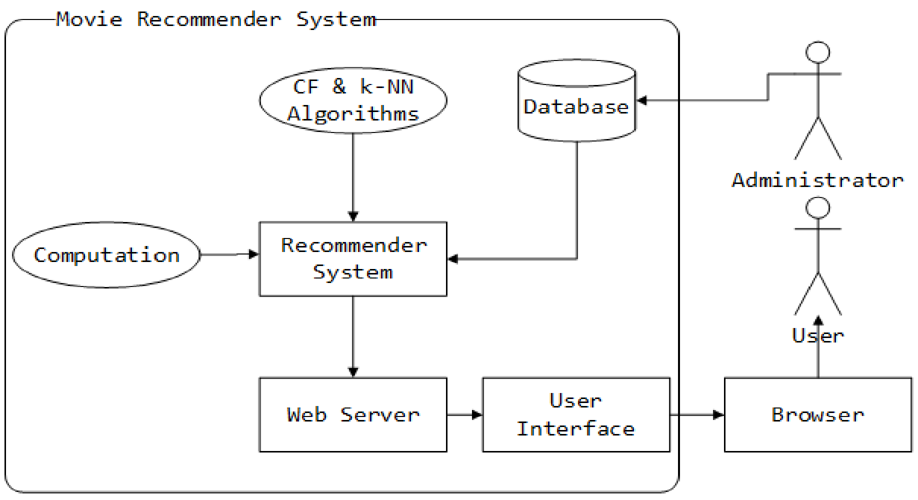


Fig. 1. Boundary Diagram

2.2.3 User views

The user views are given in the form of user cases as shown in Fig. 2.

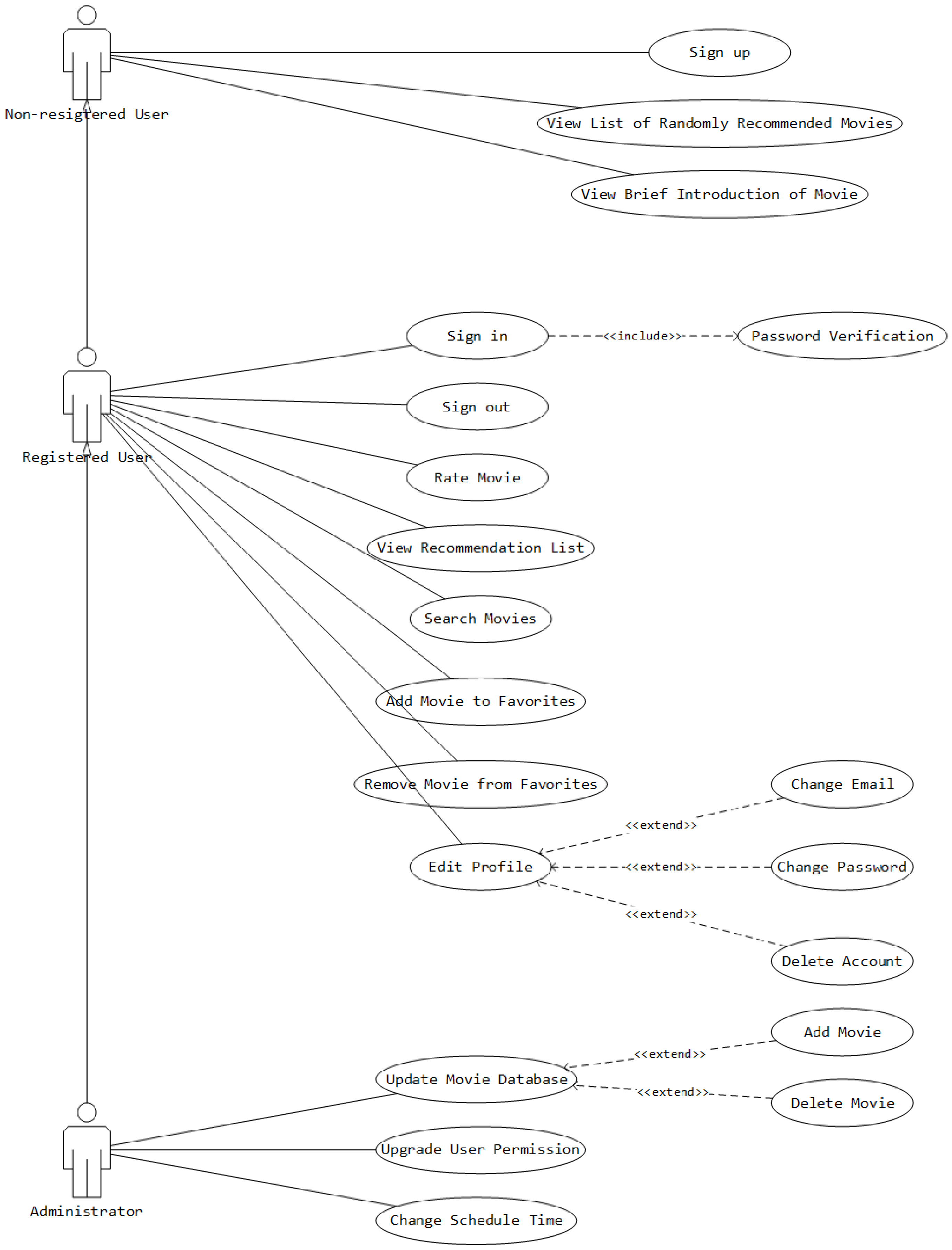


Fig. 2. Use Cases Diagram

2.2.4 Form-based specification

The specifications of this project are conducted in a form-based approach. The form-based specifications are shown from Table. 1 to Table. 20.

User	Non-registered User
Function	Sign up
Description	New user enters Account_name, Email_address and Password to create a new account
Inputs	Account_name, Email_address, Password
Source	
Outputs	Success (log in the newly created account) / Fail (Invalid information input)
Destination	User information database
Action	The UI asks the user to enter a string account name, correct e-mail address and a string password. If the format of the e-mail address is correct (Including a "@" and ".com"), then add the information pair to the database and automatically turns to the page with the status of already logged in.
Include	
Pre-condition	The user is a new user without an account
Post-condition	The new account is loaded into database and the user logs in the website
Extend	

Table. 1. Function: sign up

User	Non-registered User
Function	View List of Randomly Recommended Movies
Description	Once a user viewing the website, the website will present the movies randomly.
Inputs	
Source	Movie database
Outputs	Movie_Poster
Destination	
Action	System will show a list of movie poster on index.
Include	
Pre-condition	The user is currently in the homepage of website.
Post-condition	
Extend	

Table. 2. View List of Randomly Recommended Movies

User	Non-registered User
Function	View Brief introduction of movie
Description	The user can get a brief introduction of the conductor, movie stars, published year and a poster pf the movie they clicked.
Inputs	
Source	Movie database
Outputs	Movie information: 1. Genre 2. IMDB Rating

	3. Actors 4. Released date 5. Runtime 6. Poster 7. User Ratings
Destination	Website UI output
Action	Search the corresponding movie information from movie database, then jump into the movie’s own webpage with the corresponding data.
Include	
Pre-condition	User clicks on the movie to view brief introduction
Post-condition	A new webpage opens
Extend	

Table. 3. Function: view brief introduction of movie

User	Registered User
Function	Sign in
Description	Registered user logs in using user id and password
Inputs	Email_address and Password
Source	User inputs form keyboard
Outputs	Ture if Email_address and corresponding Password can be found in database otherwise False
Destination	Main control loop
Action	If user inputs can be matched in database, log in with user inputs. Otherwise a message that prompts user to re-enter user id and password will be sent.
Include	Password Verification
Pre-condition	User has not signed in
Post-condition	Registered user logs in
Extend	User inputs are not found in database

Table. 4. Function: sign in

User	Registered User
Function	Password Verification
Description	Registered user get password verified before logged in
Inputs	Password
Source	
Outputs	Main_Page
Destination	Main control loop
Action	Compare password with database, jump to main page, log in successfully if password get verified
Include	Sign in
Pre-condition	User has entered password
Post-condition	Log in success of denied
Extend	

Table. 5. Function: Password verification

User	Registered User
Function	Sign out
Description	Registered user logs out from the system
Inputs	
Source	
Outputs	A message showing the success of sign out
Destination	Main control loop
Action	User_Id logs out
Include	
Pre-condition	User has signed in
Post-condition	Registered user logs out
Extend	

Table. 6. Function: sign out

User	Registered User
Function	Rate A movie
Description	Registered user rates a movie
Inputs	The number of stars (from 0 to 5)
Source	User input
Outputs	
Destination	Main control loop
Action	Create a new rating record and load it into database
Include	
Pre-condition	User has signed in
Post-condition	User's rating on the movie is created and loaded into the database
Extend	

Table. 7. Function: rate a movie

User	Registered User
Function	View recommendation list
Description	Register user can view the recommendation list about movies shown by the system.
Input	
Source	User database
Output	Recommendation list about movies
Destination	Website UI output
Action	Click button 'recommendation movie list' and open a new page, drag scroll bar to view the list of movies which recommended by the system and set preferences about the type of movies
Include	
Pre-condition	User has signed in

Post-condition	

Table. 8. Function: view recommendation list

User	Non-registered User / Registered User
Function	Search for movie
Description	User can search for a movie by keyword
Inputs	
Source	Movie database
Outputs	Movie of highest corresponded name.
Destination	
Action	The website has an input box located at top right of main-page to enter.
Include	
Pre-condition	
Post-condition	
Extend	

Table. 9. Function: Search for movie

User	Registered User
Function	Add movie to favourites
Description	Registered users can add movies to the list of favourite movies
Input	Movie
Source	User input, Movie database
Output	selected movies added to the favourite list
Destination	
Action	Click button on the movies shown on the recommendation list and add to the favourites list
Include	
Pre-condition	user has registered for the system and logged in successfully
Post-condition	• User has rated at least 10 movies • The system has given the recommendation list • User opens the recommendation list

Table. 10. Function: add movie to favourites

User	Registered User
Function	Remove movie from favourites
Description	Registered users can remove movies from the list of favourite movies
Input	Movie
Source	User input, Movie database
Output	selected movies deleted from the favourite list
Destination	

Action	Click button on the movies shown on the recommendation list and remove from the favourites list
Include	
Pre-condition	user has registered for the system and logged in successfully
Post-condition	• User has rated at least 10 movies • The system has given the recommendation list • User reviews the recommendation list

Table. 11. Function: remove movie from favourites

User	Registered User
Function	Edit Profile
Description	Registered users can edit their own user profiles
Input	User information
Source	
Output	The user profile with personal information of user on it
Destination	
Action	Input information about the user according to the form given by the system to edit the personal homepage of a register user
Include	
Pre-condition	User has signed in
Post-condition	The profile of the user is updated

Table. 12. Function: Edit Profile

User	Registered User
Function	Change Email
Description	Registered user resets the his email address
Inputs	New_Email
Source	The Email of user
Outputs	The Email being deleted and replaced with New_Email.
Destination	
Action	After user enters new email address. The old email address will then be deleted
Include	
Pre-condition	User has signed in
Post-condition	The email address of the user is updated
Extend	Edit Profile

Table. 13. Function: Change Email

User	Registered User
Function	Change Password
Description	User can reset his password
Inputs	Origin_Password, New_Password
Source	The password of user

Outputs	The Password being deleted and replaced with New_Password.
Destination	
Action	After the user entering correct origin_Password, user get verified and enters new password. The old password will then be deleted
Include	
Pre-condition	
Post-condition	The password of the user is updated
Extend	Edit Profile

Table. 14. Function: Change Password

User	Registed User
Function	Delete account
Description	User deletes his user account
Inputs	Account_name, Email_address, Password
Source	User input
Outputs	A message showing the success of deletion
Destination	Main control loop
Action	Log out current account Delete all the information about this account in database
Include	
Pre-condition	User has signed in
Post-condition	User has signed out User account is deleted in database
Extend	

Table. 15. Function: Delete account

User	Administrator
Function	Update movie database
Description	Administrator updates the movie information in database
Inputs	Movie_Information
Source	User input
Outputs	
Destination	Main control loop
Action	Add new movie to the database or delete current movies in database
Include	
Pre-condition	
Post-condition	Movie database is updated
Extend	Add Movie, Delete Movie

Table. 16. Function: Update Movie Database

User	Administrator
Function	Add Movie

Description	Administrator adds a new movie to database
Inputs	Movie_Title, Movie_Year, IMDB_ID
Source	User input
Outputs	
Destination	Main control loop
Action	Add a new movie to database
Include	
Pre-condition	
Post-condition	A new movie entity is created in database
Extend	

Table. 17. Function: Add Movie

User	Administrator
Function	Delete Movie
Description	Administrator deletes a movie in database
Inputs	Movie_ID
Source	User input
Outputs	
Destination	Main control loop
Action	Delete Movie_Information in database
Include	
Pre-condition	Movie_Information exists in database
Post-condition	Movie_Information is deleted from database
Extend	

Table. 18. Function: Delete Movie

User	Administrator
Function	Upgrade user permission
Description	Administrator can upgrade a registered user to Administrator
Inputs	User_ID
Source	User database
Outputs	
Destination	User Datatbase
Action	Administrator upgrades a user account to Administrator
Include	
Pre-condition	Target user exists in database
Post-condition	Target user is promoted to administrator in database
Extend	

Table. 19. Function: Upgrade user permission

User	Administrator
Function	Change schedule time

Description	Administrator can change the schedule of recommendation algorithm
Inputs	Hour, Minute, Sceond
Source	System
Outputs	
Destination	System
Action	Administrator change the schedule time
Include	
Pre-condition	
Post-condition	
Extend	

Table. 20. Function: Change Schedule Time

2.2.5 Functional requirements

- An account is not required for users who only want to visit the home page of the website and view movie introductions.
- All users who haven't registered can register an account to become a registered with full access to all services within the website.
- All registered users could sign in with their user id and password.
- All registered users could delete their account if they want, after which they will become a non-registered user.
- All registered users could reset their email address if they want to have a new one.
- All registered users could reset their password if they want to have a new one.
- All registered users could view their profiles and edit them.
- All registered users could rate movies.
- The website should provide registered users with new recommended movies lists each day if they are not judged inactive user.
- The website should NOT provide registered users with new recommended movies lists each day if they are judged inactive user.
- An inactive user can be re-acted by rate arbitrary movie.
- The website should provide the function to search a movie.
- The website should be compatible with popular desktop browsers such as Chrome, Edge, Firefox, IE10+, Opera, Safari, etc.
- An initial administrator account should be set to update movie database.
- An Administrator account can promote a registered user to be administrator
- A friendly error page with help messages should be given if the website runs into problems.
- A user judgement should be conducted at 3 a.m. everyday.
- A new recommended movies list should be generated and show to user at 6 a.m. everyday.
- An account with 10 or more day not conduct any rate will be set to inactive

2.2.6 Non-functional requirements

- The time used to load each web page should not exceed 1 second in any condition.
- The size of each picture should not exceed 1MB.
- The website should bear a peak of 1,000 page views.
- The website should be delivered with a user specification.
- The website should be delivered with a license file.

- Using Java as the main server-side programming language.
- Using JavaScript as the main web page programming language, and jQuery as the main library.
- Using JSP as the dynamic page technique.
- Using Bootstrap 3 as the web page UI framework.
- Using Spring 4 and Apache Struts 2 as the MVC framework.
- Using Hibernate 4 as the ORM framework.
- Using Apache Tomcat 8.0 as the web container.
- Using MySQL 5.7 as the database server.
- Using UTF-8 as the page encoding format.
- Using MD5 and SALT as the encryption standard.
- Using MQTT as scheduler.
- The project is licensed under Apache License, Version 2.0.
- All user data should be encrypted before writing into database.
- The system shall not disclose any personal information about customers apart from their name and e-mail address and those that are essential to generate recommendation lists.

2.3 ANTICIPATED EXPERIMENTS

- Experiment 1

  Test recommendation algorithm in local command-line environment. The experiment should use multiple test cases and record the average response time of the system to decide the update frequency of the recommendation lists.

- Experiment 2

  Launch the web server online and test the usability of the web site. The experiment should be taken under different environments including Windows, OSX and Linux. Different web browsers such as IE11, Firefox, Chrome should be used to test the compatibility of the website.

- Experiment 3

  The final experiment will be testing the entire system. The entire process, including creating a user account, rating some films and receiving a recommendation list will be simulated. Students who are not the member of the project team will be invited to test the system.

2.4 METHODS FOR EVALUATION OF THE WORK

There are two methods to evaluate the work. The first is to choose a user from the existing data set which consists of the rating information of each user. Randomly select ten rating records of that user and enter them into the recommendation system. The system will offer a recommendation list based on existing ratings. Choose movies from the recommendation list that have been rated by the user and check whether these recommended movies truly got a high rating from the user.

The second method is to ask students who are not the members of the project team to use the website. After they try the system and practice the process of getting a recommendation list, feedbacks should be provided by them. The feedback should focus on whether they are satisfied with the result as well as whether the web site is user-friendly.

3 CONDUCT OF THE PROJECT AND PLAN

3.1 PREPARATION

The project will use a stable benchmark dataset from GroupLens as the basis of recommender system, which is collected from MovieLens. This Dataset includes 20 million ratings applied to 27,000 movies by 138,000 users. To reduce the size of dataset, the first preparation is to filter out all the information about movies that are released before 2000.

The second preparation is to search for the information about Collaborative filtering and KNN algorithm because in this project, they need to be implemented using Java.

The next preparation is to search for the information about the implementation of our web server, database, and website.

3.2 DESIGN STAGE

For the system design, Use Case Diagram, Interface Design, System Model Design, Data Structure Design, Interaction Chart, and Class Diagram will be used.

For the key algorithms and key functionalities, Pseudo-code will be written before their implementations.

For the database design, Data Dictionaries, Entity-relationship Diagrams and Table Structure Design.

For the UI design, Web Site Map Diagrams and User Interface Design will be used.

At the end of design stage, a report involving all the design methods above will be generated.

3.3 IMPLEMENTATION STAGE

The hardware this project is going to use:

A host. The server and website will be built on it.

The software this project is going to use:

- Java programming language.
- JavaScript programming language
- JSP dynamic page technique.
- Bootstrap 3 UI framework.
- Spring 4 and Apache Struts 2 framework.
- Hibernate 4
- Apache Tomcat 8.0
- MySQL 5.7

The first testing will be the test of the key algorithms. This testing can be carried out before the implementation of the web site. In a command-line environment, ten ratings of movies will be entered the recommender system and the recommender system should return a list of recommended movies in a reasonable time.

The second testing will be the test of database. This testing should focus on the reliability and usability of the database. It will include adding and updating movie information and creating and deleting user accounts.

The third testing will be the test of the whole website. This testing should simulate the whole process of using the recommender system.

In general, the plan of testing will follow the rule of Bottom-Up Testing.

3.4 TIME-TABLED SCHEDULE

The time-tabled schedule is realized in the form of Gantt Chart, which is shown in the Appendix.

3.5 RISK ASSESSMENT

Major Challenges:

- The implementation of optimized collaborative filtering algorithm and k-NN algorithm.
  Though the algorithms could be implemented, they may not be optimized. Extra time may be spent to find faster implementations of algorithms, which may influence the processing of original plan.
- Limited time to realize a complete system.
  The practicability of algorithms is based on a practical system. However, studying and developing of algorithms occupy more time resources than those of practical system. Extra time may be required to extend the original plan to ensure the integrity of system.
- Unfamiliar with new skills that required in the system.
  New skills may be required to implement algorithms and the system. However, a lack of knowledge in new skills may influence the efficiency of development. Therefore, the original plan may be modified to tolerant possible delay.

New Skills Required and Acquiring:

- The Map-Reduce model introduced in Java 8 to handle big data processing.
- Dynamic website realized by Java Servlet and JSP.
- MVC frameworks such as Spring and Struts.
- Object-relational mapping technique to link object-oriented language with relational database.
- JavaScript library such as jQuery for web page development.
- Web page UI framework such as Bootstrap.
- Version control system such as Git for project management.

These new skills could be acquired from official courses, official documents, published books, self-learning websites, technical blogs, etc.

4 REFERENCES

[1] N. Zheng and Q. Li, "A recommender system based on tag and time information for social tagging systems," Expert Systems with Applications, vol. 38, pp. 4575-4587, 4// 2011.

[2] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Adv. in Artif. Intell., vol. 2009, pp. 2-2, 200

[3] D. A. Adeniyi, Z. Wei, and Y. Yongquan, "Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method," Applied Computing and Informatics, vol. 12, pp. 90-108, 1// 2016.

5 BIBLIOGRAPHY

[1-12]

[1] C. Bauer, G. King, and G. Gregory, *Java Persistence with Hibernate*: Manning Publications Company, 2015.

[2] D. Brown, C. M. Davis, and S. Stanlick, *Struts 2 in Action*: Manning, 2008.

[3] T. C. Caputo, *Build Your Own Server*: McGraw-Hill/Osborne, 2003.

[4] S. Castano, *Database Security*: ACM Press, 1995.

[5] A. G, *Spring MVC: Beginner's Guide*: Packt Publishing, 2014.

[6] R. Haapanen, A. R. Ek, U. o. M. D. o. F. Resources, and M. A. E. Station, *Software and Instructions for KNN Applications in Forest Resources Description and Estimation*: Department of Forest Resources, College of Natural Resources and Agricultural Experiment Station, University of Minnesota, 2001.

[7] J. Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen, *Collaborative Filtering Recommender Systems*: Springer Berlin Heidelberg, 2007.

[8] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, pp. 76-80, 2003.

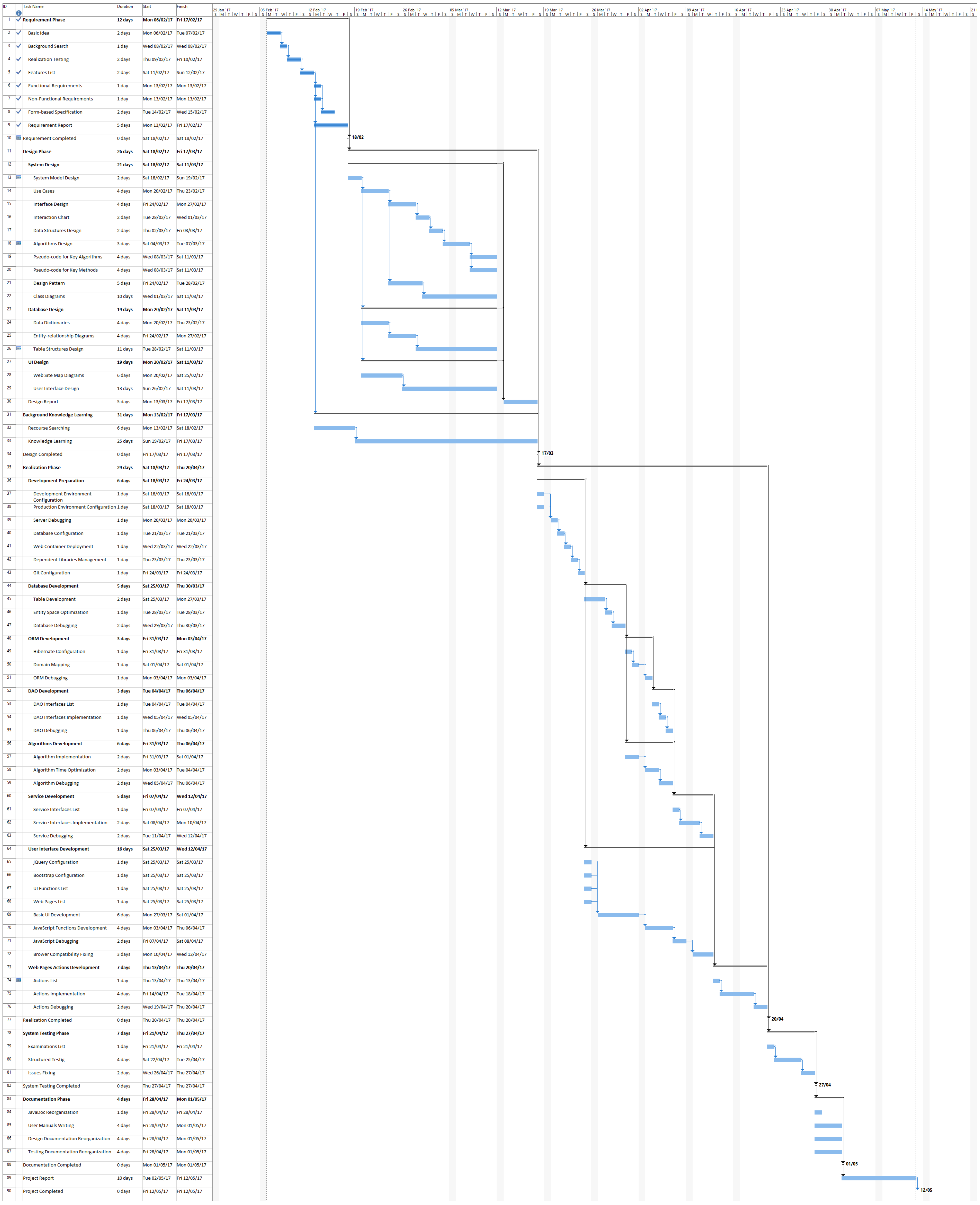
[9] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," presented at the ACM Hong Kong, 2001.

[10] S. Strunjas and U. o. Cincinnati, *Algorithms and Models for Collaborative Filtering from Large Information Corpora*: University of Cincinnati, 2008.

[11] X. Su and F. A. University, *Collaborative Filtering Using Machine Learning and Statistical Techniques*: Florida Atlantic University, 2008.

[12] R. G. Urma, M. Fusco, and A. Mycroft, *Java 8 in Action: Lambdas, Streams, and functional-style programming*: Manning Publications Company, 2014.

6 APPENDIX: GANTT CHART



ID		Task Name	Duration	Start	Finish
1	✓	**Requirement Phase**	**12 days**	**Mon 06/02/17**	**Fri 17/02/17**
2	✓	Basic Idea	2 days	Mon 06/02/17	Tue 07/02/17
3	✓	Background Search	1 day	Wed 08/02/17	Wed 08/02/17
4	✓	Realization Testing	2 days	Thu 09/02/17	Fri 10/02/17
5	✓	Features List	2 days	Sat 11/02/17	Sun 12/02/17
6	✓	Functional Requirements	1 day	Mon 13/02/17	Mon 13/02/17
7	✓	Non-Functional Requirements	1 day	Mon 13/02/17	Mon 13/02/17
8	✓	Form-based Specification	2 days	Tue 14/02/17	Wed 15/02/17
9	✓	Requirement Report	5 days	Mon 13/02/17	Fri 17/02/17
10	▦	Requirement Completed	0 days	Sat 18/02/17	Sat 18/02/17
11		**Design Phase**	**26 days**	**Sat 18/02/17**	**Fri 17/03/17**
12		**System Design**	**21 days**	**Sat 18/02/17**	**Sat 11/03/17**
13	▦	System Model Design	2 days	Sat 18/02/17	Sun 19/02/17
14		Use Cases	4 days	Mon 20/02/17	Thu 23/02/17
15		Interface Design	4 days	Fri 24/02/17	Mon 27/02/17
16		Interaction Chart	2 days	Tue 28/02/17	Wed 01/03/17
17		Data Structures Design	2 days	Thu 02/03/17	Fri 03/03/17
18	▦	Algorithms Design	3 days	Sat 04/03/17	Tue 07/03/17
19		Pseudo-code for Key Algorithms	4 days	Wed 08/03/17	Sat 11/03/17
20		Pseudo-code for Key Methods	4 days	Wed 08/03/17	Sat 11/03/17
21		Design Pattern	5 days	Fri 24/02/17	Tue 28/02/17
22		Class Diagrams	10 days	Wed 01/03/17	Sat 11/03/17
23		**Database Design**	**19 days**	**Mon 20/02/17**	**Sat 11/03/17**
24		Data Dictionaries	4 days	Mon 20/02/17	Thu 23/02/17
25		Entity-relationship Diagrams	4 days	Fri 24/02/17	Mon 27/02/17
26	▦	Table Structures Design	11 days	Tue 28/02/17	Sat 11/03/17
27		**UI Design**	**19 days**	**Mon 20/02/17**	**Sat 11/03/17**
28		Web Site Map Diagrams	6 days	Mon 20/02/17	Sat 25/02/17
29		User Interface Design	13 days	Sun 26/02/17	Sat 11/03/17
30		Design Report	5 days	Mon 13/03/17	Fri 17/03/17
31		**Background Knowledge Learning**	**31 days**	**Mon 13/02/17**	**Fri 17/03/17**
32		Recourse Searching	6 days	Mon 13/02/17	Sat 18/02/17
33		Knowledge Learning	25 days	Sun 19/02/17	Fri 17/03/17
34		Design Completed	0 days	Fri 17/03/17	Fri 17/03/17
35		**Realization Phase**	**29 days**	**Sat 18/03/17**	**Thu 20/04/17**
36		**Development Preparation**	**6 days**	**Sat 18/03/17**	**Fri 24/03/17**
37		Development Environment Configuration	1 day	Sat 18/03/17	Sat 18/03/17
38		Production Environment Configuration	1 day	Sat 18/03/17	Sat 18/03/17
39		Server Debugging	1 day	Mon 20/03/17	Mon 20/03/17
40		Database Configuration	1 day	Tue 21/03/17	Tue 21/03/17
41		Web Container Deployment	1 day	Wed 22/03/17	Wed 22/03/17
42		Dependent Libraries Management	1 day	Thu 23/03/17	Thu 23/03/17
43		Git Configuration	1 day	Fri 24/03/17	Fri 24/03/17
44		**Database Development**	**5 days**	**Sat 25/03/17**	**Thu 30/03/17**
45		Table Development	2 days	Sat 25/03/17	Mon 27/03/17
46		Entity Space Optimization	1 day	Tue 28/03/17	Tue 28/03/17
47		Database Debugging	2 days	Wed 29/03/17	Thu 30/03/17
48		**ORM Development**	**3 days**	**Fri 31/03/17**	**Mon 03/04/17**
49		Hibernate Configuration	1 day	Fri 31/03/17	Fri 31/03/17
50		Domain Mapping	1 day	Sat 01/04/17	Sat 01/04/17
51		ORM Debugging	1 day	Mon 03/04/17	Mon 03/04/17
52		**DAO Development**	**3 days**	**Tue 04/04/17**	**Thu 06/04/17**
53		DAO Interfaces List	1 day	Tue 04/04/17	Tue 04/04/17
54		DAO Interfaces Implementation	1 day	Wed 05/04/17	Wed 05/04/17
55		DAO Debugging	1 day	Thu 06/04/17	Thu 06/04/17
56		**Algorithms Development**	**6 days**	**Fri 31/03/17**	**Thu 06/04/17**
57		Algorithm Implementation	2 days	Fri 31/03/17	Sat 01/04/17
58		Algorithm Time Optimization	2 days	Mon 03/04/17	Tue 04/04/17
59		Algorithm Debugging	2 days	Wed 05/04/17	Thu 06/04/17
60		**Service Development**	**5 days**	**Fri 07/04/17**	**Wed 12/04/17**
61		Service Interfaces List	1 day	Fri 07/04/17	Fri 07/04/17
62		Service Interfaces Implementation	2 days	Sat 08/04/17	Mon 10/04/17
63		Service Debugging	2 days	Tue 11/04/17	Wed 12/04/17
64		**User Interface Development**	**16 days**	**Sat 25/03/17**	**Wed 12/04/17**
65		jQuery Configuration	1 day	Sat 25/03/17	Sat 25/03/17
66		Bootstrap Configuration	1 day	Sat 25/03/17	Sat 25/03/17
67		UI Functions List	1 day	Sat 25/03/17	Sat 25/03/17
68		Web Pages List	1 day	Sat 25/03/17	Sat 25/03/17
69		Basic UI Development	6 days	Mon 27/03/17	Sat 01/04/17
70		JavaScript Functions Development	4 days	Mon 03/04/17	Thu 06/04/17
71		JavaScript Debugging	2 days	Fri 07/04/17	Sat 08/04/17
72		Brower Compatibility Fixing	3 days	Mon 10/04/17	Wed 12/04/17
73		**Web Pages Actions Development**	**7 days**	**Thu 13/04/17**	**Thu 20/04/17**
74	▦	Actions List	1 day	Thu 13/04/17	Thu 13/04/17
75		Actions Implementation	4 days	Fri 14/04/17	Tue 18/04/17
76		Actions Debugging	2 days	Wed 19/04/17	Thu 20/04/17
77		Realization Completed	0 days	Thu 20/04/17	Thu 20/04/17
78		**System Testing Phase**	**7 days**	**Fri 21/04/17**	**Thu 27/04/17**
79		Examinations List	1 day	Fri 21/04/17	Fri 21/04/17
80		Structured Testig	4 days	Sat 22/04/17	Tue 25/04/17
81		Issues Fixing	2 days	Wed 26/04/17	Thu 27/04/17
82		System Testing Completed	0 days	Thu 27/04/17	Thu 27/04/17
83		**Documentation Phase**	**4 days**	**Fri 28/04/17**	**Mon 01/05/17**
84		JavaDoc Reorganization	1 day	Fri 28/04/17	Fri 28/04/17
85		User Manuals Writing	4 days	Fri 28/04/17	Mon 01/05/17
86		Design Documentation Reorganization	4 days	Fri 28/04/17	Mon 01/05/17
87		Testing Documentation Reorganization	4 days	Fri 28/04/17	Mon 01/05/17
88		Documentation Completed	0 days	Mon 01/05/17	Mon 01/05/17
89		Project Report	10 days	Tue 02/05/17	Fri 12/05/17
90		Project Completed	0 days	Fri 12/05/17	Fri 12/05/17