Requirements Analysis Document
Grupp 22

Adrian Lindberg
adrlin@student.chalmers.se
Institutionen för Informationsteknik

Jonatan Nylund
nylundj@student.chalmers.se
Institutionen för Informationsteknik

Jonathan Sundkvist
jonran@student.chalmers.se
Institutionen för Informationsteknik

Sebastian Nilsson
sebnils@student.chalmers.se
Institutionen för Informationsteknik

Friday 27th May, 2016

1 Introduction

A group of friends gathers a Sunday evening to watch a movie. Everything is set up, snacks and soda have been purchased, the best spots in the couch have been fought over and the only task that remains is selecting a movie to watch. 2 hours later, after endless discussions of what movie on IMDB Top 250 to watch, the friends have finally decided on 22 Jump Street for the fifth time. Since everyone have to get up early to go to school in the morning they stop the movie halfway through and decide to finish it another time.

1.1 Purpose of application

To minimize the excise and time waste of choosing a movie to watch by getting dynamic and personalized recommendations of recently watched movies from friends and a larger community.

1.2 General characteristics of application

The application will be a database driven Android application with tab navigation to navigate between the different views. The main views will be lists containing personalized content depending on which other users the user has added as friends. In general the GUI will have a simple look and feel in order to minimize excise.

1.3 Scope of application

A single user application with no direct interactions with other users (e.g. messages). The user can choose to add another user as a friend but this is not necessarily a two-sided relationship.

1.4 Objectives and success criteria of the project

Navigation inside the application should be possible without any major latency issues. The design should support a user behaviour where the user have found a movie to watch within 30 seconds from opening the application but also encourage the user to come back to rate the movie afterwards in order to contribute to the community.

1.5 Definitions, acronyms and abbreviations

- GUI, Graphical User Interface.
- Java, platform independent programming language.
- Android, mobile operating system (OS).
- Android Studio, official IDE for Android development.
- IDE, Integrated Development Environment.

2 Requirements

2.1 Functional requirements

The user should be able to:

- Navigate between the GUI:s views.
- Scroll through the lists inside the different views.
- Click on a movie list item in order to see a detailed view of selected movie.
- From within a detail view rate and add/remove selected movie from the user's playlist.
- Search for movies and other users.
- Add/remove other users as a friend.

2.2 Non-functional requirements

2.2.1 Usability

Low excise and easy access to relevant information is highly prioritized. An ideal user session should not exceed 30 seconds inside the application whereof it is required of the GUI to be simple and clear in its presentation of information.

2.2.2 Reliability

NA

2.2.3 Performance

Any latency should not be significant enough to discourage the user from further usage of application. Responsiveness while navigating between tabs and scrolling in lists is of high priority in this regard.

2.2.4 Supportability

The application will have a well defined interface towards the database, making it possible to replace the database. The model must be clearly structured in order to make an eventual implementation for another OS or platform only a matter of adjusting the code for another code language/style.

2.2.5 Implementation

The application will be developed for the Android platform and with the option of using different types of databases.

2.3 Packaging and installation

2.3.1 Without Android Studio

The application files can be cloned or downloaded from the public repository at Github: `https://github.com/lindbergan/tda367projektarbete`.

1. To run the application Gradle (the tool which builds and packages the application) needs write-permission in your file system:

```
$ chmod +x gradlew
```

2. The Android SDK needs to exist in the path variable. If it's not already, the following snippets adds the SDK.

```
$ export ANDROID_HOME=/Users/[username]/Library/Android/sdk
$ export PATH=${PATH}:$ANDROID_HOME/tools:$ANDROID_HOME/platform-tools
```

3. Build the application with Gradle:

```
$ ./gradlew assembleDebug
```

4. The Android SDK platform tools need to be installed on the system. To install the app on the emulated or real connected device:

```
$ adb install app/build/outputs/apk/app-debug.apk
```

5. Find the FlickPicker app on the device and open it.

2.3.2 With Android Studio

In Android Studio, click *File* → *Open* in the top menu. Select the folder which the files were cloned from the Github repo. Gradle will build automatically. Click *Run* → *Run* 'app' in the top menu. Pick a virtual device to install the app on. Android Studio will then launch the virtual device and install the app. When that's done the app will launch.

2.3.3 Legal

NA

2.4 Application models

2.4.1 Use case model

A diagram that visualizes the five most important use cases can be found in the APPENDIX.

2.4.2 Use cases priority

1. Search for a movie - Searches for movies to add to your collection
2. Click on a movie - Shows more information about the movie
3. Rate movie - Increases user score and removes the movie from Watchlist and Recommendations.
4. Add movie to Watchlist - Adds the movie to Watchlist and updates the Watchlist view
5. Search for user - Search for users and shows the add / remove friend functionality
6. Add / remove user as a friend - Changes the friendrelation between two users

2.4.3 Domain model

See APPENDIX

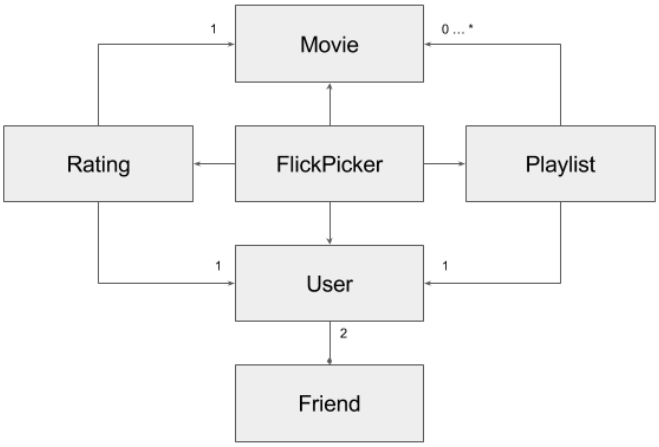
2.4.4 User interface

The application will use a fixed GUI built in Android Studio using the components and widgets provided by its library. The GUI consists of a tab bar footer and a header with an area in between designated for showing the various content. There are no options to change the theme of the application.

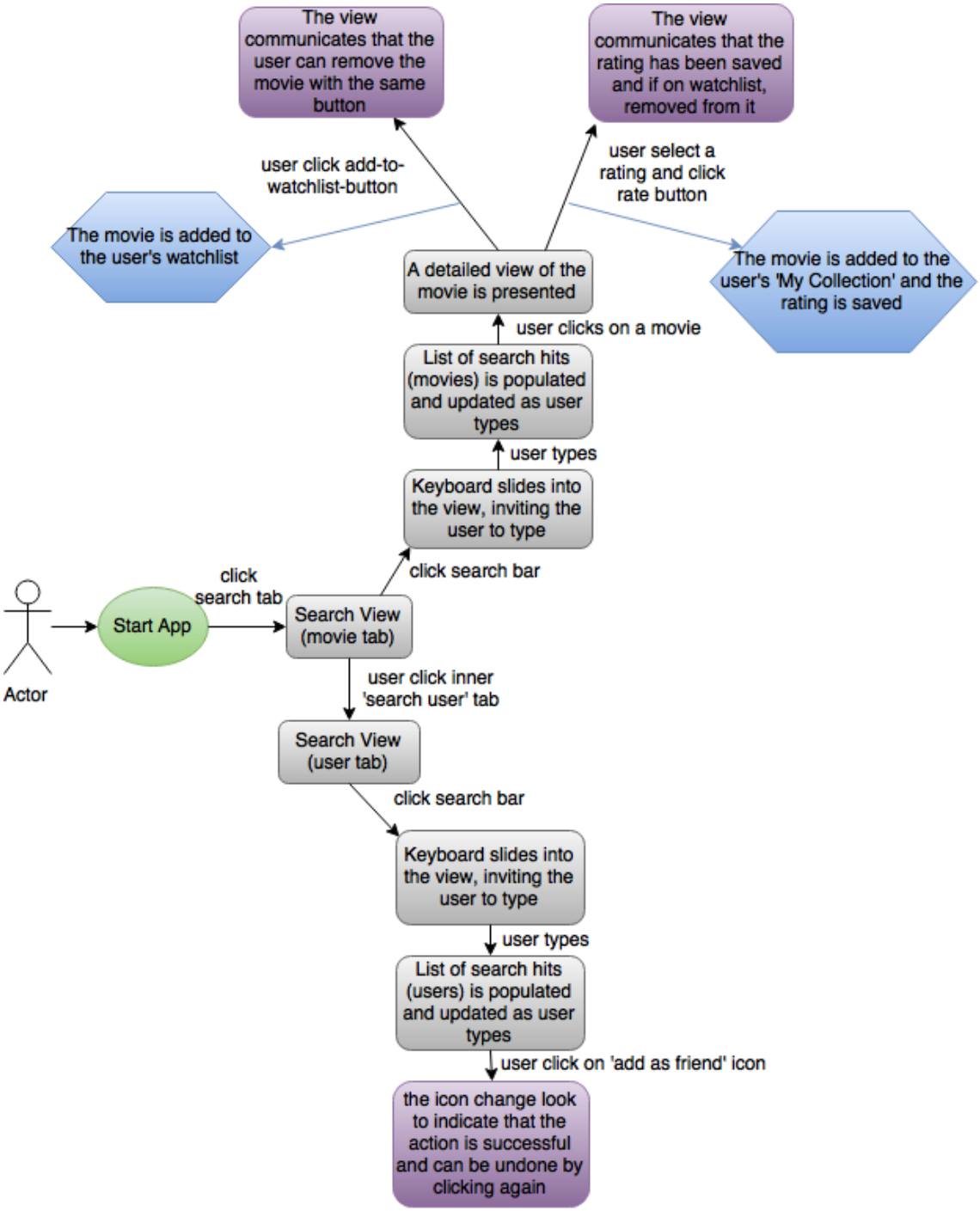
2.5 References

3 APPENDIX

3.1 Domain Model



3.2 Use Case UML



3.3 Use Case: Search movie

	Search Movie (in "Search" tab)
Summary	A search field for movies can be reached in either the "Search" tab or the "My Collection" tab. The functionality is similar for both search fields (but serve different purposes). This Use Case refers to the search field in the "Search" tab.
Priority	High
Extends	
Inlcudes	
Participators	The user, Application

	1.0 Normal Workflow	
ID	Actor	System
1	Clicks on the search field	
2		Keyboard pops up
3	Starts writing the title of the movie	
4		Displays results based on the exact serch term. Only results that start with those exact letters are presented (that is - "ro" will result in Rocky, but not Troy). A delay is implemented in order to make the search moore smooth
5	Clicks to confirm the search when finished	
6		The system hides the keyboard and the user is presented with a list of all search results that match the search term

	3.1 Alternative Workflow	
ID	Actor	System
1	Erases all search letters	
2		Keyboard is still active but instead of no movies presented, all movies are presented in the search results screen

	5.1 Alternative Workflow	
ID	Actor	System
1	Clicks outside the keyboard instead of confirming the search	
2		Keyboard hides and the user is still presented with the results that match the search term

3.4 Use Case: Search user

	Search User
Summary	The search field for users can be reached in the "Search" tab. The functionality is similar to "Search Movie"
Priority	High
Extends	
Inlcudes	
Participators	The user, Application

	1.0 Normal Workflow	
ID	Actor	System
1	Clicks on the search field	
2		Keyboard pops up
3	Starts writing the name of the user	
4		Displays results based on the exact search term. Only results that start with those exact letters are presented (that is - "ro" will result in Robin, but not Caroline). A delay is implemented in order to make the search moore smooth
5	Clicks to confirm the search when finished	
6		The system hides the keyboard and the user is presented with a list of all search results that match the search term

	3.1 Alternative Workflow	
ID	Actor	System
1	Erases all search letters	
2		Keyboard is still active but instead of no users presented, all users are presented in the search results screen

	5.1 Alternative Workflow	
ID	Actor	System
1	Clicks outside the keyboard instead of confirming the search	
2		Keyboard hides and the user is still presented with the results that match the search term

3.5 Use Case: Rate movie

	Properties
Summary	Inside a movie's detail view, the user want to rate the movie by selecting a number of stars and click the rate button.
Priority	Medium
Extends	
Inlcudes	
Participators	The user, Application

	1.0 Normal Workflow	
ID	Actor	System
1	Clicks on a star	
2		The selected star and the ones to its left are filled.
3	Clicks on rate button	
4		The rating is saved to the database and the rate button change looks to indicate that a rating have been submitted for the movie.

3.6 Use Case: Click on a movie

	Properties
Summary	The user clicks on a movie item in a list of movies. Th application changes view to a larger one with more detailed information about the movie.
Priority	High
Extends	
Inlcudes	
Participators	The User, Application

	1.0 Normal Workflow	
ID	Actor	System
1	Clicks on a movie in a list	
2		Opens a modal page with information about the movie

3.7 Use Case: Add / Remove Friend

	Properties
Summary	User adds another user as a friend. This hides the AddFriend button and shows RemoveFriend button. If clicked on RemoveFriend button user removes the other user as a friend.
Priority	High
Extends	
Inlcudes	
Participators	User, Friend, System

	Add Friend	
ID	Actor	System
1	User presses the AddFriend button	System adds the selected user as a friend
2		Hides the AddFriend button and shows the RemoveFriend button

	Remove Friend	
ID	Actor	System
1	User presses the RemoveFriend button	System removes the selected user as a friend
2		Hides the RemoveFriend button and shows the AddFriend button

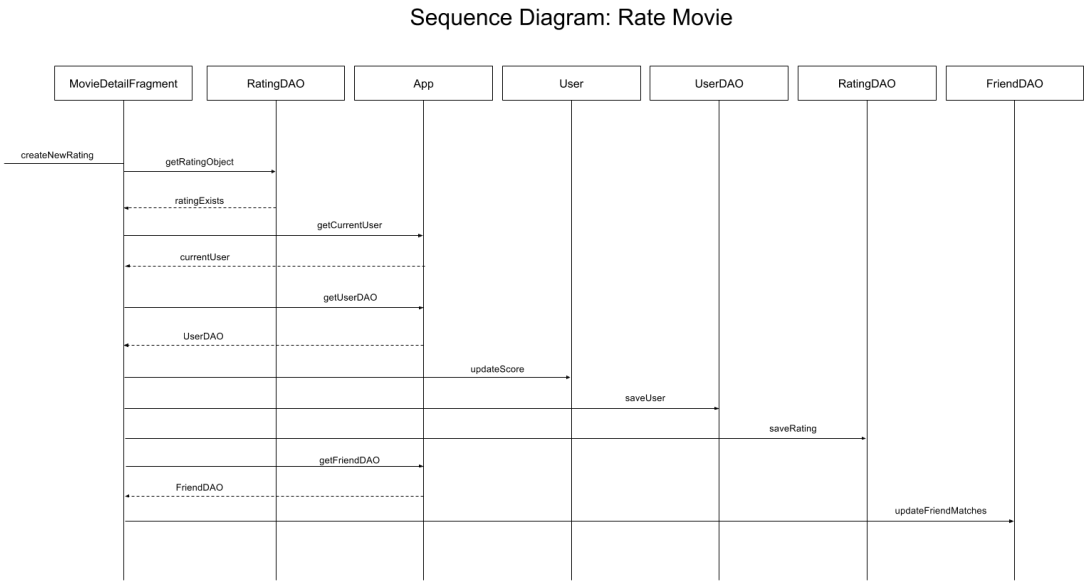
3.8 Use Case: Add / Remove Movie to / from Watchlist

Properties	
Summary	User clicks the Add Movie to Watchlist button (+). Movie is added in Watchlist section. This hides the (+) button and shows the Remove Movie from Watchlist button (-). Movie is now removed from the Watchlist section.
Priority	High
Extends	
Inlcudes	
Participators	User, Playlist, System

	Add Movie to Watchlist	
ID	Actor	System
1	User clicks the (+) button.	System adds the movie to Watchlist section.
2		Hides the (+) button and shows the (-) button.

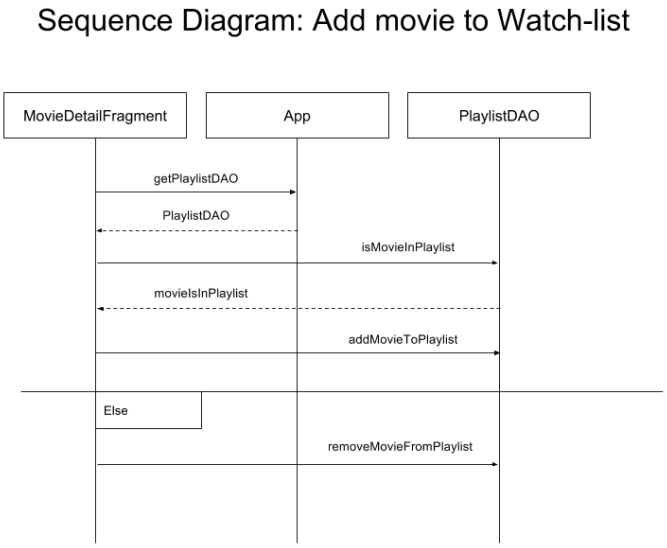
	Remove Movie from Watchlist	
ID	Actor	System
1	User clicks the (-) button.	System removes the movie from Watchlist section.
2		Hides the (-) button and shows the (+) button.

3.9 Sequence Diagram: Rate



Larger version: `http://i.imgur.com/awE5m85.png`

3.10 Sequence Diagram: Add to Watch-list



Larger version: `http://i.imgur.com/OXcvz9c.png`