Movie Match for Android

A Penny for Your Thoughts Ltd.

Matthew Penny matt.penny82@gmail.com

**Initial Project Planning**

1. *Assess the size, scope, and complexity of the project, and establishes procedures to support subsequent activities*

The size, scope and complexity of the Movie Match Application was difficult to decipher at first. When starting out, the API call was originally a source of worry until initial research showed that Netflix had deprecated their API and Disney and Prime Video had never made one. As such, we were able to find a couple of different APIs that would be more than enough to satisfy the needs of our application. We ended up choosing Streaming Availability API by Movie of the Night as they had compiled the services we needed and offered enough parameters to satisfy our needs. The major downfall of this API was the limited number of calls for the free option. Once this had been determined, we started to look for the different system requirements that would be required for us to implement the swiping style of cards that different dating applications have made so popular over the last 10 or so years. After this portion of the research was completed, we started to look into creating a wireframe to assess the number of activities, helper classes and fragments that would be needed to successfully complete the capstone requirements.

Now that the size and scope had been tentatively outlined, we started to wonder about the complexity of the project. There will be a minimum of eight activities and fragments as well as multiple API calls and three tables in the database that will be created using Firebase. We will also be using an OTP and hyperlink in an SMS message to allow users to link up when not in the same viewing area. With these different technologies(?), we were able to ascertain that the project will be quite complex and will take at least the 300 hours that were outlined in the requirements. We will be using GitHub as our VCS to stay in contact with the college and all administrators involved.

All the above indicates to us that we will have to adhere closely to the schedule that was outlined in our documentation. Allowing for breaks, time off, illnesses and any other issues that may arise, we will be planning weekdays to start from 8:30am until 6pm and will follow the Pomodoro focus method of working with 45-minute work periods, 15-minute break periods and 1 hour long breaks every 4 complete periods. Weekends will be used to reflect on the coming week and to clean up any possible deliverables, uploads to the VCS and creating test cases.

1. *Analysts identify detailed system requirements*

Having researched the different dating applications that use the swiping method to obtain matches as well the different streaming sites that we were looking to implement in this project, we determined that as far as minimum <look for wording on Android here> version will be as follows:

* To implement the application’s swiping method, the minimum requirements are **iOS 13.0 and up, Android 7.0 and up, as well as the latest versions of all major web browsers (Chrome, Firefox, Safari, Edge etc.).**

**For the different streaming services:**

* **Netflix** – available on Android phones and tablets running **Android 2.3 or later**. Current application version requires **Android 5.0 or later.** Older devices may not be able to download the application from the Google Play Store, although these devices may be able to get a version directly from Netflix. **Optional Picture in Picture is available to Android 8.0 and up only.**
* **Disney+** - Android phones and tables running **Android 5.0 or later are needed** to run the current application version. For those using Chromecast as a streaming device can cast from an Android device. Likewise, you can cast to devices such as Android TV.
* **Amazon Prime Video** –Android phones and tablets running **Android 5.0 and up are needed** to run the current application version.

1. *Requirements capture and modelling undertaken*

Without a set of stakeholders to assist in creating a list of functional and non-functional requirements, we decided to turn to people in our circle who thought the idea of this application had merit and used their input to create our requirements. When discussing the application, we came up with 3 sets of requirements: functional, non-functional, and individual requirements. The first two fall under the normal scope of any software development planning and the last falls under a separate scope of what requirements will be required by Matthew Penny, as he is the individual that is attempting to accomplish this undertaking. The different sets are as follows:

**Individual Requirements –** The planning and research behind the application, starting from the proposal that was put before Adin Ashby of Vanier College to the final presentation. Choosing an API that covers all the streaming services as outlined, as well as which database software to use, the UI/UX design as well as the decision to pivot after the mid-term meeting1. Creating and adhering to a schedule that would allow for planning, testing, and implementing user stories. During the programming sessions over the span of approximately 4 weeks, we saw that there could be some advanced features that could be implemented after more experience is gathered (Application learns from swipes and suggests similar movies upon return, for example).

**Functional Requirements –** There are many functional requirements that were needed to implement an application of this scope. There had to be a reliable, easy to use, and secure CRUD setup for the potential end-users, as well as at least two third-party register options (Google and Meta). <<<NEEDS WORK>>>

**Non-Functional Requirements –** We believe that many of the best applications have very similar non-functional requirements as these are what end-users have come to expect from high level applications. These requirements are as follows:

**Availability –** requirements for continuous running, minimum idle time.

**Reliability –** operation recovery, behaviors in case of alarm status, automatic restart.

**Scalability –** application should have ways to expand its system and avoid adversely affected performance.

**Security –** application operation and use of safety requirements related to access control, private data processing and sharing and external attack risk reduction.

**Usability –** applications ease of use, a user-friendly UI as well as a seamless UX.

**Extensibility –** required in case of the need to add new functional requirements.

**Modelling –** During our time at Vanier College, we had a course that taught us about the Software Development Life Cycle (SDLC) and the different models that companies, and groups use to plan, create, and showcase their projects. As per these teachings, we decided that the Movie Match application wouldn’t fit as well with two of the models that we had experience with (Waterfall and Agile). Because of this, we decided upon an Iterative model as this model allowed us to create a working template and continually add to it, adapting to changing needs as the project continued. As well, this allowed Adin to see progress in each of the commits as they were added to our VCS2.

1. *User stories, separated into sections of coding*

We were tasked to create at least 30 different User Stories for the application, and we decided that the easiest way to create these stories was to look at the application in stages and see what a typical end-user wants from each of these. At first, we created the Login and Registration Activities and so the early User Stories were:

1. As a Regular User, I want to be able to register my email, name, and phone number to the application’s database.
2. As a Regular User, I want to be able to login to the application.
3. As a Regular User, I want to be able to logout of the application.
4. As a Regular User, I want to be able to login to the application using a third-party source.
5. As a Regular User, I want to be able to update my information in the application’s database.
6. As a Regular User, I want to be able to access my account without logging in every time.
7. As a Regular User, I want to be able to upload an image to the application’s database.
8. As a Regular User, I want to be able to show the image that I have downloaded.
9. As a Regular User, I want to be able to delete my account.

As we progressed through the programming portion of the application, the next step was the API call to show the movies and series of different streaming services. These User Stories were as follows:

1. As a Regular User, I want to be able to select the streaming service.
2. As a Regular User, I want to be able to select from movies or series.
3. As a Regular User, I want to be able to select the genre of series or movies.
4. As a Regular User, I want to be informed of when I match on a movie or series.
5. As a Regular User, I want to be able to refresh the movies or series on the screen.
6. As a Regular User, I want to be able to undo an erroneous like or dislike.
7. As a Regular User, I want to be able to easily obtain more information on a movie or series.

The next part of the program was to implement Matches on movies or series with other end-users. The User Stories that arose from this portion were:

1. As a Regular User, I want to be able to see my Matches immediately after receiving one.
2. As a Regular User, I want to be able to send a message to one of my Matched users.
3. As a Regular User, I want to be able to share to social media from the Matches screen.
4. As a Regular User, I want to be able to start my movie or series from the Matches screen.

System Architecture Diagram

Sec

**Footnotes**

1 Original planning was for couples or friends in the same room could use the application to decide on something to watch from the desired streaming service. At the midterm meeting, Adin mentioned matches with other users as per an actual dating or meeting application. I pivoted then to create the matching algorithm along these lines.

2 As per told to Adin in a MIO from the school account, I had issues twice with not being able to commit work to the VCS and the files were deleted twice. Once I found out that these commits were being used to calculate hours, I retrieved the files, and they will be added in a zip folder as part of the final project.