Designing an Android Movie recommendation app based on genre and year of release

Student name and number: Alan Cherian C0073682

Supervisor: Dr Daniel Nesbitt | Date: 20/09/21
Word Count: 15888

Declaration

This dissertation is submitted to Newcastle University for the final module of the Computer Science MSc.

This thesis has not been submitted for any degree or other purposes. I declare that this dissertation represents my own work except where otherwise stated.

Abstract

Recommendation systems have become increasingly prevalent as data continues to be the world's most valuable resource. This data is often utilised to build models of consumer habits to create systems which are more tailored and appeal to consumers in a more customised fashion. The current movie recommendation systems are yet to tackle this with the level of accuracy that is requested from users. As a result, there is often too much time spent contemplating what movies to watch and hence users will interact less with these systems. Therefore, creating a movie recommendation system can not only allow the user to spend less time browsing this can be used to help benefit the producer of the film as it allows greater exposure and rewards movies of a higher calibre when it comes to ratings.

Acknowledgement

I would like to express my gratitude and sincere thanks to my project supervisor Dr Daniel Nesbitt for his support and guidance. Many thanks for giving me time and advice during my dissertation. Your feedback was invaluable

Table of Contents

List of Figures

List of Tables

1.0 Introduction

Ofcom estimates the time spent on entertainment to result in “a total of almost 45 hours a week, and a rise of almost a third on last year” [1]. Although this was taken during the peak of Covid in April, the post-lockdown data suggests 55% of the same watch time will continue to exist in the future. As we continuously seek more personalised and tailored approaches to technology, there is a growing consensus that “hyper-personalisation is becoming very important” [2] and will be the biggest asset in the entertainment industry.

Currently, with the rise of online streaming services such as Netflix and Amazon Prime Video, these streaming services are creating sophisticated algorithms to entice customers to stay on their services for longer periods. However, upon research, there is yet to be a separate application recommending movies exclusively. The closest application to offering this service is WatchBuddy [3] that operates using their own AI to suggest movies, based on selecting choices at the start by swiping right or left at a selection of movies. However, this application has many bugs and does not seem to operate. There are also existing applications that enable consumers to track their movie history, for example, Hobi and Cinemaniac [4]. These applications allow users to rate the movie, discover new and trending movies, and communicate with an online community of avid movie consumers to discuss opinions. However, despite offering these services, there is a clear market gap in offering a tailored movie suggestion based on IMDB ratings and mood.

It is important to draw inspiration from these existing products, particularly those currently successful ones, so perhaps I will incorporate some of the features depending on what consumers would prefer.

The idea behind this app is to minimise the amount of time spent searching for what movie to watch. The role of the app is to tailor suggestions given the individual’s preferences on previously watched movies or their current mood.

1.2 The reasons for this choice

The motivation stems from the numerous hours spent with friends and family when deciding what to watch together. From discussing with other peers, this is a prevalent issue amongst households. The user experience will also be important as the app has to be intuitive and easy to use, as I hope to provide a recommendation with a high level of accuracy. I plan on creating a separate system that will be much simpler in design and refining to only top-rated movies.

From the preliminary survey conducted a 100% of users would use this app. Hence, it is apparent that this service is application users would use as the frustration of searching for long periods to find the right movie is felt across the demographic of 18–34-year-olds.

1.3 Expected outcomes

I hope to produce a functioning app that provides a range of movie suggestions tailored to the individuals’ requirements. I also hope to bring a high level of accuracy with the

recommendations and a good user experience with the app. I will ensure I complete a survey reviewing the app to ensure consumers are happy with the product and it offers a chance to provide any issues with the product.

1.4 Project aim

I aim to build an android app in Java that will allow users to be provided with a range of movie recommendations. This will be based on either their current mood or previous watch history. The target audience will predominantly be "18-34 as 70%" [5] of these adults are currently subscribed to a streaming service. Hence, this would be the most pertinent age demographic.

1.5 Project objectives

- Research and analyse the existing Android apps that provide a similar service and analyse the features and drawbacks of these applications
- To create a functioning Java android application that operates smoothly with no bugs
- To solidify Java programming knowledge and build upon existing knowledge from previous programming modules
- To create a system that uses IMDB API to filter movies of rating 7 or higher
- To create additional filters such as genre and year of release
- To base recommendations on either movie that have been previously watched or current mood
- To develop an understanding of how to use Android Studio along with all the functions
- To allow the consumer to create an account
- Could potentially also recommend TV series
- Evaluate and test the produced system using surveys and various testing methods, including unit tests and system tests.

1.6 Outline

By following software engineering aspects to produce the system, an agile process model will be chosen. To achieve the proposed objectives, the project will go through some stages that will be discussed in the next chapters as outlined below:

Chapter 2 will be looking at existing related works about current applications in more depth, the features used, and examining their issues.

Chapter 3 will present the activities taken during the software development process, such as gathering system requirements, approaches used toward the solution, design phases, high-level explanation of implementation stage, and finally, the testing strategy used.

Chapter 4 will present the results of the implementation along with an evaluation of the whole project.

Chapter 5 will present the conclusion by explaining how well the project has met the original objectives, addressing both positive and negative aspects, and providing lessons learnt with recommendations to further works.
References

2.0 Background

In this section, background research was conducted to assist with the development of the application. This will include a literature review and the current technology used. Additionally, research was conducted to investigate the movie recommendation systems in the entertainment industry. The reason for the app to be developed on Android is that this has mass appeal and can target the most significant number of users as "72.84% is Android," according to Statista June 2021 [6].

Alongside the wave of big data, the demand for recommendation systems is now imperative in numerous fields, including e-commerce, social media networks, and various other applications. Currently, there exist systems such as MovieLens and Netflix [7].

Netflix considers "the types of shows that a customer watches and provides recommendations similar to those" [8]. However, recommendation systems are comprised of three types "collaborative filtering systems, content-based filtering systems, and hybrid systems" [8].

2.1.0 Collaborative filtering systems

Firstly, collaborative filtering systems are based on the users' previous experiences and ratings associated with other users [9]. This data is then used to build models based on the previous watch history. These are then cross-referenced with the ratings of users and other similar users who have recommended the same movies. Hence, this "approach uses the preferences of other people with similar tastes for recommending items to the user" [10]. There are two methods in which this occurs: the memory-based method that utilises user data to make recommendations. Over time utilising this data will gradually improve the accuracy. The second approach, as previously mentioned, is the model method which essentially constructs a model of a user's behaviour and then uses certain constraints to forecast future behaviour. For example, figure 1 below demonstrates the collaborative filtering process as if user A watches Movie one (M1), M2 and M3. Also, if user B watches M1, M3 and M4, we would recommend M1 and M3 for user C.

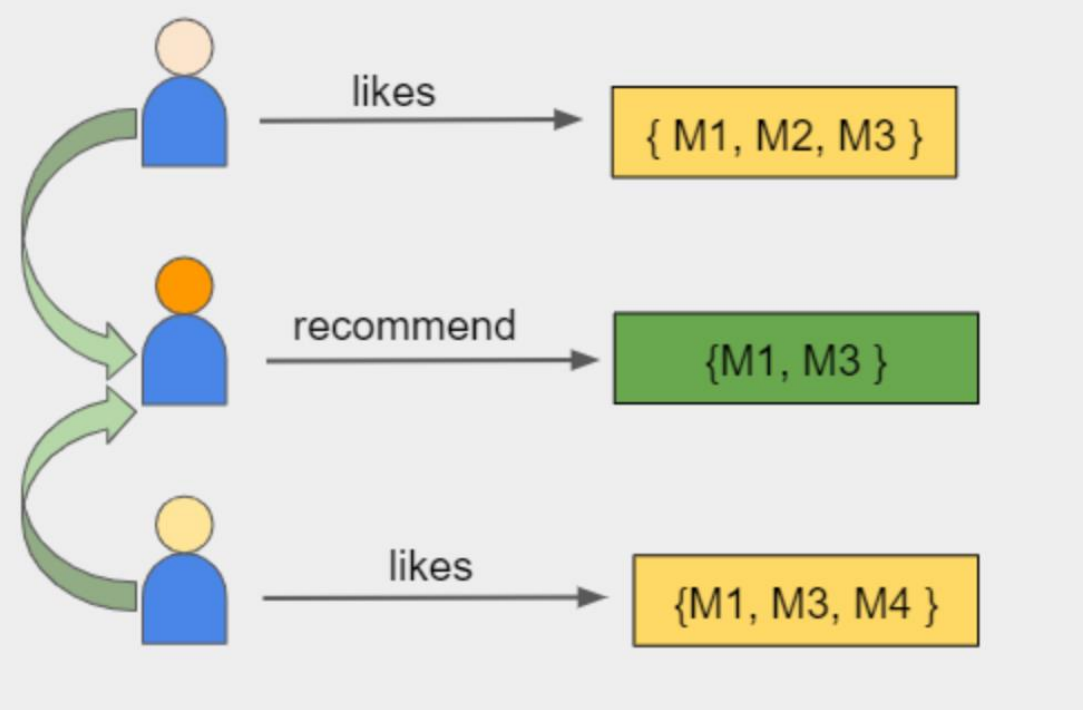


Figure 1- Collaborative Filtering diagram Source: [9]

2.1.2 Content-based filtering systems

Secondly, content-based filtering systems [9] are another type of recommendation system that attempts to analyse preferences and documents given by a user, for instance, specific interests provided during the signup, and this is used to build a model. This is achieved through three avenues. One being Wrapper methods which separate the features into subsets then an analysis is run on these subsets to determine which one is the most likely to fit the user. Then filter methods and heuristic methods rate features on the content they consume. Lastly, embedded methods are used with algorithms for feature selection during the training phase of the algorithm. A disadvantage of this method is that this requires a significant amount of data to build a reliable classifier. Figure 2 below demonstrates this as if user A watches M1 and M2 and they share the same genre comedy this could be the feature then the system will recommend movie K, which also shares the same feature.

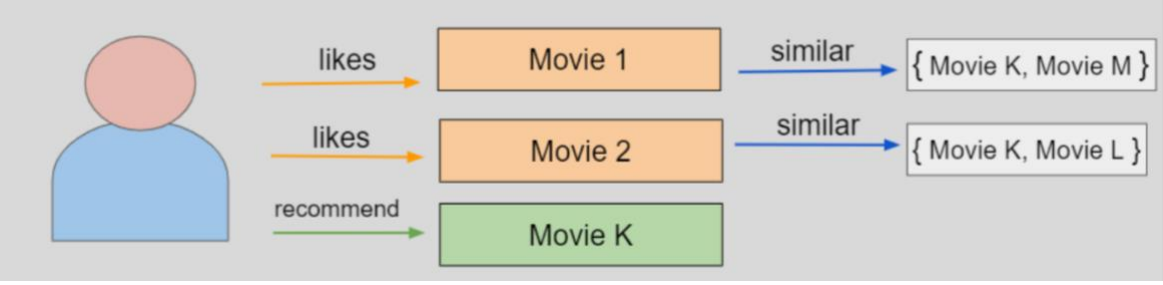


Figure 2- Content-based filtering diagram Source: [9]

2.1.3 Hybrid based filtering systems

Lastly, there are Hybrid systems in place which combine both the collaborative and content-based systems. Similarly, these are categorised into three methods. The three types include weighted hybrid systems, mixed hybrid, and cross-source hybrid. The weighted hybrid systems are carried out by maintaining a score for each object and finding the weighted sum with respect to the diverse context sources. Each weight is different based on the user's preference. In mixed hybrid systems, each source is used to rank the items, and the top few items from each ranked list are picked. The cross-sourced hybrid method works by recommending items that appear in multiple context sources, so principally is based on the more sources an item appears on, the more crucial the item.

2.1.4 Current systems

Current systems also consider the user's emotion as this "plays an unexpected critical role in the decision process" [11]. The relevance to this in movies is that viewers watch and experience movies that elicit an emotional response. As movies that fall under the comedy genre can create a feeling of joy, conversely, sadness can be felt during the loss of a character in a movie. Also, the state that the user is currently in will influence the movie genre they would like to watch. For example, if the user is in a sad mood, they may want to watch a sad movie. Equally, if they are in a sad mood, they may want to pick up their mood by watching something that is more fast paced to divert their feeling of sadness. Netflix also incorporates the time of day and the day of the week and location [12]. They even personalise the artwork and thumbnail of each movie to appeal to specific audiences. For example, they may use a particular scene from a movie to demonstrate the exciting nature of the movie as particular images may appeal to certain subscribers more than others, hence increasing the likelihood of clicking on the movie [12]. These are all factors that affect the behaviour of the viewer.

To predict the user's emotion, this is usually done through two ways [11], either questionnaire, which is a simple and inexpensive way to capture emotions, but the accuracy is not as accurate as the user may not tell the truth or find it difficult to express their emotions [11]. Secondly, we can use different colours to predict the user's state as researchers argue that colour intrinsically represents human emotions [11].

For example, yellow infer positive emotions such as joy [11]. On the other hand, colours such as black and grey are often associated with negative emotions such as sadness and loneliness. Therefore, these colours are often implemented within an algorithm that will determine the user's current emotional state when selecting colours. The only issue with this often requires the user to select up to three colours, and hence if two colours indicate one emotion and one colour indicates another, then the average is selected.

2.2 Existing systems

One of the existing movie recommendation systems I have come across is Taste [13]. Below are the screenshots of how this operates. Initially, users create an account, enter a name, and enter their gender, location, and age (figure 3 and 4). Then users are encouraged to rate different movies, which create a profile of suggestions (figure 5). The result is a percentage match of how accurate the suggestions are given the inputs (figure 6). The more movies you rate, the greater the accuracy. One of the good features when using this app is that it functions on the web and both on Android and iOS. The author believes this means a large audience can use this and the UX design on it was intuitive and straightforward to use. The recommendation system that will be created will be drawing some inspiration from this aspect of the simple design. Although Taste uses machine learning to provide recommendations, the app created will not be implementing this feature. However, this can always be implemented in a future iteration. Another good feature of this app is that it has a community basis as you can follow other users to view their recommendations or reviews. This also allows other users to view their comments on different movies to get unique perspectives or alternative tastes that would not be recommended to them. Lastly, Taste also implements various filters other than just genre as it also incorporates filters such as 'By attribute' as seen in figure 7. This allows the user to select a more nuanced selection of movies, even including the option to 'explore community' as seen in figure 8, an additional filter that will allow the user to review and follow, anticipating movies that are coming soon.

The only drawback of Taste is that there is no accurate rating system. Instead, the rating system is based on the community who have watched and reviewed the movies. This is a drawback as some community members have not yet reviewed or watched the movie. Hence there is no overall rating for particular movies.

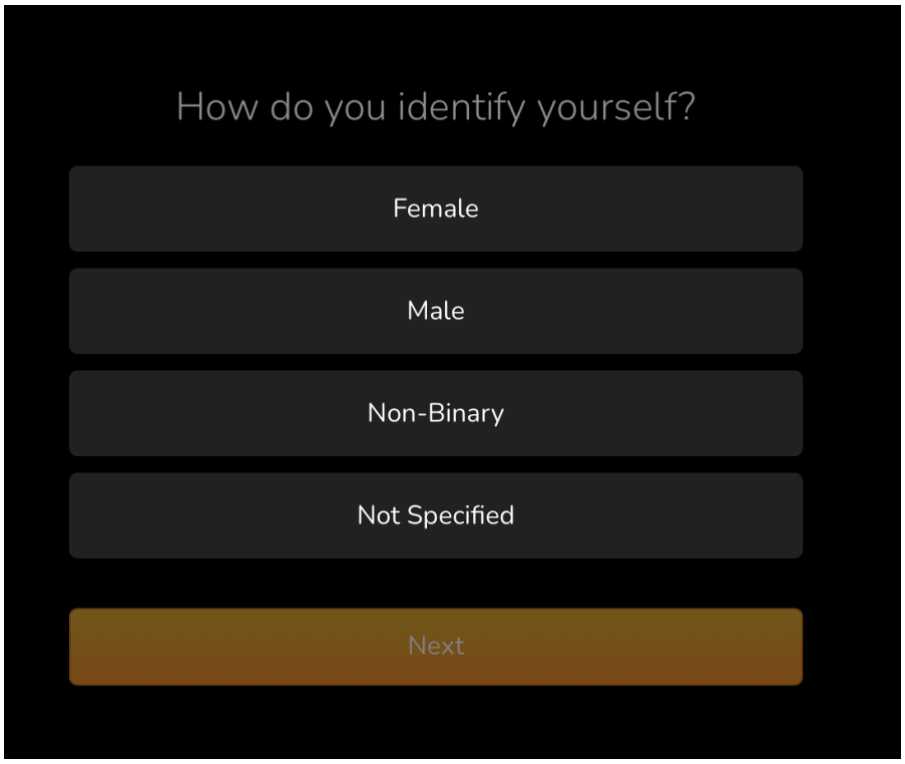


Figure 3- Taste App registration [13]

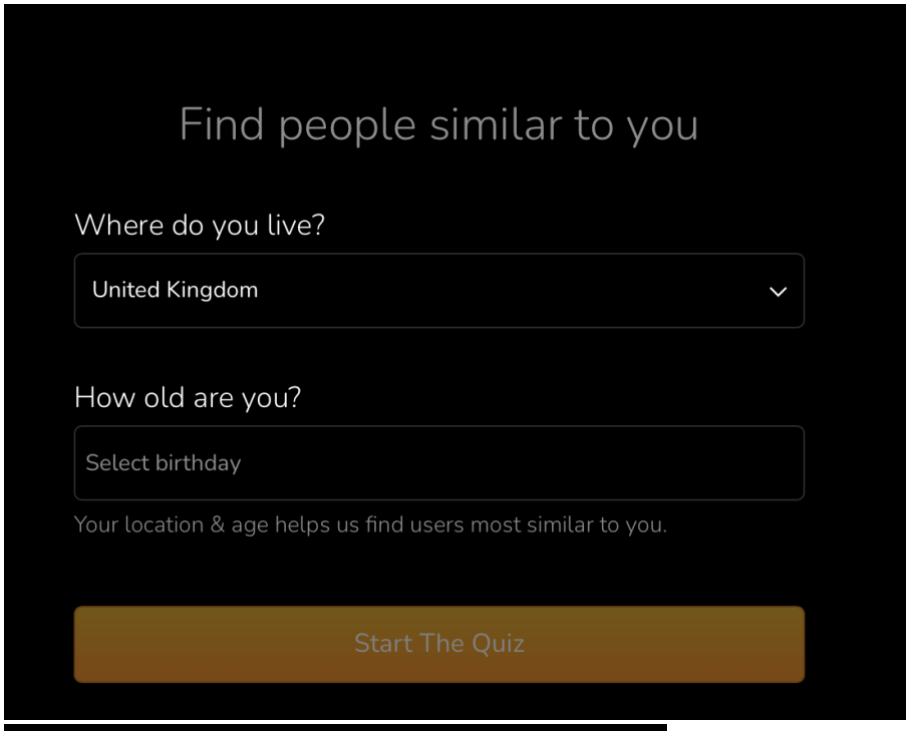


Figure 4- Taste app registration x2 [13]

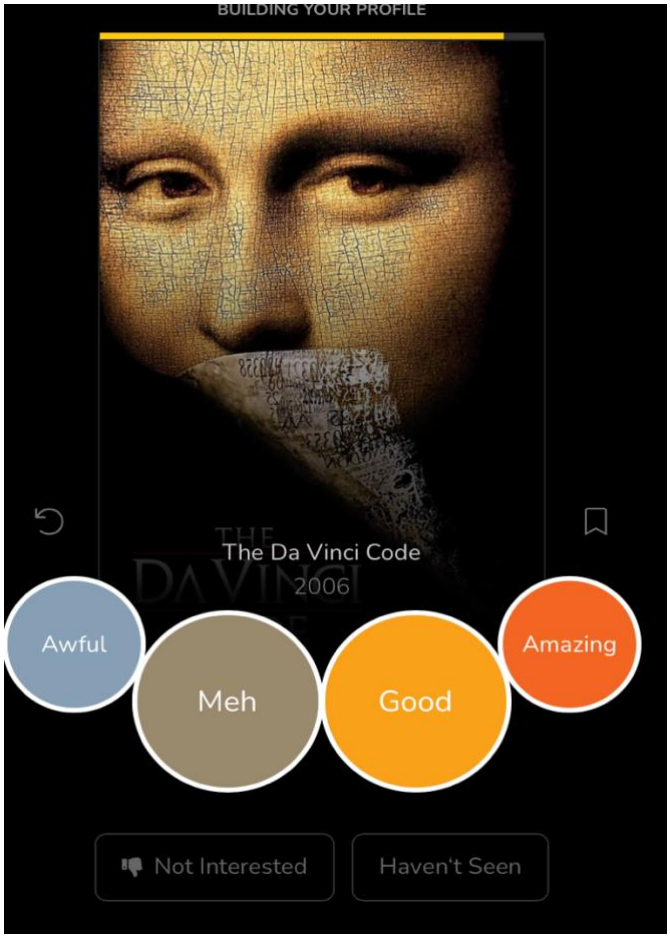


Figure 5- Taste app movie rating [13]

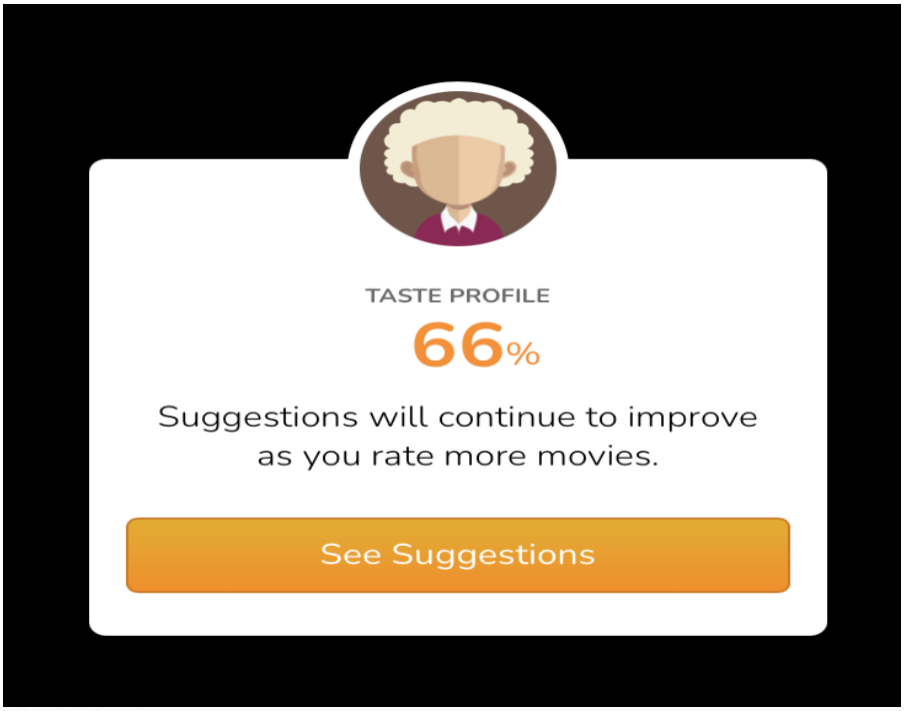


Figure 6- Taste app profile accuracy [13]

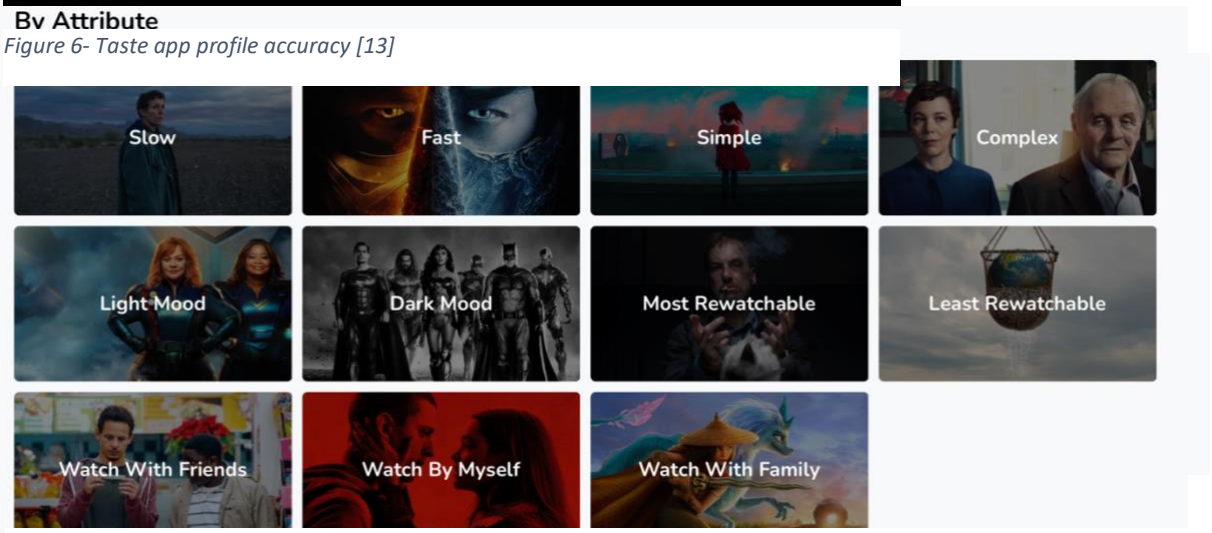


Figure 7- Taste app By attribute [13]

Much like Taste, there is another recommendation system called MovieLens. However, this is only web-based [14]. It is run by a research group at the University of Minnesota and uses collaborative filtering (CF). As mentioned, prior CF uses predictions based on other users' preferences [9]. Similarly, Taste MovieLens encourages users to register an account. Once registered, users are directed to a welcome page (see figure 9). The welcome page allows users to distribute points to various groups of movies.

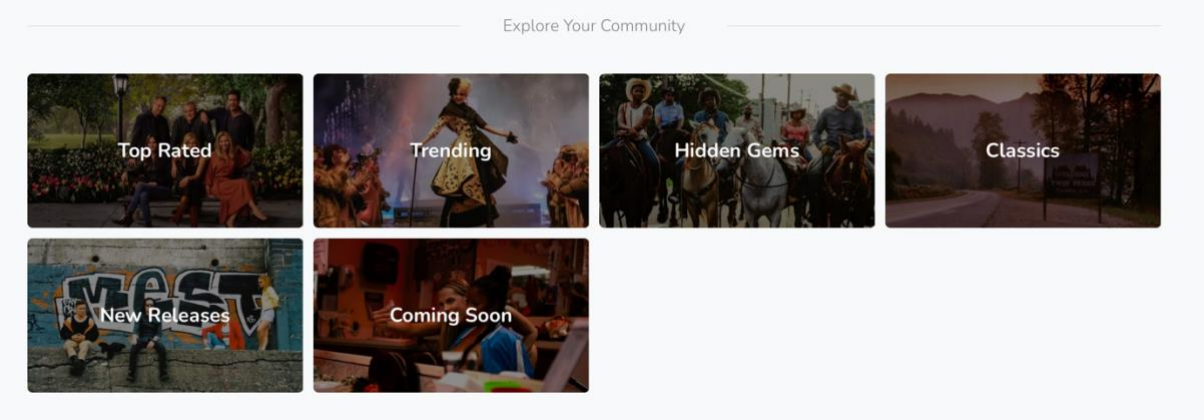


Figure 8- Taste app Exploring Community [13]

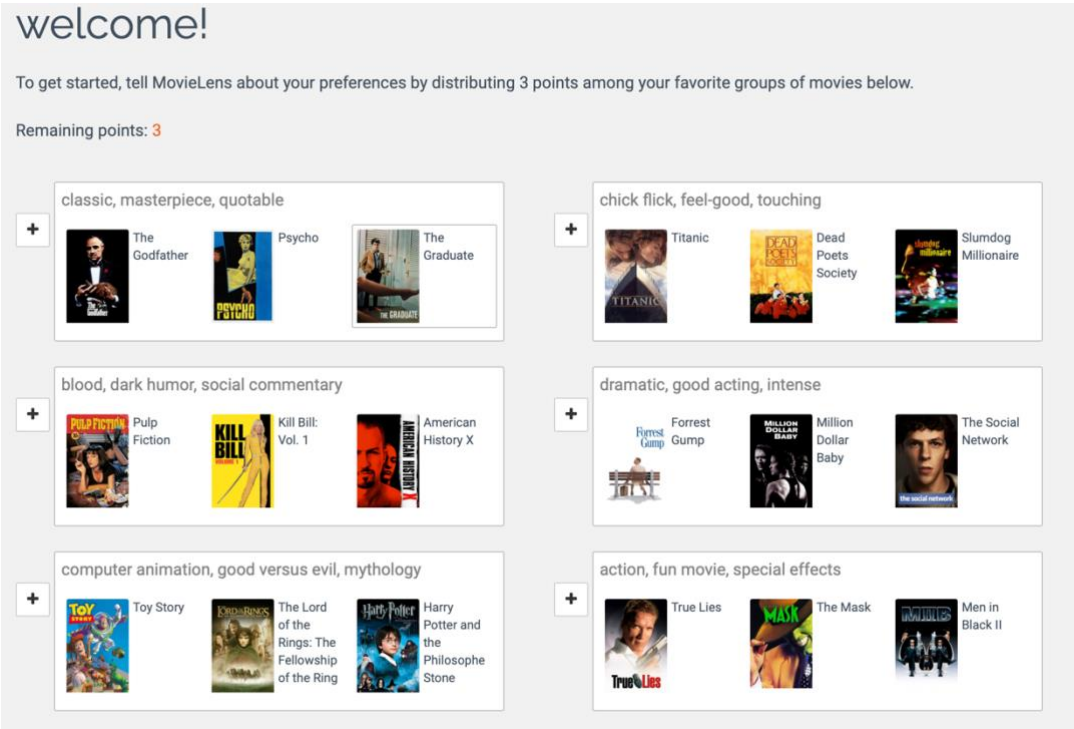


Figure 9- MovieLens Welcome Page [14]

Once the user has distributed the points, a tutorial page (see figure 10) illustrates that the user is encouraged to add a star rating to a particular set of movies. This is good for UX as it describes clear, intuitive instructions on how to use the website. Also, if you click on the movie, additional information is given see (figure 11). This feature includes a trailer and a short description of the movie. In the authors opinion this feature is handy as it gives a bit more detail on the movie to the user; hopefully, the author will be able to develop a similar feature with his project. In addition to this, as this application allows you to rate movies, a page saves all the users' individual ratings (figure 12).

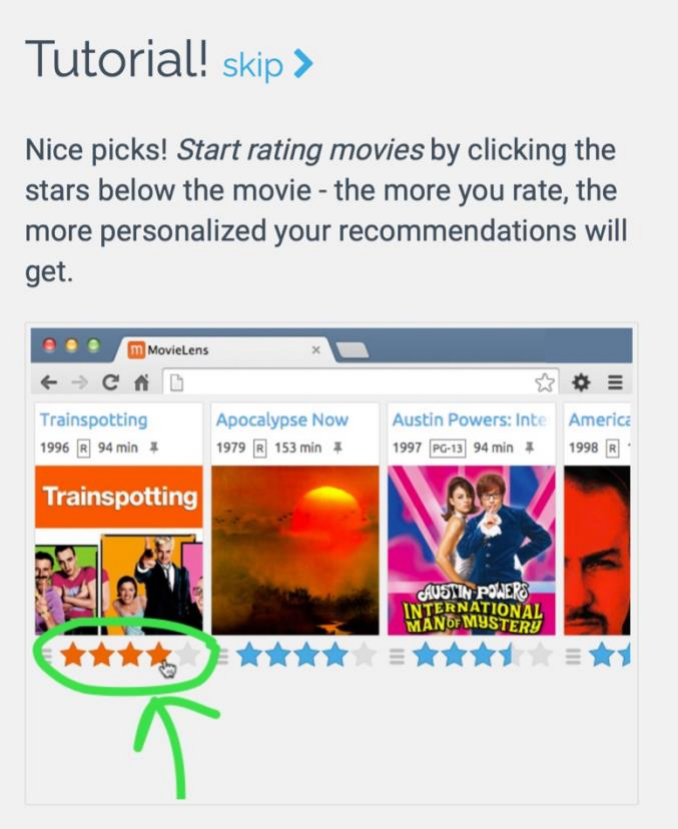


Figure 11- MovieLens Tutorial [14]

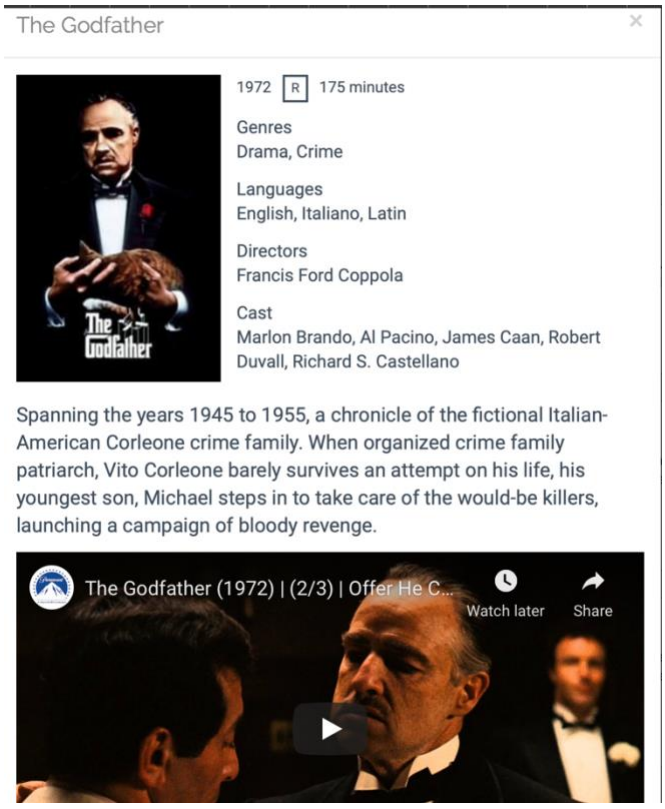


Figure 10- MovieLens Description [14]

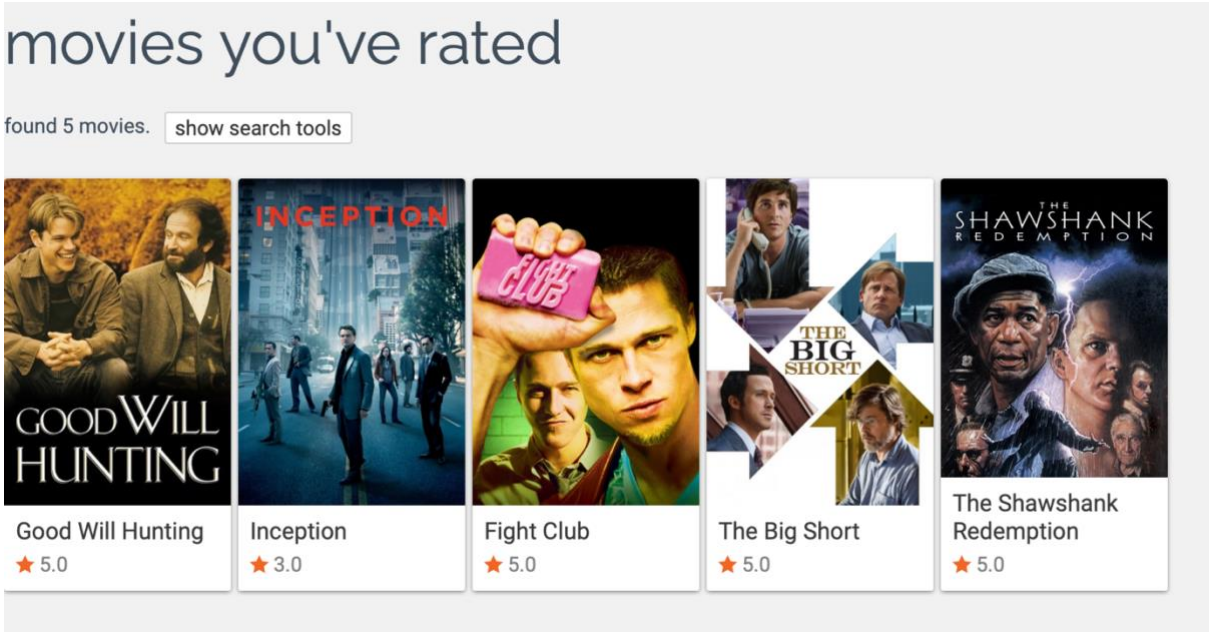


Figure 12- MovieLens Rating system [14]

An exciting feature about MovieLens is that you can choose which recommendation system you would like to use as the one selected initially is most optimised for new users but the 'warrior' recommendation system [14].

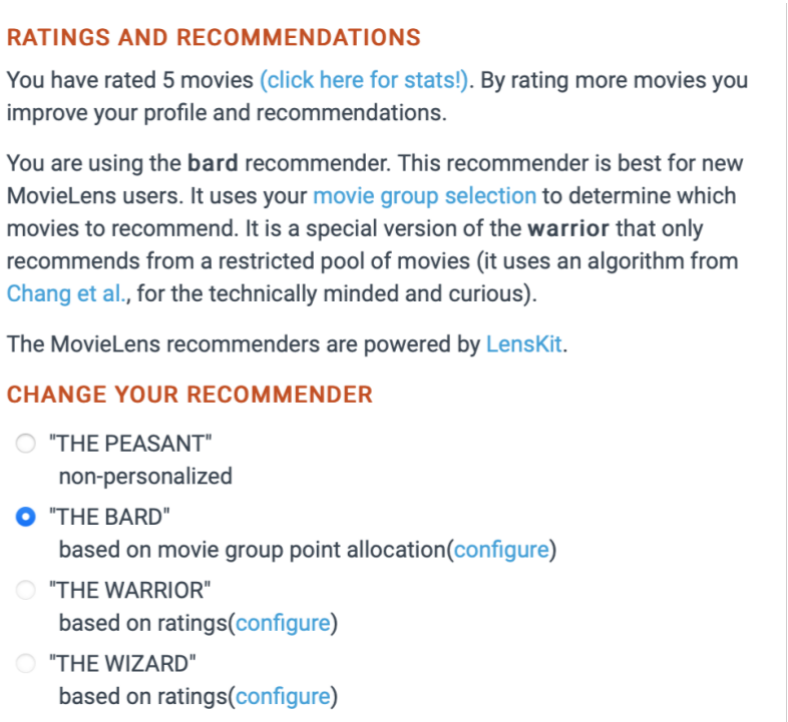


Figure 13- MovieLens Recommendation system choice [14]

One thing that stood out with using MovieLens was that there are statistics about the reviews you do with the movies you see (figure 14). The last screenshot demonstrates the outcome

from the points assigned to the movie groups and the user rated movies. These were displayed with different categories, including the popularity of the rated movies, how highly you rated movies, what date you rated the movie and the release date of the years. Also, including the number of movies rated per genre and average rating per genre. These statistics are insightful given it tells the user more information into what movie behaviour they tend to exhibit and prefer. Although we will not include this in our application, it is a possible extension we may include in further iterations.

Figure 14- MovieLens Statistics [14]

The results below display see figure 15, the movies that MovieLens would recommend based on the points given at the start and the individual reviews we proceeded with. The display is interesting as it shows the movies panning from left to right. This is possibly because it is on

a webpage as our application will be mobile based. Therefore, it is more likely to hold a list view instead of this panel view coming across the page.

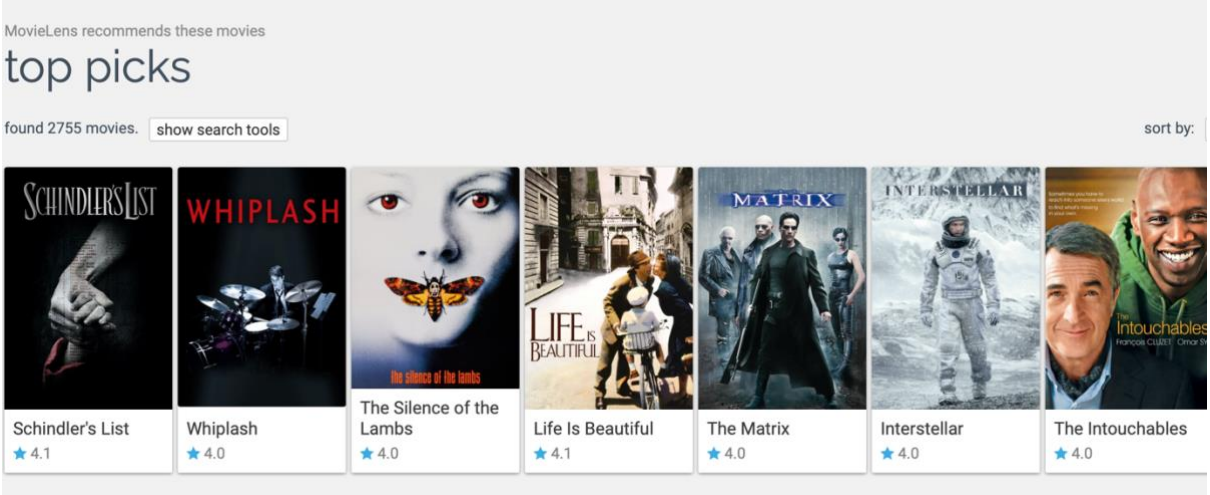


Figure 15- MovieLens Results of recommendation [14]

One of the drawbacks of MovieLens is the lack of devices you can use their application on as it is only web-based. Thus, the device compatibility is limited as phones cannot be used. Furthermore, this can create a smaller demographic as younger audiences spend more time on their phones than on any other device[6]. Another drawback that was noticed was that the ratings again were based on users that used the platform MovieLens and although there are several users, it still is not close to the number of reviews posted on IMDB. Hence, making the ratings somewhat less accurate.

2.2 Preliminary Survey

As this app will be catering to individuals who want to reduce the amount of time spent searching for a movie or tv show, it would be appropriate to gauge whether individuals would use this product. Hence, a preliminary survey was conducted through Microsoft Forms before creating the app to gather opinions and suggestions on what features should be implemented and if users had come across similar systems. The form was distributed to a range of ages and genders between the target demographic of 18–34-year-olds.

There were some qualitative metrics recorded as each participant will be given a pre scenario quiz (refer to Appendix C). These qualitative questions were used to extract additional information that would supplement the results found in the questionnaire. The pre-scenario quiz will consist of some open-ended questions in which quotes can be taken from the user. These are intended to complement the information gathered from the survey.

In total, there were 11 questions and seven responses. Due to the circumstances of covid, the sample size is somewhat smaller, but the survey still gives you an idea of the similarities and differing opinions on what individuals would like to see.

2.2.1 Most Important aspects to user

The first question for the respondents was “Rank in terms of importance the following aspects of our product to you?

Ease of use, Accessibility, Navigation, Design, Device compatibility (1 being most important 5 being least)". All respondents answered the question, and 51.7% agreed that Ease of use was the most crucial aspect for the app second was Design, with 28.6% consensus that it is the most important aspect. The results for this demonstrate that Ease of use and design comes in as the main priority for the application. Interestingly, accessibility was deemed the lowest priority. However, the author would still consider this important given that the app has to be inclusive of all people. This could also be used to extend the features that are not implemented in the first iteration. Moreover, it could be added in further iterations. Also, it is important to note that device compatibility was low on the priorities; the author assumes this is because the target audience was aware this would be coded in Java and would be only available for Android users. Perhaps if the target audience were not aware of this, it would be higher on the priority list. Also, the respondents were answering this in terms of whether they would be able to use it both on their laptops and smartphones.

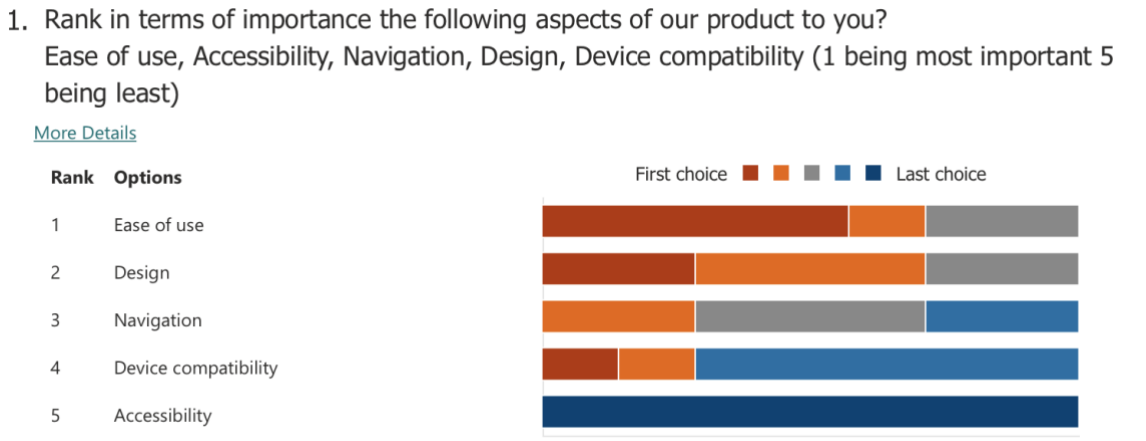


Figure 16- Preliminary survey. Rank in terms of importance

2.2.2 How often would you use this

The second question was, "When you think about this product, is this something you would use and how often?". This question was designed to understand whether there was any opposition and whether this would be used frequently. Again, the respondents all answered yes with varying degrees of usage. From this, we can infer that there is a demand for this kind of service.

2. When you think about this product, is this something you'd use and how often?

7 Responses

ID ↑	Name	Responses
1	anonymous	Yes, whenever I decide to watch a movie
2	anonymous	Yes, once a week
3	anonymous	Yes, twice a week
4	anonymous	Yes, maybe on the weekends
5	anonymous	Yes, every time I need to think of a movie
6	anonymous	Yes
7	anonymous	Yes, whenever I have free time

Figure 17- Preliminary survey. Would you use this product?

The third question was, "How likely is it that you will recommend this kind of product to a friend or colleague?
1- Not likely 9- Very likely". This question was designed to understand whether they also think their friend or colleague would find it helpful to create a more expansive product usage. The respondents all answered yes with varying degrees of usage. We can infer that with an average of 8.14, almost all the respondents are very likely to refer this product to their friends or colleague. This underpins the notion that this would have great demand in the market.

Figure 18- Preliminary survey. Likelihood of referral?

2.2.4 What type of device do you use

The fourth question was, "Are you an Android phone user examples of which include: Samsung, Google, OnePlus, Motorola, Huawei or any other brand that runs on Android OS or an Apple iOS user?". This question was imperative as this is what device the application should run on and what programming language needs to be written. The respondents predominantly used Android, which is interesting given that "72.84% is Android," according to Statista [6]. Hence, this supports the studies and reaffirms there is greater demand for the Android market than Apple's iOS.

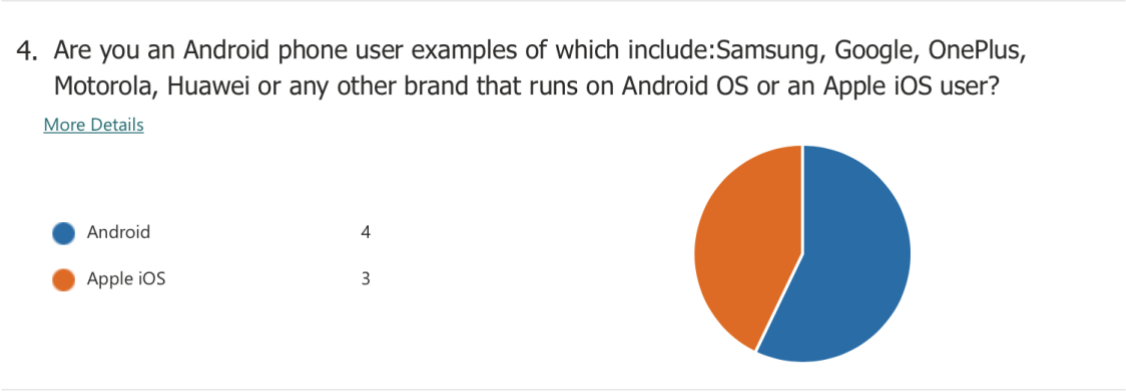


Figure 19- Preliminary survey. Android or iPhone?

2.2.5 Hours spent watching

The fifth question was, "How many hours do you spend watching a movie or tv show in a week?". This question was posed to the respondents to gauge whether they are avid movie consumers and just trying to figure out whether they spend much time watching entertainment. As previously mentioned at the start, an average of 45 hours a week is spent on entertainment [1]. However, our survey results interestingly only illustrated 3 members spending 4-5 hours a week. This could be due to underreporting from fear of judgement or just unaware of the actual time spent; this could also be due to the umbrella term entertainment could include social media and phone use amongst other entertainment avenues.

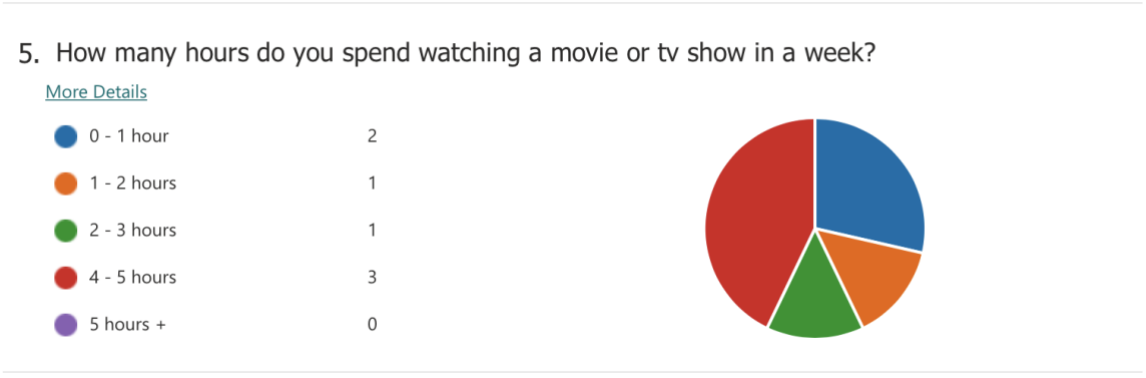


Figure 20- Preliminary survey. How many hours spent on movies?

2.2.6 Movie Streaming websites

The sixth question was, "Do you use movie streaming websites, and if so, which ones?". This question was asked to understand what movie streaming websites were most popular to the respondents. Again, Netflix was the most common, with 4/7 respondents answering this. Others included Amazon Prime, NowTV and Apple TV. According to the World Economic Forum, Netflix has 204 million users; Amazon Prime has 150 million. This explains the prominence in Netflix and Amazon, although the others were not mentioned in this study.

6. Do you use movie streaming websites and if so which ones?

7 Responses

ID ↑	Name	Responses
1	anonymous	Netflix
2	anonymous	Amazon Prime
3	anonymous	AppleTV
4	anonymous	NowTV
5	anonymous	Netflix
6	anonymous	Netflix
7	anonymous	Netflix

Figure 21- Preliminary Survey. Which movie streaming websites do you use?

2.2.7 Accuracy of Recommendations on Streaming websites

The seventh question was, "How accurate are the recommendations from these movie streaming websites?". This was asked to see whether respondents were content with the recommendation services provided by their current streaming providers. The results were mixed, given that a third of respondents were satisfied with their recommendations. On the other hand, a third of respondents were not satisfied and 43% were only somewhat satisfied. This highlights the need for an alternative application which deals with the recommendations separate or could work in tandem with the streaming service. As these results validate the need for the application, despite the existing streaming services making their in-built

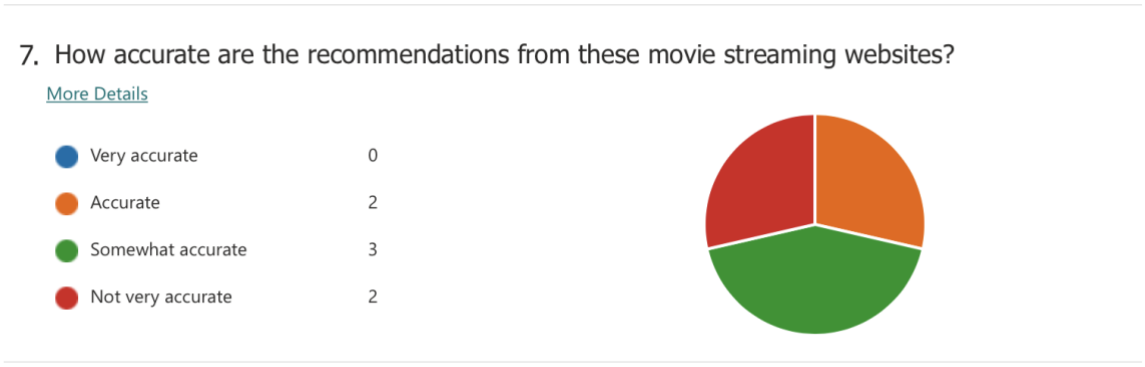


Figure 22- Preliminary survey. How accurate are the recommendations from these movie streaming websites?

2.2.8 IMDB Rating

The eighth question was, “Do you check IMDB movie ratings before watching them?”. Again, most respondents said yes, with 71% saying yes and 29% saying no. This is significant given that the rating is what the movie recommendation system will stem from IMDB ratings of 7 or above. From this, we can infer that IMDB is important to users before watching the movie.

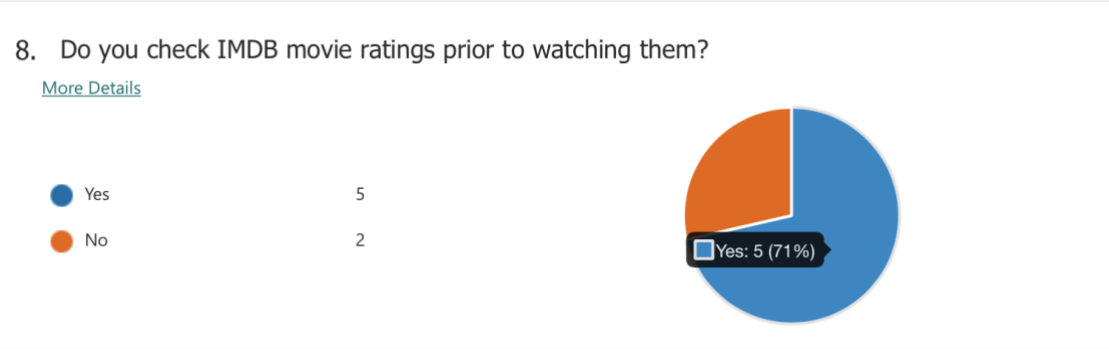


Figure 23- Preliminary Survey. Checking IMDB

2.2.9 Key Features

The ninth question was, “List 3 features you would like to see in a movie recommendation app?”. This was one of the key questions given that it asked the user directly which features they would like to see in the app. Although not all the features mentioned by the users will be implemented, their opinions are important as it gives direction to what consumers want to see.

One of the key features that were mentioned across the board was registration. Allowing the user to register an account creates a more tailored experience, and the user will feel unique as the details can be saved and updated. Instead of constantly having to refresh and get different results each time, this would also allow the user to enter personal details such as age which may influence their choice in movies, so perhaps this could be used to compare what types of movies particular age groups are watching.

Another feature that may not be implemented in the product but was a good idea was a community page where users can connect with other users of the application to post comments on the movies and communicate with a community of users. This could be used in tandem with social media so users could post their recommendations and share them on the likes of Facebook.
Lastly, a good user interface that is simple and intuitive was reiterated by respondents. As this is important to users, the application will ensure it meets these requirements.

2.2.10 Time taken to choose a movie

The penultimate question was, “Typically, how long would you say it takes you to find a movie you would like to watch?”. This was asked to see how much time was wasted on

choosing a movie. The results show that 43% of respondents spend 20 minutes or more choosing a movie and 29% spend 0 – 5 minutes and 14% for the other two, respectively. This again reaffirms the need for this application as it will cut the time spent discussing or pondering what movie to watch and hopefully eliminate that problem.

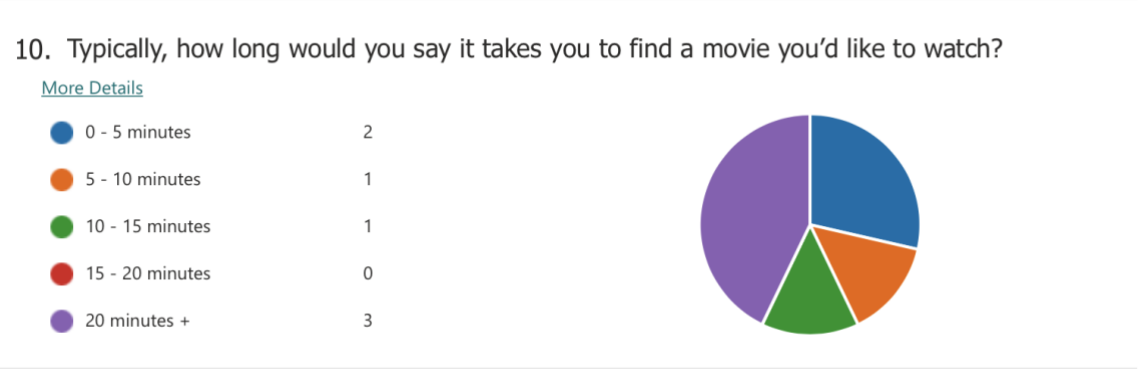


Figure 24- Preliminary survey. Time it takes to pick a movie

2.2.11 Any existing movie recommendation apps aware of

The last question was, "Are there any movie recommendation apps you have come across yet? If so, which ones?". This was asked to gauge whether any likely competitors or any apps users were currently using to draw inspiration from. There was only one that was mentioned, "Popflake". Upon researching this application, it had a nice interface and was quite simple to use. However, there was no registration for the user. From this, we can infer that there are very few applications that at least these respondents were using and aware

of, which means there is still space in the market for this kind of application as there is a demand for it.

3.0 Building the system

The function of the application is to provide movie recommendations based on the user's genre and year released. In addition, the application will consider recommendations from the

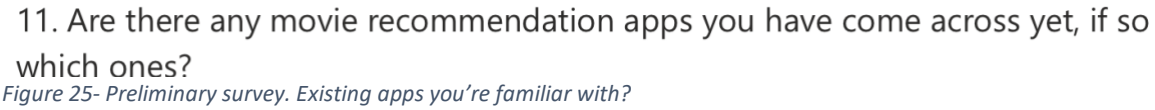


Figure 25- Preliminary survey. Existing apps you're familiar with?

ID ↑	Name	Responses
1	anonymous	No
2	anonymous	Yes Popflake
3	anonymous	No
4	anonymous	No only movie review apps
5	anonymous	No not yet
6	anonymous	No
7	anonymous	No

preliminary survey. To fulfil these requirements, adequate planning and organisational needs to ensure these objectives are met.

One of the crucial components to project development is establishing which software methodology will be implemented for the project. After consideration of the different processes, the final verdict was to choose the Agile Model. This made sense as it allows for greater flexibility and allows several iterations and is conducive to a change in requirements if needs be [17]. Particularly given at the outset where the project is uncertain. In principle, the project details should be all be finalised before starting the project. However, this is not always possible. Moreover, the Agile model is the best suited for this project as a new software program had to be learnt (Android Studios). Hence the learning curve was steeper, and as this is only a small project, this process is more appropriate.

Another reason for this choice is that Agile's vital principle is to satisfy the customers need through early and continuous delivery of valuable software [17].

This process allows the software deployment to be much more streamlined as one way of working in Agile is the Scrum framework, which comprises assessing, selecting, developing, and reviewing. This essentially is done through sprints lasting a fixed time between 2-4 weeks. Given the time constraint in the project, this had to be altered to every 1-2 weeks.

The advantages of this framework include the following: the project is allowed to be broken into smaller, more manageable chunks, there are more iterations of the project, so this allows greater fine-tuning and hence is easier to implement changing requirements from the customer. Another advantage of this is that in a team setting, the communication is often improved as the whole team is aware of everything going on, and there is immediate feedback that can be used to improve the project for the next iteration.

Figure 26- Agile diagram [18]

The only issue with the Agile method is that much time was spent on adding other features to satisfy the customer's needs, which led to some features such as community integration not being implemented because of the time constraint.

The author has not chosen the Waterfall method because this method is outdated, as this linear way of methodology does not reflect the distinct nature of this project given the added complexity of learning the new software environment.

Also, this method creates a longer delay in the actual code being produced and any errors will only be fixed once everything is completed as there are no continuous iterations, unlike in Agile. Therefore, using this method will cause further delay in updating the efficacy of accuracy of recommendations and increase time spent caused by errors. Another reason why Waterfall may not be appropriate is that the given system is not so detailed, at least at the outset, in Waterfall the given requirement usually must be very detailed as there has to be a clear separation of stages, unlike Agile, which is more experimental in its approach. Hence, it is easier to make any changes along the way. Thus, reinforcing the need for Agile as there may be a need to constantly change things as the problem statement is quite vague.

Lastly, although the Evolutionary Model could be considered for this project, the system is not very detailed. Therefore, it is more accommodating to the user as it may have fewer features, the issue however with this method is that it often exhibits code and fix problems as the system design may be poor as the systems may be developed quickly, but their often structured poorly and is often not very cost-effective [17]. Thus, it reduces the system's scalability, and this perhaps could be expanded in the future, but this will not be possible if it is poorly structured.

3.1 Functional Requirements

A functional requirement is defined by "The basic system behaviour. Essentially, they are what the system does or must not do"[19]. The application will consider the feedback given from the preliminary survey.

Table 1- Functional Requirements

Requirement	**Priority**
The user will create their login by using their full name, email address, password. Also, age will need to be implemented, which can be used later to filter movies within that age range.	High
The application will use an API key to access movies from the TMDB database as the IMDB database does not allow filtering by genre.	High
The application will include different activity pages, including a registration page, a login page, a movie dashboard, a movie description page.	High
The user will access interactive elements such as external links to a movie trailer for the recommendation through the movie description page.	High

The user will be able to view the rating of each movie on the dashboard.	High
The user will be able to go back to search a different genre or year of the movie.	High
The app must also operate on a landscape view	Medium
The app will have a top bar explaining what is on the page.	High
Notification of registration when user successfully registers	High

3.1.2 Non-Functional requirements

A non-functional requirement is defined by "specify how the system should do it"[19]. These are often "product properties and focus on user expectations" as opposed to user requirements.

Table 2- Non-Functional Requirements

Requirement	**Priority**
When a user creates a login on the registration page, the username, password and age will be passed into the Firebase database with a unique ID associated with it to differentiate each user.	High
The application will store the user's session to keep the user experience smoother. This will be done by using the Datasnapshot feature in android studios. Which essentially operates as copies of the data and stores any data on the emulator to pass into the other activity pages and reads through the firebase database.	High
The application will keep a consistent colour scheme using only two main colours, black and blue. This is so it remains easy to read and has a clean interface. The font scheme will be reviewed using Jakob Nielson's Ten Usability Heuristics to ensure good practice is exhibited in the application.	High
The application will be running on a Google Nexus 5X emulator, which can be done through Android studios by downloading the required SDK tools for your device and creating an Android Virtual Device (AVD). Then in the toolbar, select the target device you want to run on and then click run program. To navigate around the emulator, you can use a keyboard and mouse.	High
There will be a user manual. This will be accessed through appendix section A and will be in pdf format.	High

If any users need to be deleted, this can be done through the Google Firebase database by clicking on the Realtime database and then clicking the cross button, which deletes the user. Alternatively, deleting the user attribute will delete all users if all users need to be deleted. Equally, if any user data needs to be edited, the same process applies but click on the individual attribute of age username or password.	High
The application will be maintained and updated to add further functionality in the future.	High
Navigation will be simple so that a "Get movie" request for a movie will be placed on the application in less than 5 minutes.	High

3.1.3 The Technologies

As the application will be mobile based, there are two options in terms of what technologies to use. The first being Swift which would be appropriate for iOS operating systems, and the second was Java which could be used for Android systems. Then, whether it should be cross-platform (web and app-based) needed to be decided as if it were to be cross-platform, JavaScript would have to be implemented [20]. Initially, Kotlin was also considered, but this was not incorporated as Java was the language the author had the most experience with. Also, the author had experience with APIs in Java from a previous project, so this seemed to be the logical language to use. Finally, the author was recommended to use Google firebase as a virtual database. Despite not having experience in this, it was proven to be relatively straightforward after reviewing the documentation.

3.2 Design

3.2.1 Entity-Relationship (ER) Diagram

This ER diagram represents what is stored in the Google virtual Firebase Database. The rectangles represent the entities, and the ovals represent the attributes which is the only information stored in the database is the user's information [21]. Hence, this ER diagram is rather simplistic. Perhaps the author will consider saving each of the user's interactions with the application in the future. As it currently stands, it only saves the existing entered selection of the app during when the application is running, rather than saving the choices directly onto the database.

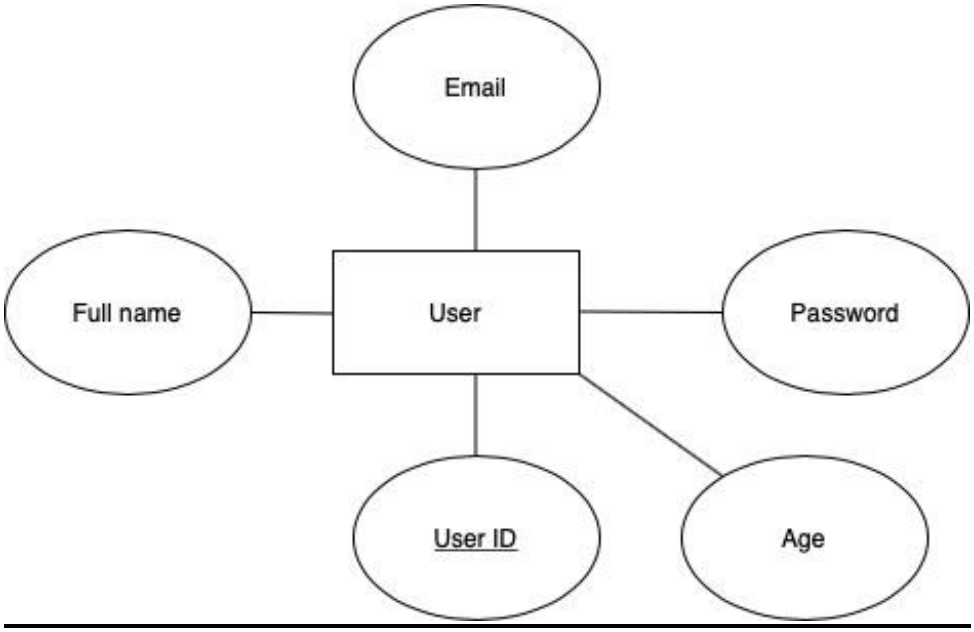


Figure 27- ER Diagram

3.2.3 UML Activity Diagram

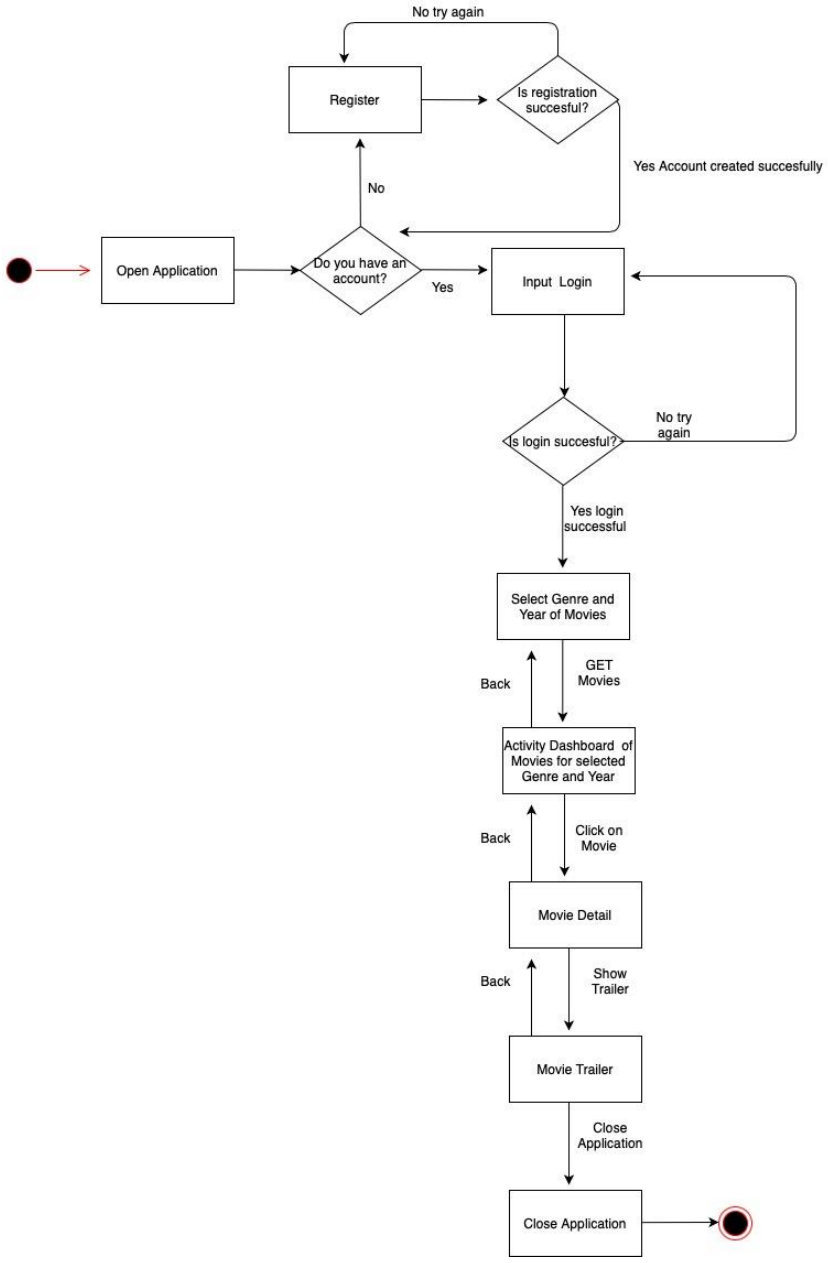


Figure 28- UML Activity Diagram for application

An activity diagram is a diagram that demonstrates the system behaviour, in this case of the movie recommendation app.

A rectangle represents each activity. The connected arrows represent transitions between each activity.

The diamonds represent the decisions; these decisions have two different activity outcomes depending on the decision. The black dot represents the start of the activity, and the black dot with the red circle around it represents the end.

3.2.4 Use case UML diagram

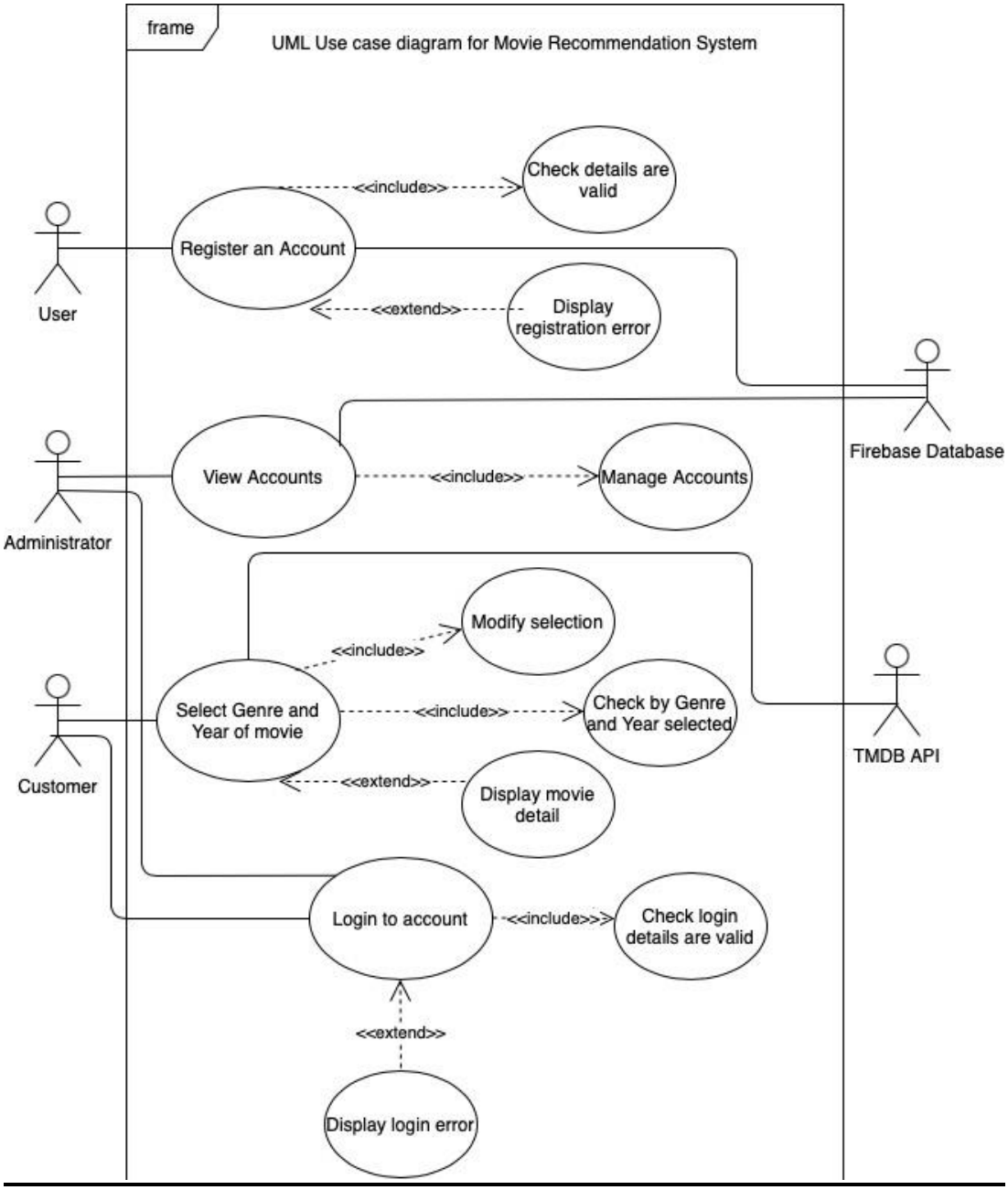


Figure 29- UML Use case diagram

From this, we can see the crucial interactions between the system and other Actors. The actors are represented as stickmen, each use case by an oval and the system as the frame containing all use cases.

The solid lines illustrate the interactions between actors and the use cases. For instance, both a user and the Firebase Database partake in the registering of an account. The user inputs the data and creates the account, then the database stores this information if it fulfils the requirements. The dashed arrows demonstrate the relationship between the use cases. The '<<include>>' "is a relationship in which one use case includes the functionality of another use case". The '<<extend>>' relationship is used "to specify that one use case extends the behaviour of another use case". For example, the login details are always checked when

logging into an account, hence <<include>. However, a login error will only be displayed if the login details are incorrect <<extend>>.

3.2.5 User Interface

This discussion on the user interface is three-fold. Part one will discuss what the user interface should look like, part two will identify what principles this was based on, and the latter will show screenshots and evaluate if the design principles were achieved.

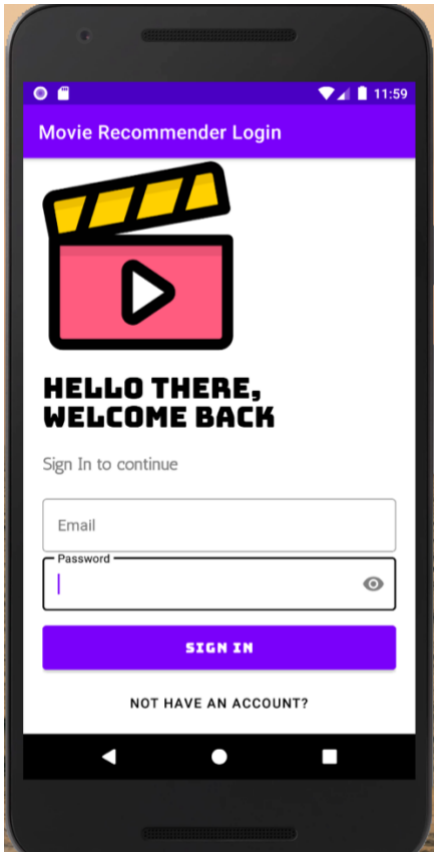
1. What will the user interface look like?
   The user interface shall appear clear and visually appealing. To ensure this is achieved, the background colour used for the login and other pages should contrast with the top bar and navigation buttons and the font colour and ensure the font is easy to read. To summarise, the interface shall give the user feelings of simplicity, intuition, and welcoming on their first interaction.

2. What principles will it be based on?
   The principles the user interface will be based on is two-fold. Firstly, it will follow the four critical rules.

   - “Place users in control of the interface
   - Make it comfortable to interact with a product
   - Reduce cognitive load
   - Make user interfaces consistent” [22]

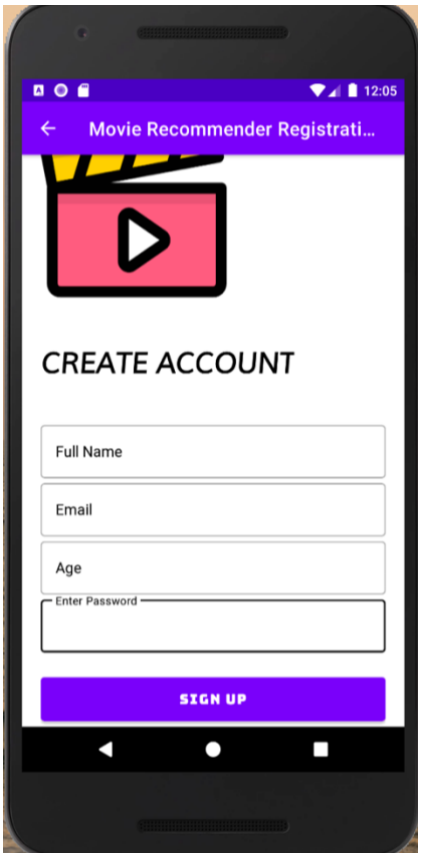
3. Visual representation of user interface and an evaluation of this with Jakob Nielson's Ten Usability Heuristics and addressing any violations based on the scale (0=no problem to 4= catastrophic problem).

Login Page



Google Nexus 5X emulator- Login Page

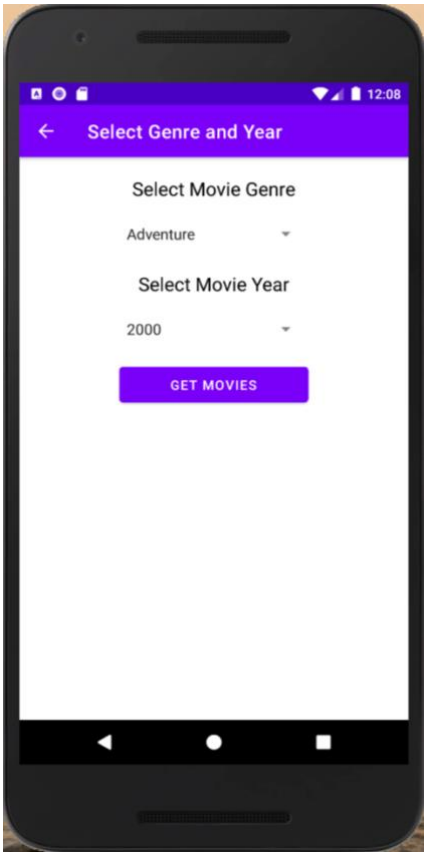
Registration Page



Google Nexus 5X emulator- Registration Page

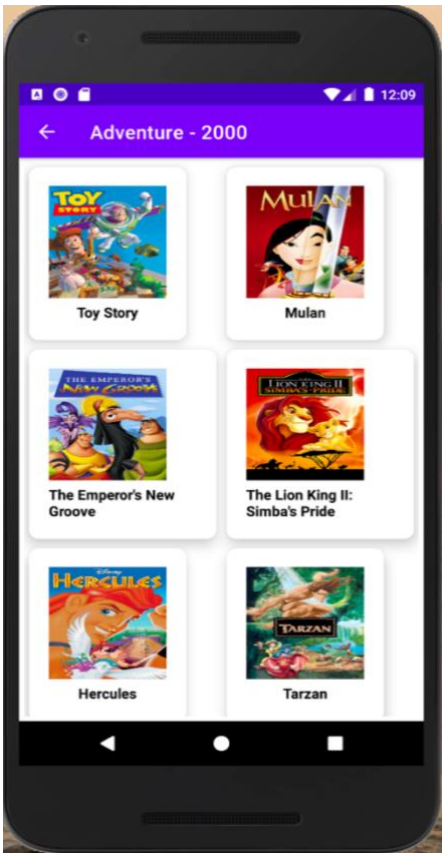
From the above images, we can see that we have incorporated the same font style and background colour consistently across both pages. Also, we have ensured that the password text box cannot show what is being typed. This is to ensure privacy remains. Additionally, we have incorporated images to be more aesthetically pleasing and inviting to the user. This was sourced from [23].

Activity Dashboard



Google Nexus 5X emulator- Dashboard

Movies View holder



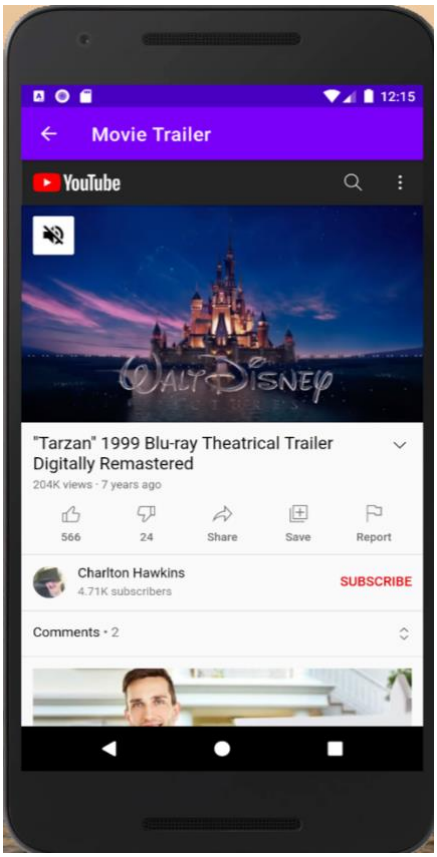
Google Nexus 5X emulator- View holder

The images above represent the Activity Dashboard; once a user successfully logs into the system, this is displayed. The user is then given a choice of genre and year to filter movies from; this is done through a drop-down menu consisting of nine different genres and the years 2000-2020. Once the user selects the year and the genre, they are redirected to the Movies View holder. The Movies View holder will display the movies under the year and genre selected on the top bar. This also illustrates the chosen genre and year. If the user decides to change their choices, they can click on the back arrow to navigate them back to the dashboard.

Movie Detail

Google Nexus 5X emulator- Movie Detail

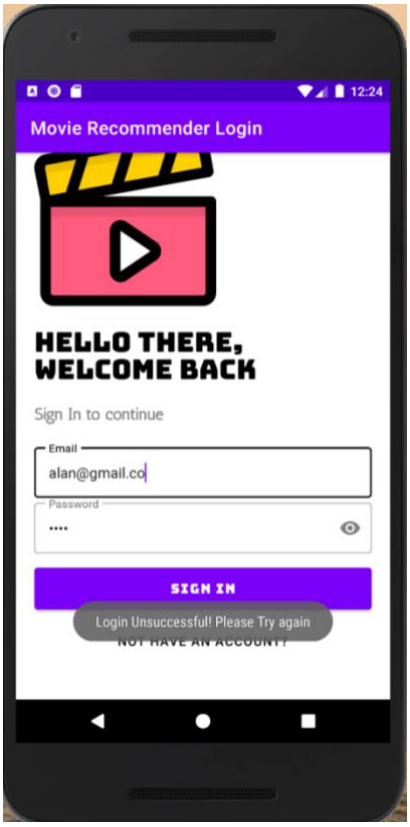
Movie Trailer



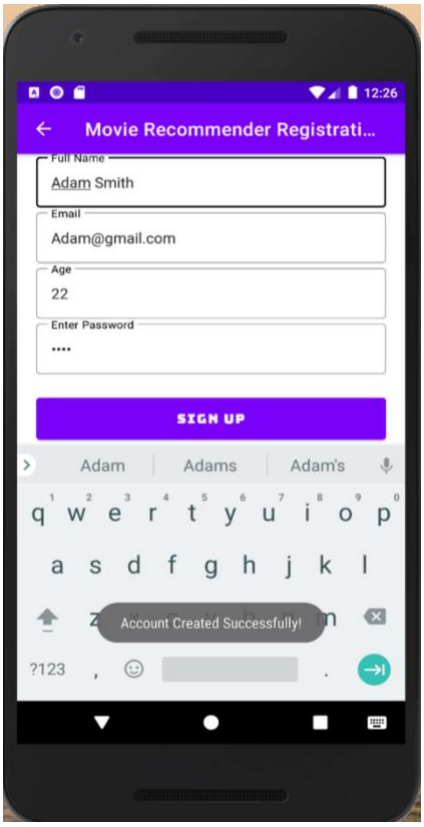
Google Nexus 5X emulator- Movie Trailer

The images above show the Movie detail page and the Movie Trailer page. The Movie detail page can be accessed when a user clicks on one of the movies displayed on the Movies View holder page. The Movie detail page illustrates the image from the movie, the movie title, the movie rating, the movie overview and finally, the trailer. The Movie Trailer can be accessed once a user clicks the 'Show Movie Trailer' button, transporting the user to the specific movie trailer using YouTube. This was done so that the user could know what the movie entails before watching it.

Notification Pop-ups



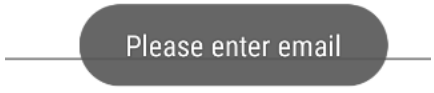
Google Nexus 5X emulator- Login error



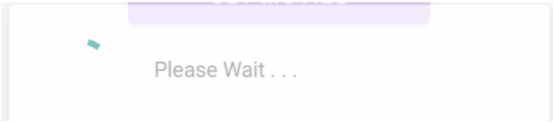
Google Nexus 5X emulator- Account created

Registration page- missing Full name error

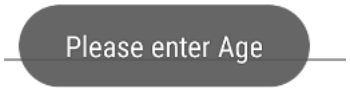
Login page- Successful login



Registration page- missing Email error



Please wait label- all pages



Registration page- missing Age error

Registration page- Password error

Evaluation of Jakob Nielson's Ten Usability Heuristics

Heuristics evaluation "is a usability engineering method for finding the usability problems in a user interface design so that they can be attended to as part of an iterative design process" [24]. This means the Heuristic evaluation aims to seek compliance of usability principles, and this is conducted typically with a small set of evaluators (3-5), who examine the application individually then come together to discuss their findings. As this is an individual project, the author will conduct his own review by focusing on a few key areas.

Before the author conducts the Heuristics, it is essential to understand the advantages and limitations of using this method.

One way heuristics evaluation is important and distinct from traditional user-testing such as Think Aloud is that the evaluators are, in essence, experts in their domains. Hence, these systems are evaluated in-depth and provide critical feedback.

However, researchers such as Bailey would refute this notion as he cites that "up to 43 potential changes" occurred during a heuristic evaluation that was no problems, somewhat false alarms [25]. This demonstrates the need for user testing to work in tandem with heuristic evaluation, as even experts are inclined to make mistakes.

Despite this, Tatari asserts this method is still used in industry as it is "cheap", and there is "no requirement for advanced planning" [26]. Thus, despite having its flaws, it is still valid when conducting tests.

There were three phases to the evaluation. The first comprised of being familiar with the domain of use, so familiarising with the application and similar applications. Then the second phase consisted of evaluating the application individually through navigating how to request a movie recommendation. Then the author would highlight each of the ten heuristics, indicating the severity rating and any recommendations.
There were three phases to the evaluation. The first comprised of essentially being familiar with the domain of use, so familiarising with the application and similar applications. Then the second phase consisted of evaluating the application individually through navigating how to request a movie recommendation. Then the author would highlight each of the ten heuristics, indicating the severity rating and any recommendations.

1. “Visibility of system status”.

The First heuristic to focus on is "Visibility of system status". The above page provides all the examples of notification pop-ups used in the application, fulfilling the first heuristic. This is fulfilled through notifying the user with appropriate feedback on what is going on in the system. As communication throughout the application helps make the user aware of the current system status with their requests. The rating would be 0, inferring there is not a problem with this heuristic. The only improvement the author would suggest is perhaps changing the colour of the notifications to red to be more clear.

2. “Match between system and the real world”

The second heuristic to focus on is "Match between system and the real world". This heuristic is important to users as this breeds familiarity and comfort if this has been fulfilled correctly. Hence, the importance of this heuristic. Although, the author believes this has not been as adequately executed as he would have liked to. For instance, with error notifications, usually, these systems will be established with a logo, and the logo is usually in red to mark as an error as this is universally known. On the other hand, the application does include familiar language and clear top bar titles and buttons. Overall, the author believes the rating to be 2 for this heuristic.

3. “User control and freedom”

The third heuristic focuses on "User control and freedom". This heuristic is imperative as users want to feel they can have control of the application with an easy exit process without having to commit to anything. Therefore, no arduous set of tasks to leave the application. The author believes again that this could be improved next time with a clear exit application button. As the emulator already has this hardcoded, the author did not think to incorporate one, given there are many back buttons to navigate out of the application. However, this could have perhaps improved the author's perception of achieving this heuristic. Consequently, the author has decided to give this rating a 2.

4. “Consistency and standards”

The fourth heuristic focuses on "Consistency and standards". This heuristic is essentially reviewing whether it fits industry standards. For example, as previously mentioned, when reviewing existing systems such as Taste and MovieLens, there is usually a login and registration page with some form of the filtering process, and this falls under the typical industry standards for movie recommendation services. As the author has these features included in the application, the author has evaluated this section with a rating of 1.

5. “Error prevention”

The fifth heuristic is "Error prevention". This heuristic ensures that the design practises ensuring there are steps considered to prevent an error from occurring in the first place. This is emphasised through the notification pop up section. As discussed earlier, when a user enters incorrect information upon registration or login, the following error messages will appear (refer to images in the notification pop up section).

6. “Recognition rather than recall”

The sixth heuristic is "Recognition rather than recall". This insinuates that the user should not experience a substantial cognitive load when interacting with the application. To combat this, the author has used top bar labels throughout the application to indicate where they currently are in terms of the application pages. This is combined with visible and clearly labelled buttons which navigate the user around the different pages. The author has decided to rate this section a 1 as the author believes the application does not require much cognitive load as it is very intuitive to use.

7. “Flexibility and efficiency of use”

The seventh heuristic is "Flexibility and efficiency of use". This implies that the user should have accelerators such as keyboard shortcuts, personalisation, and customisation. The author believes that although traditionally, accelerators would fulfil one part of this heuristic, given this is a mobile application. It would not make sense to incorporate keyboard shortcuts. However, some touch gestures are incorporated in the future, such as swipe left to go back. In terms of personalisation, the author believes this could have been improved in the future as the movie suggestions could come with a personalised message for the user. For example, "Hi Adam, I hope you enjoy these movie recommendations". Lastly, in terms of customisation, as users are allowed to sift through both genre and year of movies, the author believes this suffices, and the overall rating is 2.

8. “Aesthetic and minimalist design”

The eighth heuristic is "Aesthetic and minimalist design". This infers that there should not be any distracting elements that take away from the design, and the design should prioritise the key features and content to support the primary functions. The author's opinion is that it fulfils these requirements through a consistent style, colour scheme, and fonts. Although it could be argued as this is possibly the most subjective of all the heuristics, it comes down to personal taste. The author also believes it is relatively minimalistic in design and not too overcrowded in terms of the application. Therefore, the author has decided a 1 for this rating.

9. “Help users recognize, diagnose, and recover from errors”

The ninth heuristic is "Help users recognise, diagnose, and recover from errors". This is slightly different from the previous heuristic of "Error prevention" as this heuristic is more about the clarity of your errors. The author believes that although there are multiple error pop-ups, the specificity of the errors could be improved. For instance, if a user enters their full name and enters a number, it does not say the full name can only contain letters. Perhaps the application in the future could also offer shortcuts to errors. For instance, if you have missed the full name, click here to enter the full name again instead of directing yourself to that box.

10. “Help and documentation”

The final heuristic is "Help and documentation". This heuristic seeks a clear explanation of how the system works, whether that be a help button or a tutorial. The author has attached a user manual to the appendix of this document. However, there is no help button on the system. The user also has not attached a tutorial to the application as the author presumed it was relatively self-explanatory, but this could be implemented at a future date. Therefore, the author's opinion is that for this section, it should be considered a 2.

3.3 Implementation

The application was built in several stages, as the learning curve was steep, given that the author did not have experience with Android Studios. He completed the "Android Java Masterclass" on Udemy to assist him.

In this section, the author will explain each of the classes and how the system operates. Initially, the idea was to use IMDB API to connect to their database, however, when the author attempted this, it was apparent that it would not allow the user to filter through genre,

and it was rather challenging to use. Hence the author decided to proceed with TMDB API; this is discussed later.

3.3.1 Manifest File

The Manifest file is a file where all the key information about the app is described in. This includes where the application package name is and the app's components, such as all the activities and their designated classes [27]. This also includes the permissions of the app to access the internet. Lastly, any intent filters that allow for a launcher activity to occur is the first screen that is viewable to the user.

3.3.2 Login Page

The login page is the first screen that appears to users, once the user has registered this information is added into an array list into the firebase database [28]:

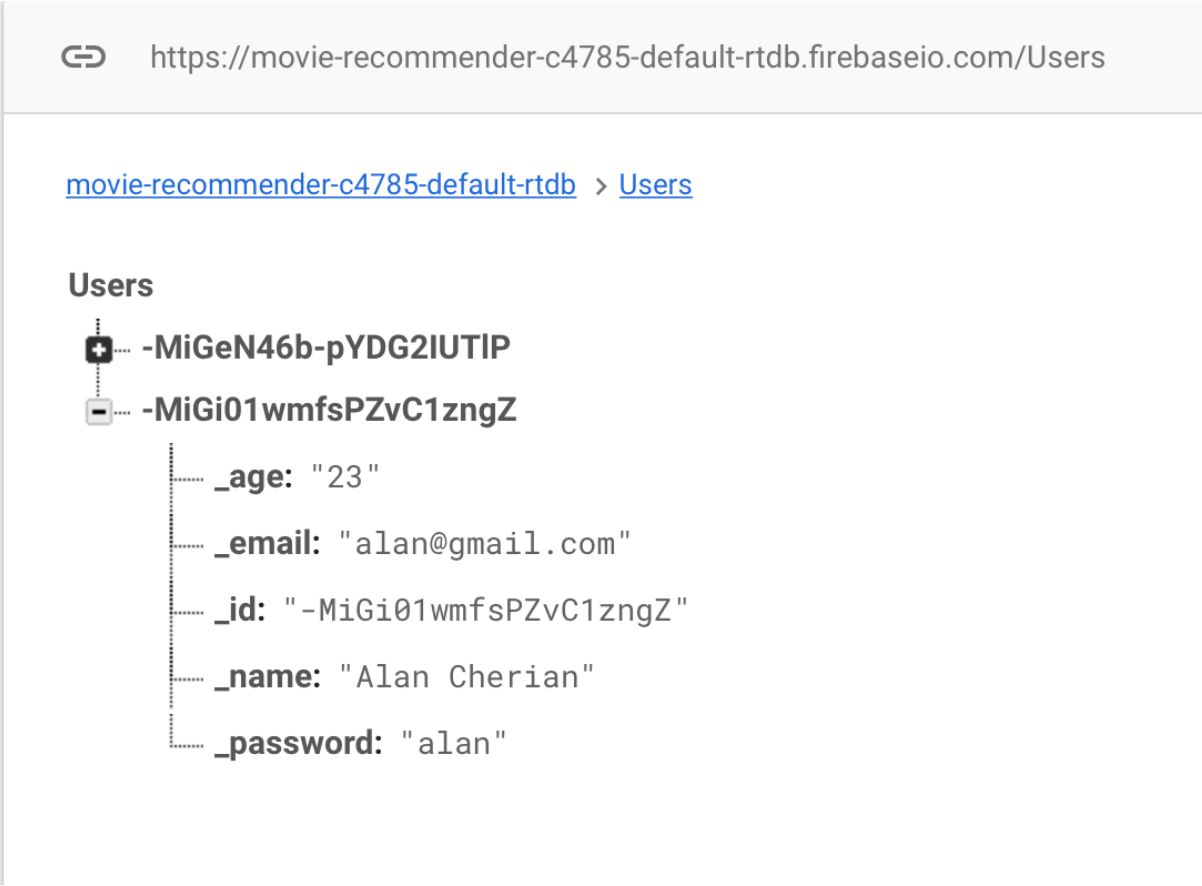


Figure 30- Google Firebase Database users

ArrayList<UsersInfo> usersInfos = new ArrayList<>();

This is then located through the following code from the onCreate method which instigates the activity using the login page (see below for the code and image of activity_login layout) [29]:

```
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_login);
this.setTitle("Movie Recommender Login");
// get instance of firebase database
firebaseDatabase = FirebaseDatabase.getInstance();
// get reference of particular location from firebase database
databaseReference = firebaseDatabase.getReference("Users");
email = (TextInputLayout) findViewById(R.id.txt_email);
password = (TextInputLayout) findViewById(R.id.txt_password);
btn_signin = findViewById(R.id.btn_signin);
btn_register = findViewById(R.id.btn_register);
```

After this the Onclick listener is instigated. This reads if the user clicks the sign in button. This then goes into the Helper class to check if the email format is valid:

```
public static boolean isEmailValid(String email) {
String expression = "^[\\w\\.-]+@([\\w\\-]+\\.)+[A-Z]{2,4}$";
Pattern pattern = Pattern.compile(expression,
Pattern.CASE_INSENSITIVE);
Matcher matcher = pattern.matcher(email);
return matcher.matches();
```

After this has been checked, the onDataChange() method is used to read a static snapshot of the contents path, as they existed at the time of the event. Thus, this means it is checking the current database data. This method is triggered once when the listener is attached, and again every time the data changes. The User array list is then cleared to ensure it is updated. Then by using the ValueEventListener will return the entire list of data as a single DataSnapshot, which you can then loop over to access individual children. Then the get.value() method is used on the snapshot, which returns the Java object representation of the data. This is then added to the updated array list.
This array list is then iterated over to check the email and passwords match up with the emails and passwords on the database. If successful, then the user will move into the next activity which is the Activity Dashboard. If not the user will be informed with an error message stating the login in was unsuccessful.

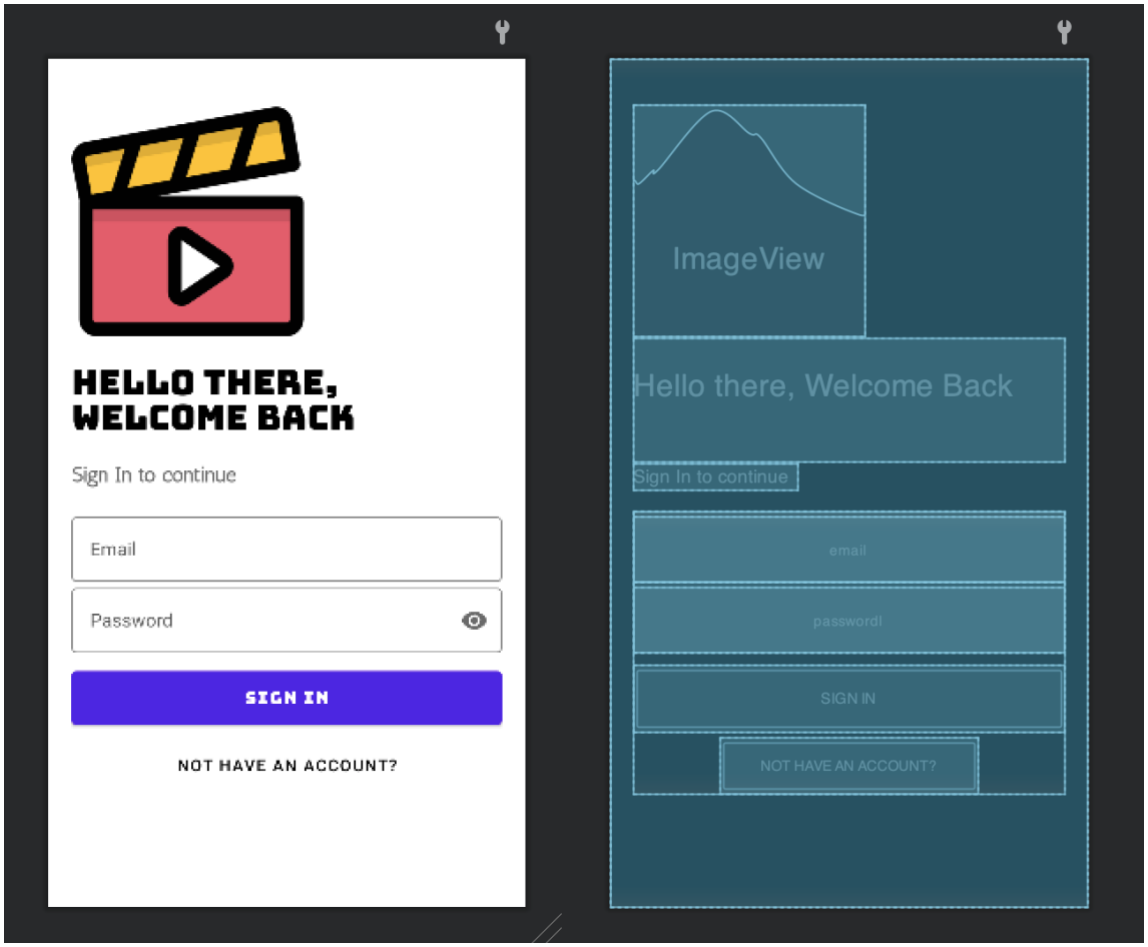


Figure 31- Login Page Layout- Design and Blueprint

3.3.3 Activity Dashboard

The activity dashboard consists of an array of categories for all the genres:

```
String[] movie_category = { "Adventure", "Animation",
"Comedy", "Drama",
"Family","Fantasy","History","Horror","Thriller"};
 // created an array of codes according to category
 String[] category_code = {  "12", "16",
"35","18","10751","14","36","27","53"};
```

Each category was assigned a code. For example, Adventure was assigned code 12. The same was done for movie year, but instead of codes, it was selected position, as they were numbers, hence, were all integer values.

Then to display them as in the image of the Activity Dashboard Layout, drop-down menus had to be integrated. To do this, we have to initialise the Spinner object [30].
The Spinner object then needs to be populated with the arrays of both genre and year, this is done by using an instance of the Array adapter.

Then to respond to the user selections to the movie and year the 'AdapterView.OnItemSelectedListener' interface and the corresponding 'onItemSelected()' call back method is called in order to record the options of the user. This is then passed into the Main Activity class.

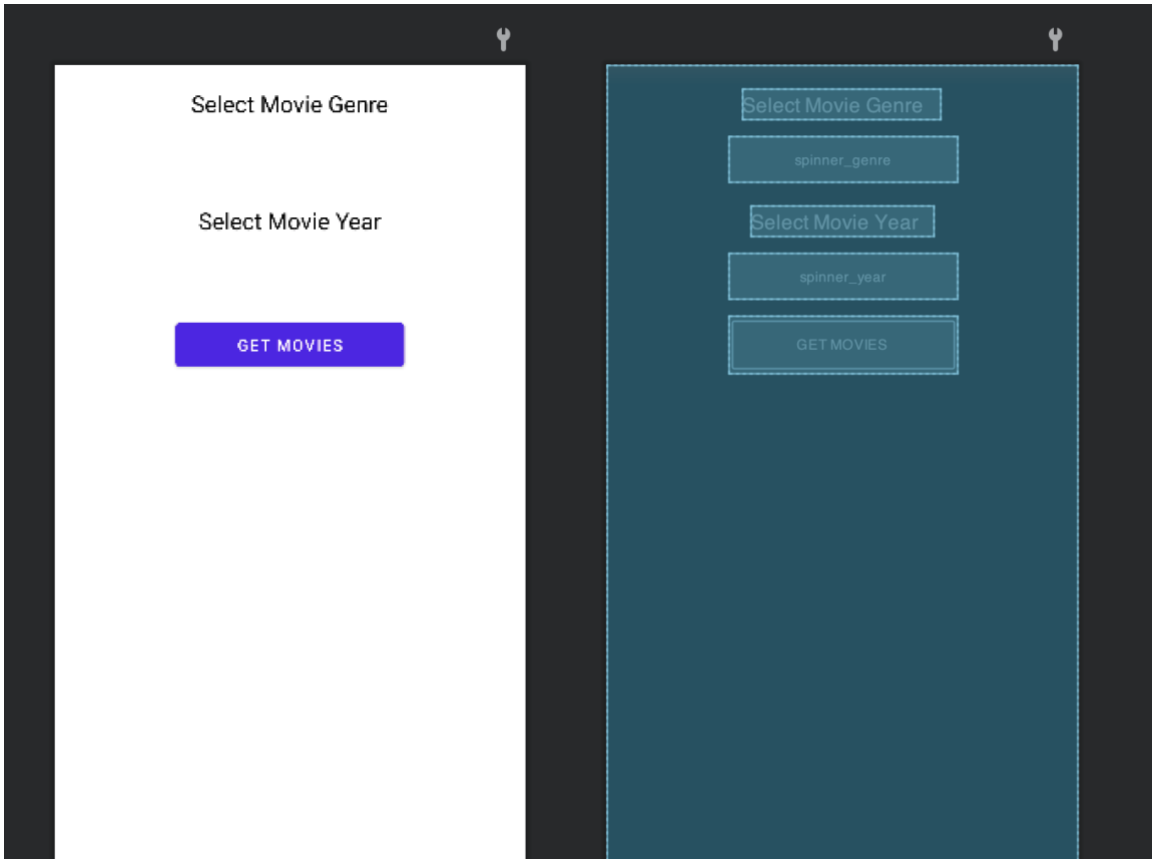


Figure 32- Activity Dashboard Layout- Design and Blueprint

3.3.4 Main Activity

In the main activity, an array is created for the movie list based on the selections of the user from the activity dashboard. Then the Recycler view is implemented, a scrollable container for displaying large amounts of data and arranging the movie list in a different format. This is specified as a grid layout with two items in each row.

Then we make a get call request to fetch data from the TMDB API using an API key (‘*cf33d2437d41aa3d4afcb9525b0fd6bf*’) which we are using [31]. Then we send two things into URL to get particular response. In our case we need movies of specific category and year, therefore we pass those two parameters in it:

```
newGetAsynTask().execute("https://api.themoviedb.org/3/discove
r/movie?api_key=cf33d2437d41aa3d4afcb9525b0fd6bf&with_genres="
+
extras.getString("genrecode")+"&year="+extras.getString("year"
));
```

The AsyncTask (Asynchronous Task) [32] allows us to run the instruction in the background and then synchronises again with our main thread. This class will override at least one method i.e doInBackground(Params) and most often will override second method onPostExecute(Result).

AsyncTask class is used to do background operations that will update the UI (user interface). Mainly we used it for short operations that will not affect our main thread.

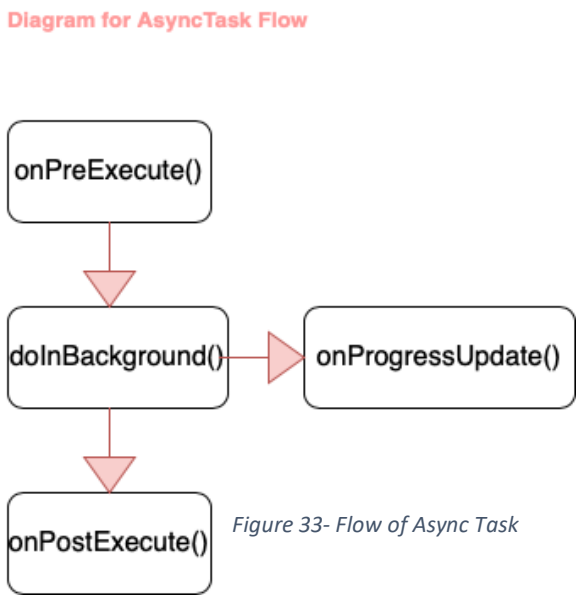


Figure 33- Flow of Async Task

To ensure we do not affect our user interface, we ensure these server requests are made through a background worker class. This is because all the server requests we deal with is on the background thread. Hence, we have the onPreExecute method, which is required before starting the background work, then a doInBackground method executes the server request to get our required data. Then finally, we have an onPostExecute method that receives a response in JSON format, then we are parsing that JSON with key values already defined in the JSON data.

So first, we create a JSON object for parsing our JSON data in the form of a string. Our received data is in the form of an array. Then we loop this array to iterate through all JSON arrays to get our required movie details (found in movie class).

After parsing all this data, we save all the data into a list. If the list size is greater than zero, we pass the data to the movie_viewholder class to show all data in the form of a list, and of

course, this has been adapted to the recycler view. As previously mentioned, this will be displayed with a grid layout.

3.3.5 Movie Viewholder

In the movie_viewholder class we have the method that displays the data at the specified position. This method is used to update the contents of the itemView to reflect the item at the given position. Also, in this class we have a specific intent which directs the user to the Movie_detail class which contains all the data including the title, movie cover picture, rating, brief description, and movie trailer. To get the picture of the each of the movie covers we use the following:

```
Picasso.get().load("https://image.tmdb.org/t/p/w500"+_movies.g
et(position).getCover()).into(holder.movie_cover);
```

Then if the user wants to see more information on the particular movie, the user can click on the desired movie tab, and this will take them into the movie detail activity.

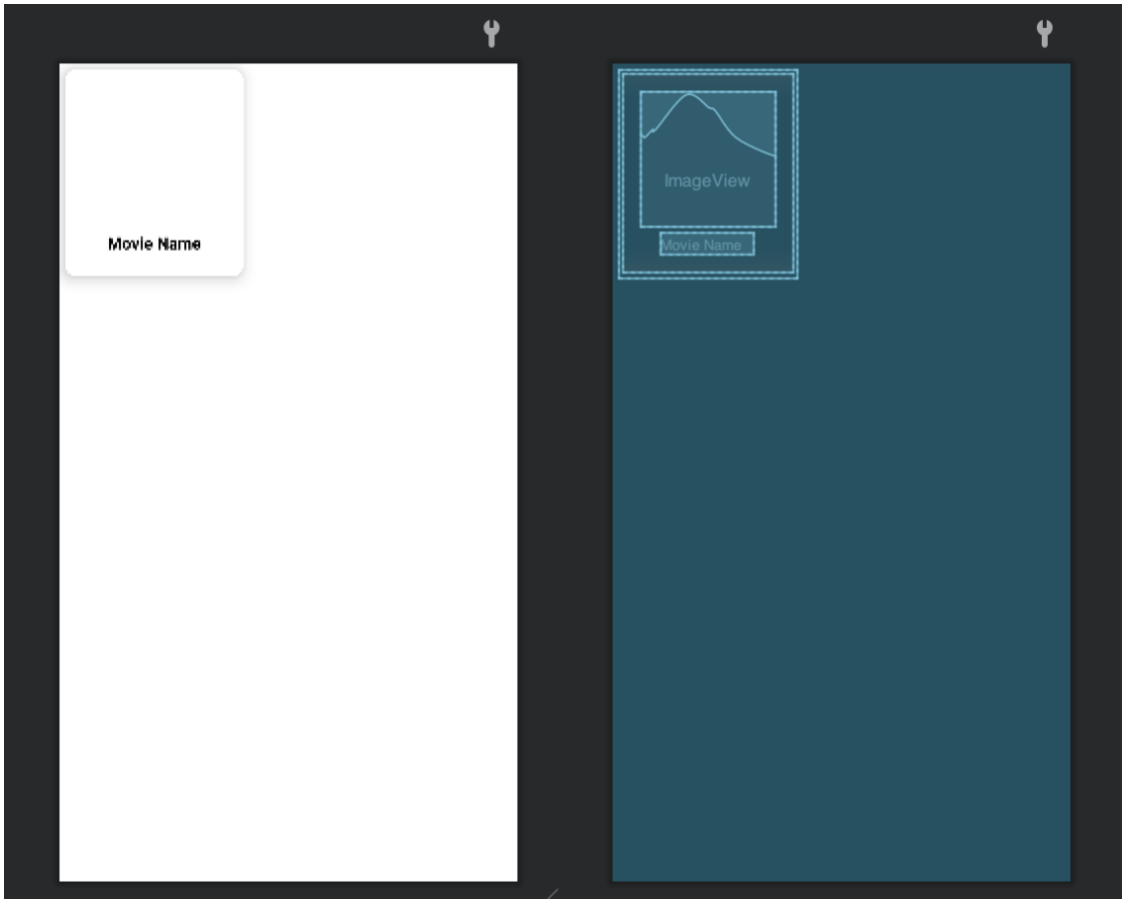


Figure 34- Movie View Holder Layout- Design and Blueprint

3.3.6 Movie Detail

In this class there are several pieces of information as it contains all the data including the title, movie cover picture, rating, brief description, and movie trailer. Each of these pieces of information are identified by a unique id.

Similar to the main activity class to ensure we don't affect our user interface we have to have background worker class to make another API call to access the movie trailer:

```
newGetAsynTask().execute("https://api.themoviedb.org/3/movie/"
+ movie_id +
"/videos?api_key=cf33d2437d41aa3d4afcb9525b0fd6bf");
```

Again, we must implement an onPreExecute method this includes a please wait by accessing the helper class. Then we move on to the doInBackground method [32].

Lastly, we then move on to the onPostExecute method, which gets a response from the server API after the request then the JSON response is parsed into a JSON object. Then they are put into an array, and each trailer is given a key and then passed into the movie trailer class.

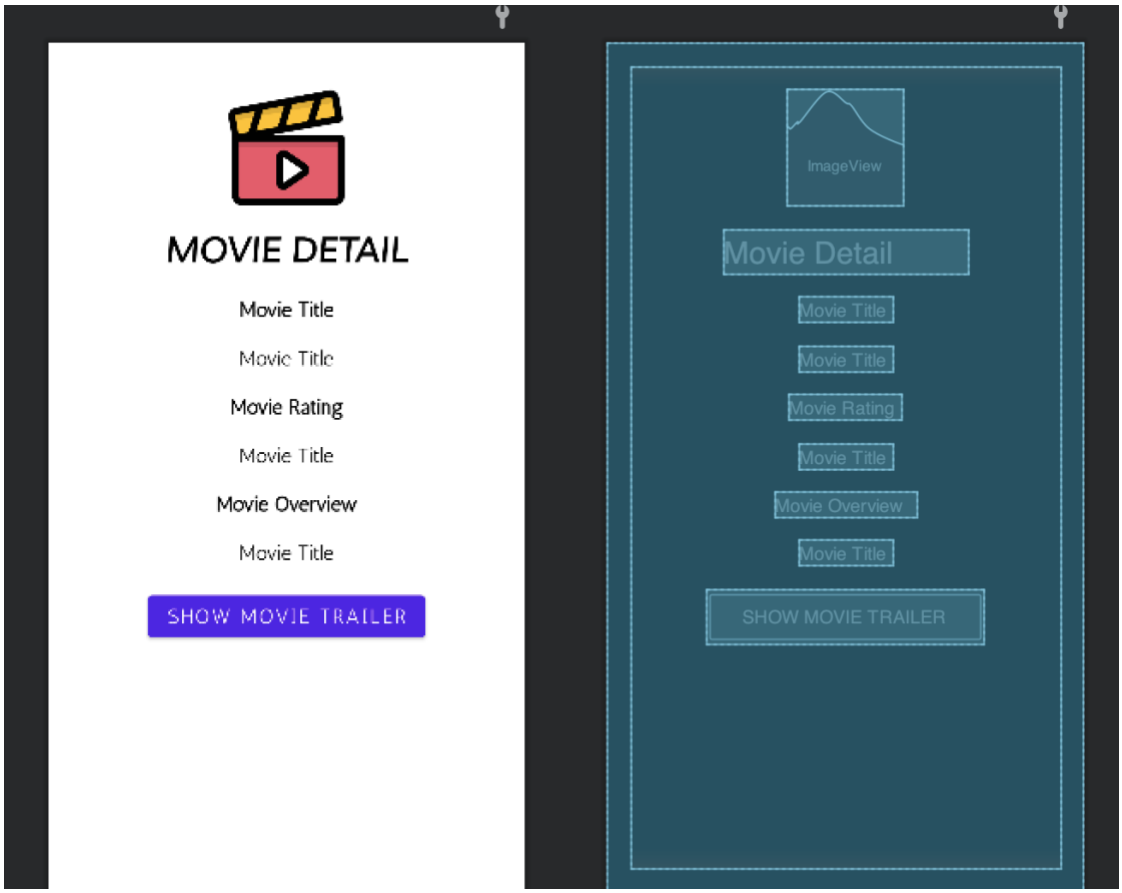


Figure 35- Movie Detail Layout- Design and Blueprint

3.3.7 Activity register

The registration page operates like the login page. The same firebase reference is instigated to locate the database.

After this, the onCreate method is instigated. This initialises the activity using the activity_register layout (see image below). Then the code refers to a new User info class being created and stores all the variables within that user. Then the Onclick method is instigated when users click the sign-up button.

Then the register new user method is instigated to check whether users have entered a valid name, email address, password, and age. If any of the values for this is null, an error will occur, as all the variables are not null for the registration process to be finished. If the user is successful in registration, a "Toast" message will pop up stating the account was created successfully, and the opposite will occur if it is not successful. Then users must navigate back to the login page to start the process.

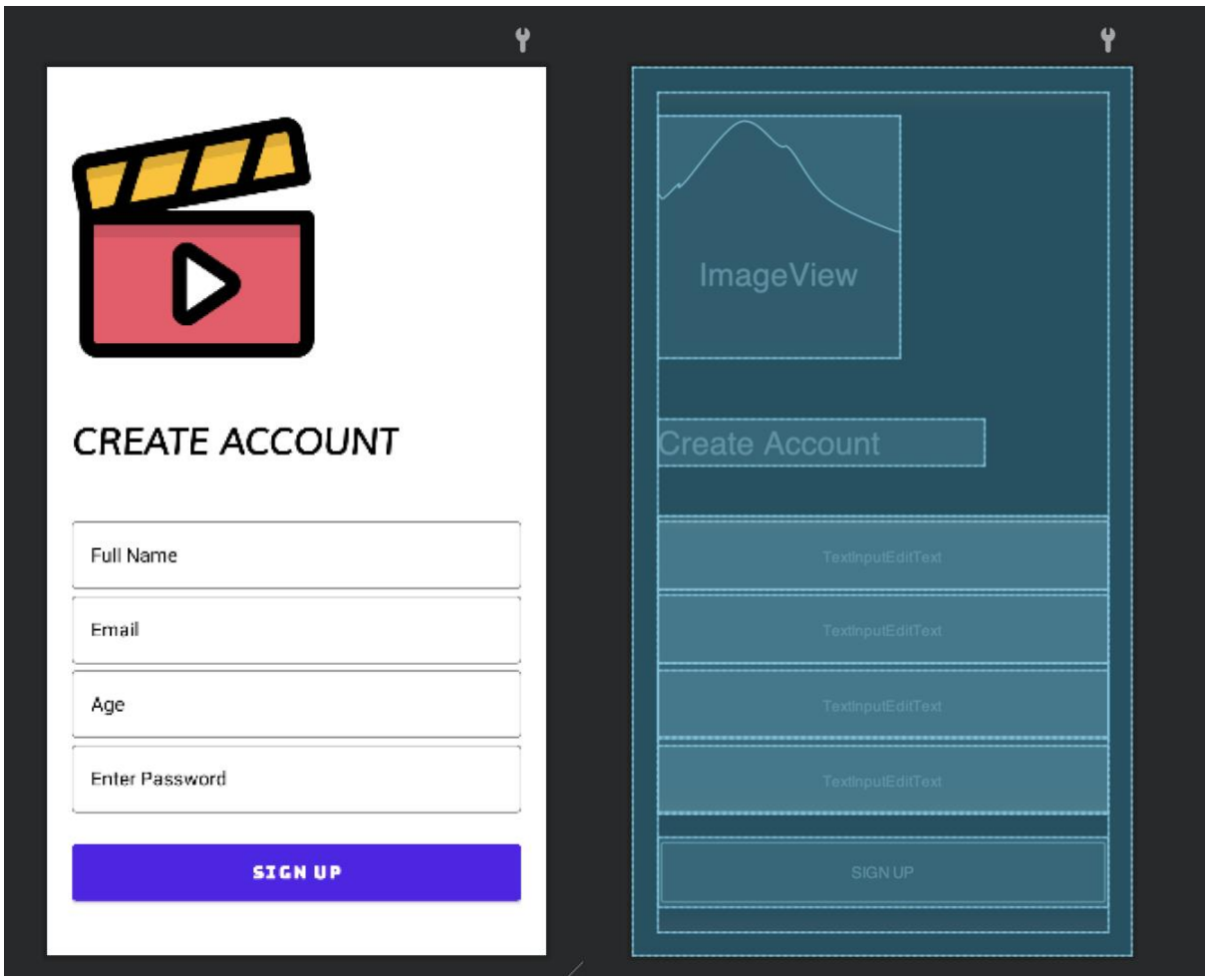


Figure 36- Registration Layout-Design and Blueprint

3.3.8 Helper class

This class is where the HTTP URL connection is taken place. This class sends an HTTP request and gets the HTTP server response in the Android application [33].

To start this process a URL instance is created:

```
URL urlCon = new URL(url);
```

Then we open the URL connection and cast it to the HTTP URL connection:

```
HttpURLConnection connection = (HttpURLConnection)
urlCon.openConnection();
```

There is an industry-standard for device identification that apps use. It is the User-Agent string [34]. The User-Agent string is a field in the HTTP header that can identify the app that is making the request. In a web browsing experience, the User-Agent is set by the browser (which is also an app). Now, instead of a browser, our native app will be making the HTTP request. Hence, we need to take steps to ensure our User-Agent includes meaningful information (Android and what version). Otherwise, the app will get the default values, which is not very helpful in any context.

We then ensure the connection request is set to request a GET, then use below code to open input stream and read server response data:

```
HttpURLConnection connection = (HttpURLConnection)
urlCon.openConnection();
connection.setRequestProperty("User-Agent", "Universal/2.0
(Android)");
connection.setRequestMethod("GET");
connection.setDoInput(true);
connection.connect();
```

To maintain a long session between client and server we have implemented cookies.

3.3.9 Movies Class

This class is essentially for getting and setting the values of the movie data, which includes the rating, the overview, the id, cover photo and name. A constructor is created to instantiate these objects. This class is used in the main method for processing the data from the API and passing into the movie_viewholder class.

3.3.10 Users Info

This class shares similarities with the movie class as it is used to get and set the user info values. This includes the users fullname, email, age, password, and id. Again, constructors are used to instantiating the objects. This class is used in the login and registration activities.

3.3.11 Build Gradle

There are two Gradle files. The project level build.gradle and the module level build.gradle.

The project level build.gradle is in the project's root folder, and its configuration settings apply to every module in the project.

The module level build.gradle allows you to configure build settings for the specific module it is located in. Configuring these build settings allows you to provide imported libraries such as the library for the firebase database and the library for loading images from the server. This is done in the dependencies section. In this file, you can also add plugins and compile the correct SDK version. The build.gradle also allows you to override the main/ app manifest or top-level build settings.gradle file [35].

3.3.12 Testing

Testing is a critical section of the software development process; extensive testing was carried out throughout the product life cycle. As the Agile method is integrated with this project, each functionality would have to be tested prior to adding any other functionality.

As the nature of the project was consistent testing, some of the ongoing testing's were not documented. However, all the functionality testing was recorded.

Table 1 depicts the testing that took place. The first column, 'Test Summary', is a description of what test was conducted. The second column, 'Expected Outcome,' describes what should happen when the functionality takes place. The 'Actual Outcome' explains what occurred, and the last column confirms whether the test passed or failed.

Table 3- Testing

Test and Results Table			
Test Summary	**Expected Outcome**	**Actual Outcome**	**Pass or Fail?**
Login Screen: Entering a username or password that has not been registered on the Firebase database	The user will not be able to login to the android application.	After the user presses login, the following error message appears "Login Unsuccessful please try again",	Pass
Sign up: Pressing the sign-up button	The user will be redirected to the registration page.	User is redirected to the registration page.	Pass
Registration: Pressing the register button	The user will have added their details to the database.	The user will have added their details to the database, this is confirmed with a "Account created successfully".	Pass
Registration: Any input data on full name, age, email, or password which is	The user will not be able to register an account if any of this	The user is not able to register, and an appropriate error message is shown on	Pass

missing, the user will then be unable to proceed to register	information is missing.	the screen depending what field of data is missing, for example if the full name is missing the error message will state "Missing full name please enter full name"	
Registration: The password must be longer then 6 characters to ensure safety	The user will not be able to register with a password less than 6 characters.	The user is not able to register with a password less than 6 characters. If this is not fulfilled an error stating "Password length must be above 6 characters".	Pass
Firebase Database: The data from the registration matches the data in the database	The user data will be passed into the database with no errors.	The user data is passed into the database with no errors.	Pass
Firebase Database: The admin can delete users	The admin should be able to delete any users.	The admin can delete users once logged into the firebase database.	Pass
Back button: The user can navigate back throughout the application	The user shall be able to navigate back to any section of the application.	The user can navigate back to any section of the application apart from the first page which is the login (launcher page)	Pass
Activity dashboard: Genre and Year drop down work	There should be a drop-down menu for a selection of the different genres available to the user and the same with the year. This should function smoothly with no lag.	The drop-down menu is working efficiently with no lag and each selection is saved into the application via cookies.	Pass
Activity dashboard: Get movies button	This button should request the movies given the selection of genre and year; this should redirect the user to the movie_viewholder page.	The user is redirected to the movies_viewholder page with the given results for the selection of genre and year	Pass

Navigation bar for Viewholder: displays genre and years selected	The navigation bar (top bar) should change given the different pages.	Each page has a different title, for example login displays “Movie Recommendation Login” and once this has been redirected these changes correspond with the different activity pages.	Pass
Main Activity: API connects to the TMDB database with the correct	The API request should parse all the data given the parameters stated.	The API parses all the data successfully and all the data is saved as a list in the view holder class	Pass
Helper class: Show loader is displayed	If there is any delay in either logging in or registering and getting the movies from the API a display should inform the user to wait.	The display performs only where there is a short wait and informs the user to “Please wait” with a loading sign.	Pass
Movie_viewholder: Displays the correct movies for the given genre and year in grid layout structure	Given the selection of the user the correct movies are displayed in a grid layout.	The correct movies are displayed given the genre and year, in a grid layout structure.	Pass
Movie_viewholder: Pressing on an individual movie to gain further detail	The user should be able to select a movie to gain further detail on it.	The user can access further detail on the movie when they click on the individual movie, this is then redirected to the Movie_detail page.	Pass
Movie_detail: Displays the brief description, cover photo and rating	This should display to the user further detail about the movie including the overview, cover photo and rating.	Illustrates clear synopsis of the movie and includes the cover photo and rating.	Pass
Movie_detail: Show Trailer button	This button navigates to the YouTube Trailer for the specific movie	The button navigates successfully to the YouTube Trailer for the specific movie, however some of the links are outdated so may show the unofficial trailer link	Pass but could be improved

		for some of the movies depending on the year of release.	

Alongside testing the functionality of the application, the author also conducted Junit validation tests [36]. One of which is demonstrated below:

```
package com.test.movierecommender;

import org.junit.Before;
import org.junit.Test;

import static org.junit.Assert.*;

public class ValidationTest {

    Helper helper;
    activity_login activity_login;
    @Before
    public void setUp() {
        helper=new Helper();
    }
    @Test
    public void check_email(){
        String email="test@gmail.com";
        assertTrue(helper.isEmailValid(email));
  }
```

This is in the helper class:

```
  public static boolean isEmailValid(String email) {
        String expression = "^[\\w\\.-]+@([\\w\\-]+\\.)+[A-
Z]{2,4}$";
        Pattern pattern = Pattern.compile(expression,
Pattern.CASE_INSENSITIVE);
        Matcher matcher = pattern.matcher(email);
        return matcher.matches();
    }
```

This test is done to evaluate the email and whether this is in the correct format and that it matches, once ran the test results indicate that it is successful.

The author also tested the application in landscape mode to ensure all the details were still displayed and all the functionalities operated correctly.

4.0 Evaluation and Results

This chapter will discuss the results that were accomplished because of the efforts made on this project. This chapter will also present an outline and evaluation of the application and any design decisions that changed.

4.1 Evaluation of requirements

4.1.1 Functional Requirements

This section will examine the functional and non-functional requirements and whether they have been achieved in the application. Requirements that have been achieved will not be discussed as the implementation section provides details on existing features. However, requirements that have only partially been achieved will be discussed, and the reasons why they were not achieved will be examined.

Additionally, possibilities for how to successfully fulfil these requirements will also be counted. 100% of the high priority requirements were fully or partially achieved. 100% of the medium priority requirements were fully or partially achieved. Overall, 100% of the requirements were achieved.

Table 4- Evaluation of Functional Requirements

Requirement	**Priority**	**Achieved**
The user will create their login by using their full name, email address, password. Also, age will need to be implemented, which can be used later to filter movies within that age range.	High	Partially as it does not yet allow to filter movies within age range.
The application will use an API key to access movies from the TMDB database as the IMDB database does not allow filtering by genre.	High	Yes
The application will include different activity pages, including a registration page, a login page, a movie dashboard, a movie description page.	High	Yes
The user will access interactive elements such as external links to a movie trailer for the recommendation through the movie description page.	High	Yes, but could include more interactive elements

The user will be able to view the rating of each movie on the dashboard.	High	Yes
The user will be able to go back to search a different genre or year of the movie.	High	Yes
The app must also operate on a landscape view.	Medium	Partially as this could be extended further to use different size devices
The app will have a top bar explaining what is on the page.	High	Yes
Notification of registration when user successfully registers	High	Yes
Notification of successful login when user login	High	Yes

4.1.2 Non-Functional Requirements

Table 5- Evaluation of Non-Functional Requirements

Requirement	**Priority**	**Achieved**
When a user creates a login on the registration page, the username, password and age will be passed into the Firebase database with a unique ID associated with it to differentiate each user.	High	Yes
The application will store the user's session to keep the user experience smoother. This will be done by using the Datasnapshot feature in android studios. Which essentially operates as copies of the data and stores any data on the emulator to pass into the other activity pages and reads through the firebase database.	High	Yes
The application will keep a consistent colour scheme using only two main colours, black and blue. This	High	Yes

is so it remains easy to read and has a clean interface. The font scheme will be reviewed using Jakob Nielson's Ten Usability Heuristics to ensure good practice is exhibited in the application.		
The application will be running on a Google Nexus 5X emulator, which can be done through Android studios by downloading the required SDK tools for your device and creating an Android Virtual Device (AVD). Then in the toolbar, select the target device you want to run on and then click run program. To navigate around the emulator, you can use a keyboard and mouse.	High	Yes, but could include more device compatibilities
There will be a user manual. This will be accessed through appendix section A and will be in pdf format.	High	Yes
If any users need to be deleted, this can be done through the Google Firebase database by clicking on the Realtime database and then clicking the cross button, which deletes the user. Alternatively, deleting the user attribute will delete all users if all users need to be deleted. Equally, if any user data needs to be edited, the same process applies but click on the individual attribute of age username or password.	High	Yes
The application will be maintained and updated to add further functionality in the future.	High	Yes
Navigation will be simple so that a "Get movie" request for a movie will be placed	High	Yes

on the application in less than 5 minutes.		

4.2 Evaluation by questionnaire

There were some qualitative metrics recorded as each participant will be given a pre scenario quiz (refer to Appendix C). These qualitative questions were used to extract additional information that would supplement the results found in the questionnaire. The pre-scenario quiz will consist of some open-ended questions in which quotes can be taken from the user. These are intended to complement the information gathered from the questionnaire.

The questionnaire was provided to 20 individuals, and the number was not higher than this as we did not publish the app into the Play Store to allow access to more individuals. However, this number still allowed for a mixture of feedback. All participants who partook in the survey were known to the author, but the information was submitted anonymously. The participant's ages ranged from 18-34, which was the target audience for this application. They also came from a mixture of technically savvy and non-technical whilst ensuring a mixture of gender. Again, this was to ensure a comprehensive set of opinions and ideas. A total of nine questions were given to the questionnaire participants. Once the responses had all been received, the information was collated and displayed through a diverse range of bar charts and pie charts with long-form responses.

The overall responses were largely positive; the respondents also discussed improvements and suggestions, which can be considered for further product iterations. The author will go through each of the questions in this section and discuss the feedback given.

The first question was, “How satisfied were you with the mobile app?”. This question was asked to measure the level of satisfaction users were after using the application. As seen in the figure 37 below, the response to this question revealed that 10 in 20 (50%) of participants were very satisfied with the mobile application, with a further 10 in 20 (50%) claiming they were somewhat satisfied with the final product.

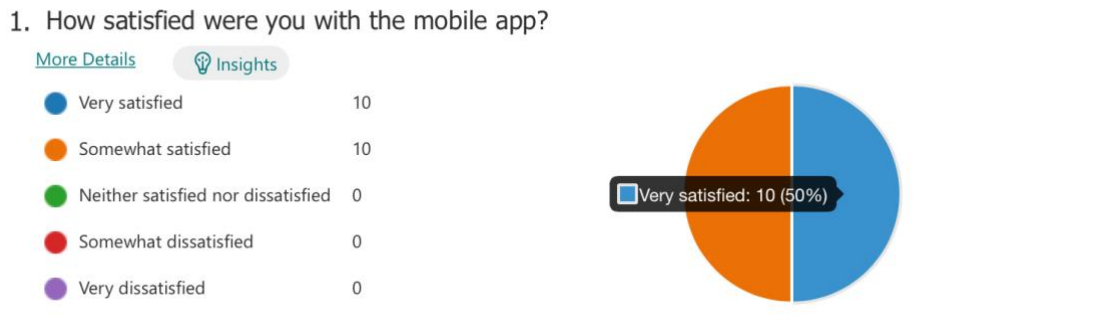


Figure 37- Illustrating user satisfaction levels of application

The second question was, “How does the application compare to any similar mobile applications you have used?”. This question compares the application to similar products on the market that users may have previously used before. As seen in the figure 38 below, the

response to this question revealed that 11 in 20 (55%) of had not used a similar mobile application, with a further 2 in 20 (10%) claiming they viewed the product as well above average to the applications like this. Along with 30% describing it as above average and 5% claiming it was average.

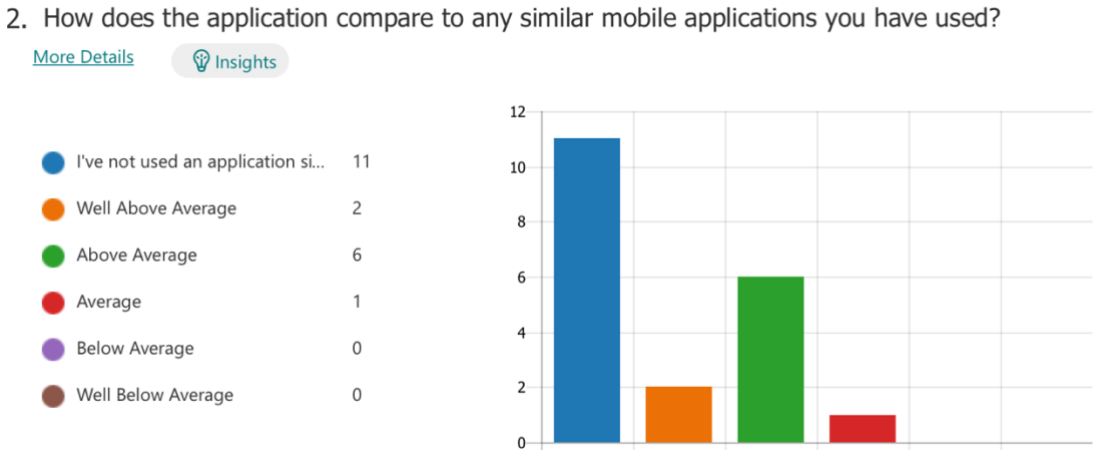


Figure 38- Illustrating the comparison between similar applications

The third question was, "How likely are you to use this as your primary movie-based application?" as seen by figure 39. The results here show that the respondents answered 50% for both very likely and 50% for somewhat likely. This could perhaps be explained by the fact that 55% of respondents have not used this kind of application before using this application. So, the reluctance to use one could explain how 50% was only somewhat likely. Another reason for this could be that users could be currently using an application which they have become accustomed to and have had long term use of a different application. This would suggest that they may not want to change.

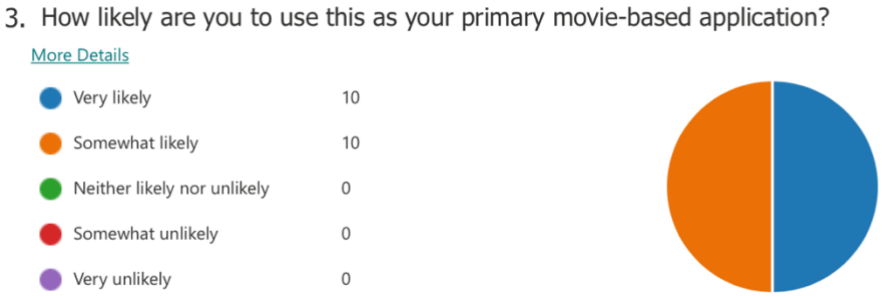


Figure 39- Illustrating likelihood of users using this as main movie application

The fourth question was, "Have you been experiencing any problems with the mobile app if so please describe?" as seen by figure 40. The results for this were varied, and one user forgot to input the answer for this question; hence there are only 19 responses. The following are the results, 13 out of 19 (68%) respondents answered, "No problems/ Not applicable", 4 out of 19 users (22%) said there was a "slight delay". Interestingly the delay for each user varied as one user experienced a slight delay in the login. Alternatively, another experienced a slight delay in the register, get movies had a slight delay and lastly show movie trailer also did. These delays could be explained by that each time the author showed the product to the user, it was on different days, so the application would need to start up again. This may take some time for all the connections to the database and API to restart. Lastly, two further delays had been

picked up by users. This was the "slight scrolling lag". This scrolling lag could perhaps be explained using a magic trackpad on the author's computer as some users could be used to using a mouse scroller, but this could also be attributed to an error that the author has not been able to amend as the emulator can only mimic the functions of an android phone and may not be best suited for the trackpad.

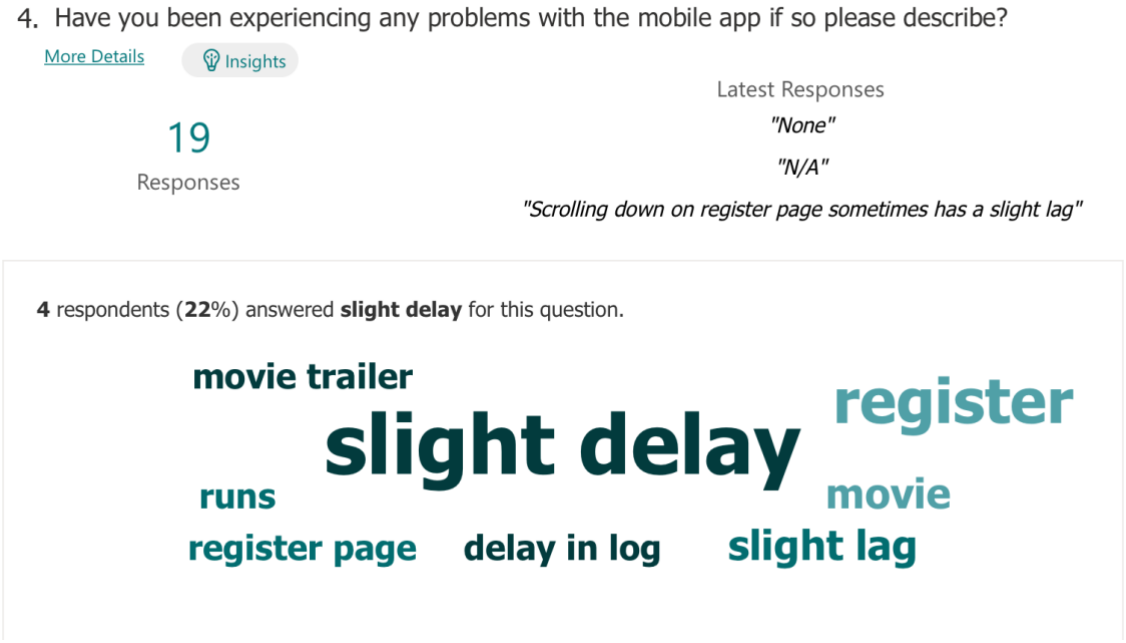


Figure 40- Illustrating users' problems with the application

The fifth question was, "How innovative would you say the application is?" as seen by the figure 41 below. This question was designed to ask whether users thought that this app was creative and pushing the boundaries in terms of what they had previously experienced. The results for the following question are 5 out of 20 (25%) respondents answered, "Extremely innovative", 14 out of 20 (70%) responded "somewhat innovative", and one out of 20 (5%) answered. The author's opinion of the results skewing to "somewhat innovative" stem from the fact that because most respondents are used to recommendation systems, whether that be for Netflix or YouTube, the idea of the application is not hugely innovative as it currently exists just in other forms.

Figure 41- Illustrating users' perception of how innovative the application is

The sixth question was "What do you like most about this product?" as seen by the figure 42. Again, the results for this were varied, and one user forgot to input the answer for this question; hence there are only 19 responses. This question was proposed to find out what features the respondents liked the most about the application. The responses ranged from design elements that users liked and, for example, the aesthetic and theme to the intuitive and straightforward ease they can be operated.

The seventh question was "What do you dislike most about this product?" as seen by the

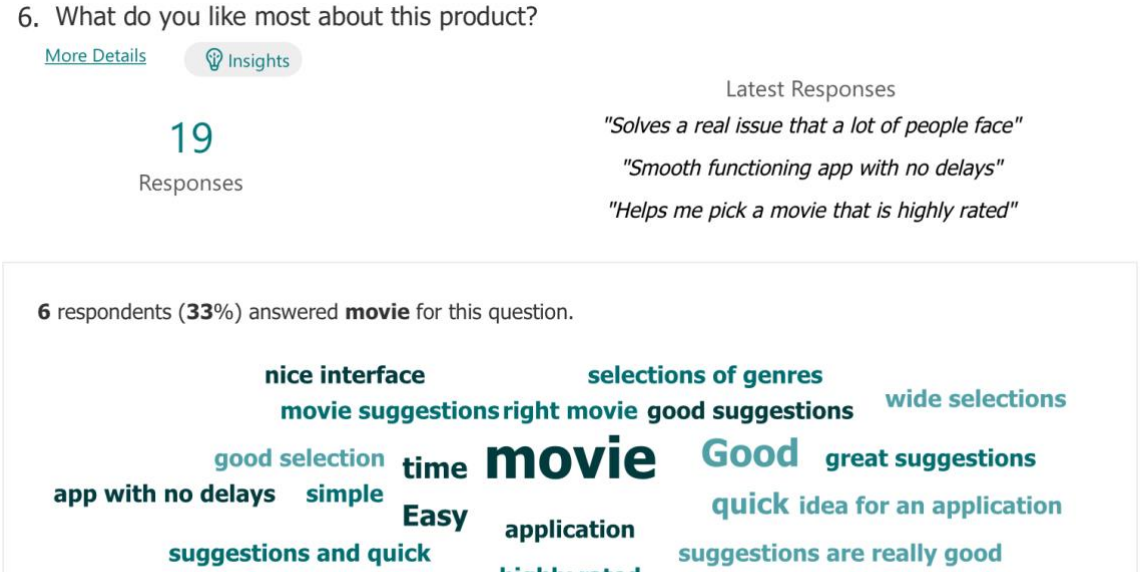


Figure 42- Illustrating users' favourite aspects of the application

figure 43. This was asked to gauge whether certain features could be either removed or altered at a later iteration. The author saw it necessary to ask this to ensure more critical comments were made on the application. Again, the responses were unique and ranged from some users preferring Rotten Tomato ratings to IMDB. An interesting suggestion was made to save suggestions of the movies to the individual account as this is not currently stored. Also, there was a suggestion to implement more filters for the movies, so the suggestions were less generic as it currently only filters genre and year.

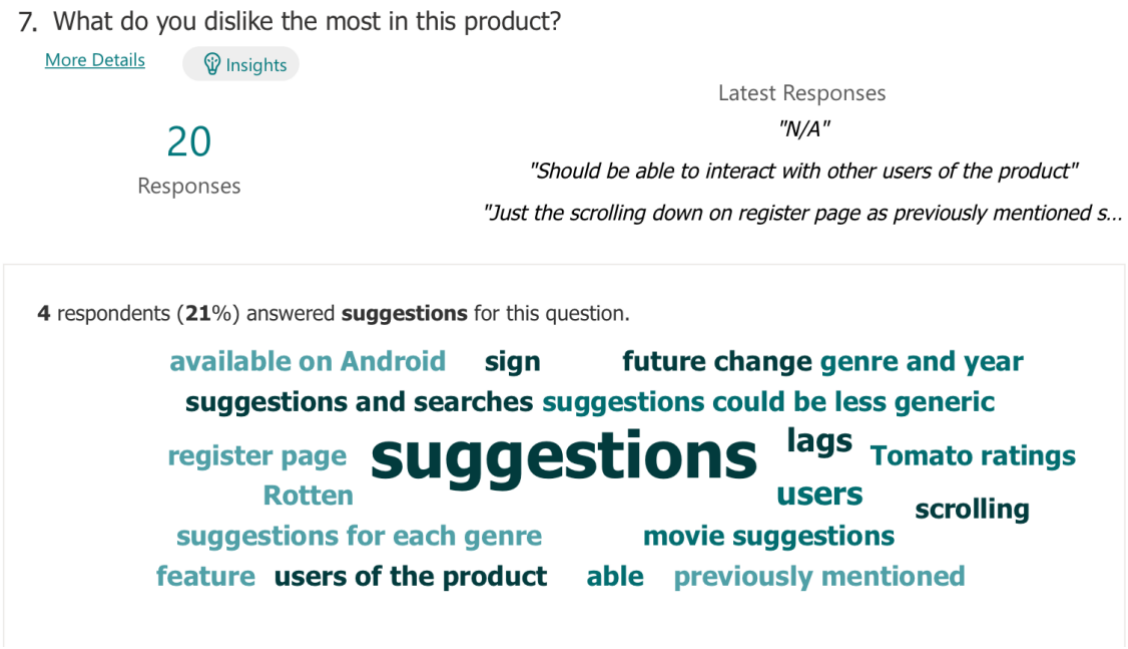


Figure 43- Illustrating the downfalls of the application

The eighth question was "How would you improve the application?" as seen by the figure 44. The responses towards this question were a mixture. The author has decided to list the ones that most appealed to him. The first one the author thought was interesting was that as users register, they must go back to sign in and instead, the user should just be redirected to the login page instead of navigating backwards. Another suggestion was that there should be a community thread in which users can chat with one another.

Additionally, there was a suggestion that users should select movies before registering to allow for more accurate suggestions. Like this suggestion, a user said the app could connect to Netflix or other streaming websites to access currently watched movies to offer more specific suggestions. Lastly, a user suggested the application should allow users to post their rating or even edit the current rating based on an average user vote.

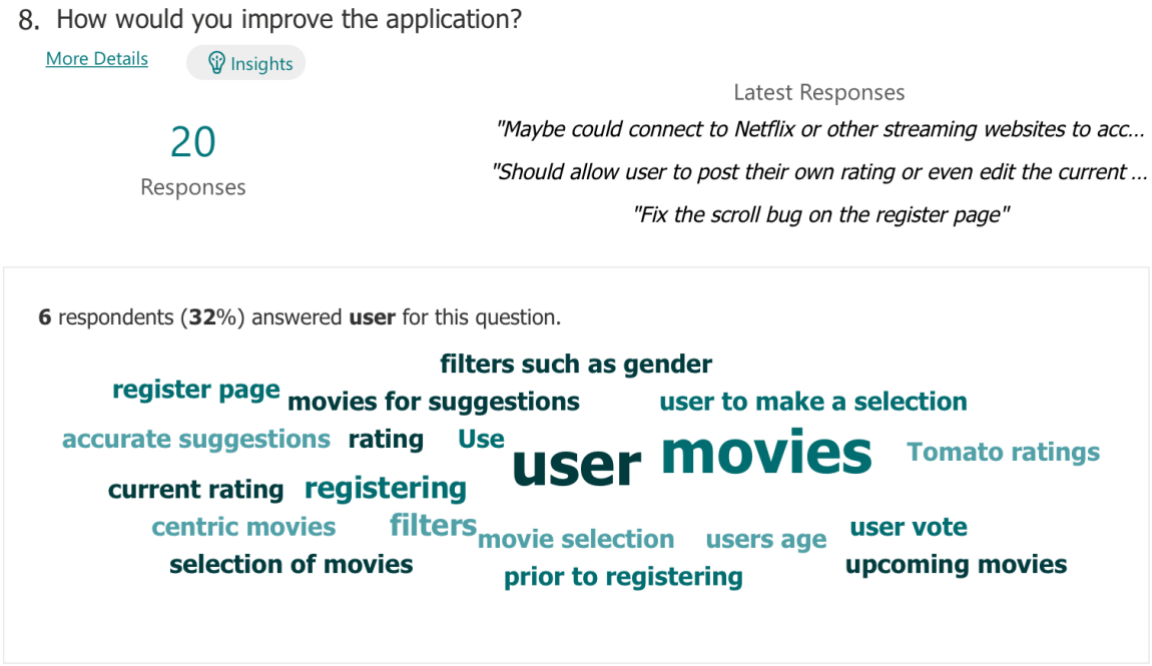


Figure 44- Illustrating users' suggestions on improvements to the application

The last question was, "After using this product, is this something you would recommend to a friend or a colleague?" as seen by the figure 45. This was asked to summarise whether the users found that the application was valuable enough to share with other users. The respondents answered 100% that they would. This infers that the overall product is helpful to consumers and would provide value to their friends and colleagues.

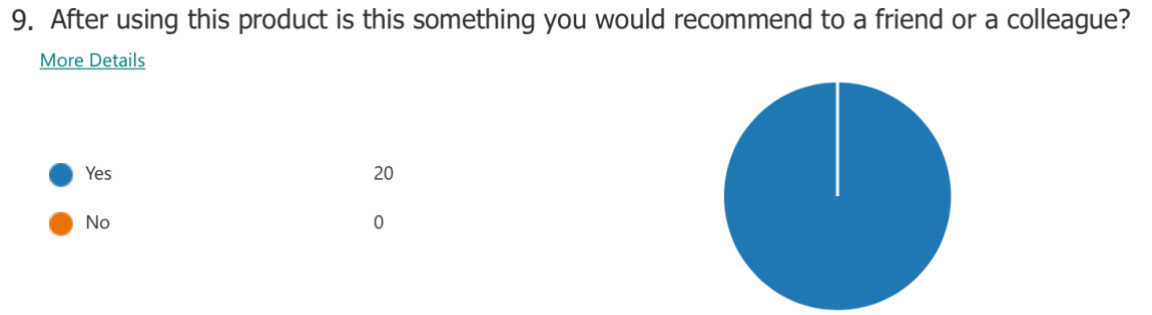


Figure 45- Illustrating whether the user would recommend this application after using it

4.3 Summary of Evaluation by questionnaire

Overall, the questionnaire has provided some valuable insight into how users perceived the application. From these results, recommendations were taken into consideration for the next iteration of the application and used to document the particular features that the users appreciated the most.

4.4 Design Decisions that changed and limitations

There were a few changes the author had to make regarding design decisions that had to be altered. This was because of the time constraint and lack of expertise in particular technologies. One example of design changes the author had to make was using TMDB API to access the movie database to filter the movies by genre and year. Initially, the author had decided to use IMDB API as this was the rating system used to filter movies rated seven or above in the movie view holder. Nevertheless, this was proven to be too difficult to use. Hence, the author decided to use the TMDB API as it was more straightforward and only required an API key.

Another aspect of the design that was supposed to be implemented was a sign out button. However, this was not implemented again as the remaining time was spent on documentation and testing the product. As all the other functions were operating smoothly, the author focused on areas he thought needed more time.

The author initially also wanted to incorporate the user's emotional state before selecting the movie. This was intended to be done in two different methods. The first method would capture the current user's state, and depend on that current state, a specific movie genre or movie suggestion would be suggested to the user. So, for example, if the user were feeling happy (indicated by the colour choice they choose, which they feel represents their current mood), they would be suggested a comedy or light-hearted movie. The second method would be just a question to inquire what kind of mood they feel in or what they would like to feel from a movie. The author is convinced this would give a more accurate representation and work favouring the user more.

Alongside emotional state, the author aspired to include age as a filter that helps tailor movies to individuals. The author believed certain age groups are more likely to watch similar movies of either the same year or just popular with certain crowds. However, this was not accounted for as it was somewhat challenging to group certain movies into ones that certain age groups would like. Moreover, as the demographic was 18-34, the groups of movies would be pretty similar across this age. If the author did decide to implement this, it would be complex, given that the author believes some ages have such variation in taste, and one age group may not share the same tastes at all. Hence, making the approach less tailored.

Similarly, to age as a filter, the author also intended to filter by gender. The author believed that genders might vary in taste as male viewers are more likely to be inclined to watch a movie with some form of action than they are to watch a romantic comedy. Although this may be too reductionist as male viewers may also watch romantic comedies, they may either enjoy it or watch it with their partners. However, the author assumed that this might run the risk of inclusion given that some members of the public do not identify with just two genders. Thus, the author decided not to include this to ensure greater inclusivity.

The author intended to fit the application for various android screen sizes to ensure a diverse audience could use the application. Despite this, the author decided to use the Google Nexus 5X emulator as this was considered good support for most applications and was simple to use. However, the author did not achieve the desired outcome of device compatibility over multiple screen sizes as this was not thoroughly tested and as the application was not published to the play store, this would also reduce the likelihood of it adapting to different screen sizes.

As previously mentioned, the author anticipated publishing the application to allow a more significant number of users to access the app. Hence more user reviews could have been conducted. Retrospectively, the author realised this would not have been the wisest decision given that the app still needs further changes and needs to evolve with features that had not yet been implemented. The author intends on continuing working on the application as a personal project to add to his portfolio of work to show to employers.

The author intended to include TV shows and more genres, although this was not the primary objective but an extension of the project. The authors reasoning behind adding more genres such as "New Releases" or filters such as "Watch with Family" like in the application Taste. He felt this would allow longer user retention, just filtering by genre and year was not enough to captivate the user with the most tailored suggestions.

Upon examination of existing applications, the author did appreciate the importance of creating some form of accuracy meter which builds a profile of the user and saves the movie suggestions and searches. This meter would have been an extension of the app's attributes, given that this would have been difficult for the author as he is not experienced with machine learning to build models of each user.

An additional feature that was initially not implemented was on the movie view holder page was that the top bar title did not initially pull the details from the user's selection to show the genre and the year of the movie selected. This was advised by the supervisor to be implemented. Initially, the author was reluctant to implement this feature as he was unsure how to create this. Nonetheless, after the author experimented with the code, he figured out how to implement this feature.

5.0 Conclusion

This project aimed to create an Android application that can help assist consumers in their movie selection process. The project aimed to create a service that would limit the time spent choosing a movie. Whilst tailoring these recommendations specifically for the user. The author illustrated this by accessing TMDB API and filtering through genre and year. In the author’s opinion, this was achieved through the application that was developed.

5.1 Meeting the original objectives

The aims and objectives stated in the dissertation in chapters 1.3 to 1.5 were essentially all reached. Furthermore, these objectives have been validated from the system, user feedback, and analysis of the results obtained through the testing process.

The initial objective of researching and analysing the existing applications that provide a similar service and analysing the features and drawbacks of these applications has been met by evaluating systems such as Taste and MovieLens and drawing out the key features that were common through both and any unique features that stood out. Thorough research was also conducted into the literature of recommendation systems to illustrate their importance and the broader scope as these systems are not only in the movie realm. In the background section of the dissertation, the importance of these systems is established. These systems are integrated into social media networks and eCommerce, to name a few. The section also covered how these systems operate and what techniques they use to improve their accuracy.

The second objective was to create a functioning Java android application that operates smoothly with no bugs. This was mainly achieved. The only criticism the author would point out was the issue with the movie_detail class. The issue with this was that the movies released in 2000 experienced a fault within the trailer link. Often, exporting the user to an unofficial trailer for the movie was not a huge issue, given it still provided a trailer for the movie selection. However, this could not be fixed as the trailer link was from the TMDB database.

The third objective was to solidify Java programming knowledge and build upon existing knowledge from previous programming modules. The author would say this was primarily met and was done through assistance from the Udemy Android Java Masterclass course, which taught the fundamentals. The android developer's documentation was also proven to be incredibly useful as a lot of the areas, such as creating a drop-down menu and creating a simple interface, were clearly instructed. Also, the firebase documentation on how to create a database was relatively straightforward.

The fourth objective was to create a system that uses IMDB API to filter movies of rating 7 or higher. Although this was later changed to TMDB, the author would still say this objective was met. Despite not using the initial database, the alternative proved to be much easier to implement, and it worked through filtering genre, unlike IMDB. The rating system was also achieved through TMDB.

The fifth objective to create additional filters such as genre and year of release was accomplished through the application. Although it was limited to the set genres and years on our system, this objective was met.

The sixth objective was based on recommendations on both movies that have been previously watched or current mood. Unfortunately, this objective was not met, given that the recommendations were based solely on genre and year.

The seventh objective was to develop an understanding of how to use Android Studio and all the functions. This objective was met using the well-documented libraries and using the Udemy online course on Java.

The eighth objective was to allow consumers to create an account. This was met by using Google Firebase to store the users account details.

The ninth objective was to allow the recommendation of TV series as well as apps. Although this objective was not met, it was an extension instead of a primary requirement as it is a movie-based recommendation system.

The last objective was to evaluate and test the produced system. This was done through Microsoft surveys both before the application build and post-application build. In addition, a range of tests was conducted through the development process, such as unit tests.

5.2 Positive and negative aspects

The positive aspects of this project have superseded the negative ones. According to the author, it was a novel experience learning to study Android studios and catching a glimpse of what android developers must face on a day-to-day basis. Discovering how to start a project

from start to finish and writing about the process has given the author newfound respect for developers, particularly Android ones. The ease of use of the Firebase database makes the idea of working with this in the future more appealing, given that it is relatively new and still implementing new features.

It has been a pleasure to delve into the android sphere and learn how recommendation systems operate. The author is aware that these systems may benefit consumers and acutely aware that there are ethical implications of recommendation systems as they can have quite alarming implications regarding data privacy concerns.

One of the most frustrating aspects of the project was, of course, time management. As circumstances prevented the author from submitting on the original time frame, this only added pressure to perform given extra time was granted. Hence the quality of work should have reflected that. The author believes that also the nature of this year being online has affected the entire experience as Java was predominantly taught online. Of course, this has its limitations given that a lab experience, in the author's view, would have possibly given more insight in terms of depth and naturally created better opportunities to discuss with peers and staff on areas they felt needed clarification.

However, all in all, it has been a delight and to learn from esteemed academics, albeit online, but an opportunity that will help develop the author's career and navigate through the field of Computer Science.

5.3 Lessons Learned

- How to use APIs and specifically TMDB API and formatting it correctly to filter by genre and year
- The existing literature of how recommendation systems work more broadly, not just in the scope of movies
- A more comprehensive understanding of machine learning and the applications of this regarding recommendation systems
- A greater knowledge of Java and exposure to how to use Android studios
- Being able to time manage an actual project and the issue faced during this process

5.4 Future work

A few additional implementations could be added to future work as some limitations of the system were not included because of the time constraint. The features that were not integrated could be suggested as systems to improve consumer retention of the product. Features that were not implemented in this system are the following:

- Social media integration, by this the author means allowing users to connect to their social media via the app, this could include sharing of recommendations to their friends, or even inviting them to download the application via social media, which would be suitable for scaling audiences
- Similarly, there could have been a community feature where users can communicate with other users who are also registered on the application. This could have been done through threads as there perhaps could have been a community thread where all users could post and interact. Alternatively, separate threads depending on the movie they

are watching or the movie genre and could post their thoughts on their evaluation of the movie.

- Currently, there are no machine learning methods incorporated in this application. One of the ways this could be used, as seen on other applications like Taste, is the registration details such as age and gender to assist in choosing movies for users. As age and gender can be factors that may affect choice in a movie and capturing the user's emotional state could be a feature that would be interesting to implement, given there is literature on how this could be implemented.
- The objective of connecting to the users previously watched history was not met, so this could be implemented in a future project iteration.
- Lastly, a unique feature seen on MovieLens was statistics for the user's choices in movies and recording those selected options. The author believes this is an exciting feature that should be integrated to learn more about the individual user and suggest a more fine-tuned recommendation instead of just filtering through genre and year.

6.0 References

[1]
Ofcom. 2020. *Lockdown leads to surge in TV screen time and streaming*. [online] Available at: <https://www.ofcom.org.uk/about-ofcom/latest/features-and-news/lockdown-leads-to-surge-in-tv-screen-time-and-streaming> [Accessed 17 June 2021].

[2]
Marr, B., 2021. *Artificial Intelligence Can Now Taste - Transforming Winemaking With Tastry*. [online] Forbes. Available at: <https://www.forbes.com/sites/bernardmarr/2021/06/28/artificial-intelligence-can-now-tastetransforming-winemaking-with-tastry/?sh=740274937da5> [Accessed 17 June 2021].

[3]
App Store. 2018. *WatchBuddy - Movie Suggestions*. [online] Available at: <https://apps.apple.com/us/app/watchbuddy-movie-suggestions/id1300407919> [Accessed 17 June 2021].

[4]
Maheshwari, R., 2017. *7 Best Movie and TV Series Recommendations Android Apps - TechWiser*. [online] TechWiser. Available at: <https://techwiser.com/movie-and-tv-series-recommendations-android-apps/> [Accessed 18 June 2021].

[5]
Stoll, J., 2021. *Subscriptions to streaming services in the U.S. by age 2019 | Statista*. [online] Statista. Available at: <https://www.statista.com/statistics/742452/media-streaming-services-penetration-rate-age/> [Accessed 18 June 2021].

References for Background

[6]
O'Dea, S., 2021. *Mobile OS market share 2021 | Statista*. [online] Statista. Available at: <https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/> [Accessed 19 June 2021].

[7]
J. Zhang, Y. Wang, Z. Yuan & Q. Jin 2020, *Personalized real-time movie recommendation system: Practical prototype and evaluation*.

[8]
Reddy, S., Nalluri, S., Kunisetti, S., Ashok, S. and Venkatesh, B., 2018. Content-Based Movie Recommendation System Using Genre Correlation. *Smart Intelligent Computing and Applications*, pp.391-397.

[9]
Vidiyala, R., 2020. *How to Build a Movie Recommendation System?*. [online] Medium. Available at: <https://towardsdatascience.com/how-to-build-a-movie-recommendation-system-67e321339109> [Accessed 20 June 2021].

[10]
A. Eyjolfsdottir, E., Tilak, G. and Li, N., 2008. [online] Citeseerx.ist.psu.edu. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.4954&rep=rep1&type=pdf> [Accessed 20 June 2021].

[11]
Thanh Ho, A., L. L. Menezes, I. and Tagmouti, Y., 2006. *E-MRS: Emotion-based Movie Recommender System*. [online] Ptidej.net. Available at: <http://www.ptidej.net/teaching/ift6251/fall06/article/Projet%20Ai%20-%20Ilusca%20-%20Yousra.doc.pdf> [Accessed 21 June 2021].

[12]
Medium. 2019. *How Netflix's Recommendation Engine Works?*. [online] Available at: <https://medium.com/@springboard_ind/how-netflixs-recommendation-engine-works-bd1ee381bf81> [Accessed 21 June 2021].

References for Existing Systems

[13]
Taste.io. n.d. *Taste - Movie recommendations and reviews from likeminded people.*. [online] Available at: <https://www.taste.io> [Accessed 22 June 2021].

[14]
Movielens.org. n.d. *MovieLens*. [online] Available at: <https://movielens.org/explore/your-ratings> [Accessed 22 June 2021].

[15]
En.wikipedia.org. n.d. *Collaborative filtering - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Collaborative_filtering> [Accessed 23 June 2021].

Reference for Survey Questions

[16]
Wallach, O., 2021. *Which streaming service has the most subscriptions?*. [online] World Economic Forum. Available at: <https://www.weforum.org/agenda/2021/03/streaming-service-subscriptions-lockdown-demand-netflix-amazon-prime-spotify-disney-plus-apple-music-movie-tv>.

References for Building System

[17]
Lotz, M., 2018. *Waterfall vs. Agile: Which Methodology is Right for Your Project?*. [online] Segue Technologies. Available at: <https://www.seguetech.com/waterfall-vs-agile-methodology/> [Accessed 24 June 2021].

[18]
Nvisia.com. 2020. *What is Agile Methodology? Benefits of using Agile | nvisia*. [online] Available at: <https://www.nvisia.com/insights/agile-methodology> [Accessed 26 June 2021].

[19]
QRA Corp. n.d. *Functional vs Non-Functional Requirements: The Definitive Guide*. [online] Available at: <https://qracorp.com/functional-vs-non-functional-requirements/> [Accessed 27 June 2021].

[20]
Smith, R., 2021. *Top 10 Best Cross-Platform App Development Frameworks in 2021*. [online] Codename One. Available at: <https://www.codenameone.com/blog/top-10-best-cross-platform-app-development-frameworks-in-2021.html> [Accessed 28 June 2021].

[21]
Ibm.com. n.d. *IBM Docs*. [online] Available at: <https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-include-relationships> [Accessed 30 June 2021].

[22]
Babich, N., 2019. *The 4 Golden Rules of UI Design | Adobe XD Ideas*. [online] Ideas. Available at: <https://xd.adobe.com/ideas/process/ui-design/4-golden-rules-ui-design/> [Accessed 30 June 2021].

[23]
Flaticon. 2021. *Video Player free vector icons designed by Prosymbols*. [online] Available at: <https://www.flaticon.com/free-icon/video-player_777242> [Accessed 2 August 2021].

Reference for Evaluation of Jakob

[24]
Nielsen, J., 1994. *10 Usability Heuristics for User Interface Design*. [online] Nielsen Norman Group. Available at: <https://www.nngroup.com/articles/ten-usability-heuristics/> [Accessed 4 August 2021].

[25]
Bailey, R. W., Allan, R. W., & Raiello, P. (1992). Usability Testing vs. Heuristic Evaluation: A Head-to-Head Comparison. Proceedings of the Human Factors Society Annual Meeting, 36(4), 409–413. https://doi.org/10.1177/154193129203600431

[26]
Tatari, K. (2021). Appropriate Web Usability Evaluation Method during Product Development. ResearchGate. Retrieved 7 May 2021, from https://www.researchgate.net/publication/242525773_Appropriate_Web_Usability_Evaluation_Method_during_Product_Development.

Reference for Implementation

[27]
Android Developers. n.d. *App Manifest Overview | Android Developers*. [online] Available at: <https://developer.android.com/guide/topics/manifest/manifest-intro> [Accessed 9 August 2021].

[28]
Firebase. 2021. *Read and Write Data on Android | Firebase Realtime Database*. [online] Available at: <https://firebase.google.com/docs/database/android/read-and-write> [Accessed 11 August 2021].

[29]
Android Developers. 2021. *Activity | Android Developers*. [online] Available at: <https://developer.android.com/reference/android/app/Activity> [Accessed 11 August 2021].

[30]
Android Developers. n.d. *Spinners | Android Developers*. [online] Available at: <https://developer.android.com/guide/topics/ui/controls/spinner> [Accessed 15 August 2021].

[31]
Developers.themoviedb.org. n.d. *API Docs*. [online] Available at: <https://developers.themoviedb.org/3/getting-started/introduction> [Accessed 15 August 2021].

[32]
Android Developers. 2021. *AsyncTask | Android Developers*. [online] Available at: <https://developer.android.com/reference/android/os/AsyncTask> [Accessed 16 August 2021].

[33]
Android Developers. n.d. *HttpURLConnection | Android Developers*. [online] Available at: <https://developer.android.com/reference/java/net/HttpURLConnection>.

[34]
ScientiaMobile. 2017. *How to Correctly Form User-Agents for Mobile Apps | ScientiaMobile*. [online] Available at: <https://www.scientiamobile.com/correctly-form-user-agents-for-mobile-apps/>.

[35]
Android Developers. n.d. *Configure your build | Android Developers*. [online] Available at: <https://developer.android.com/studio/build>.

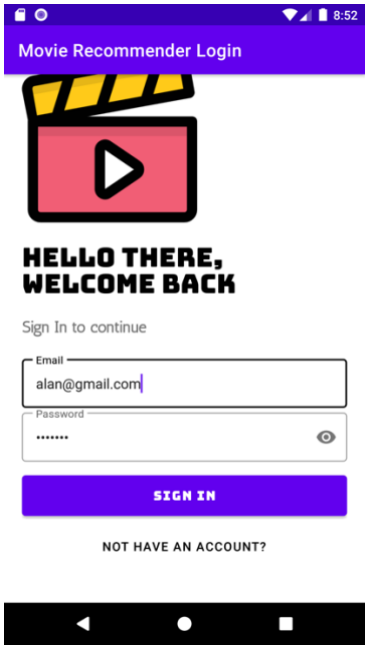
[36]
Android Developers. n.d. *Build local unit tests | Android Developers*. [online] Available at: <https://developer.android.com/training/testing/unit-testing/local-unit-tests>.

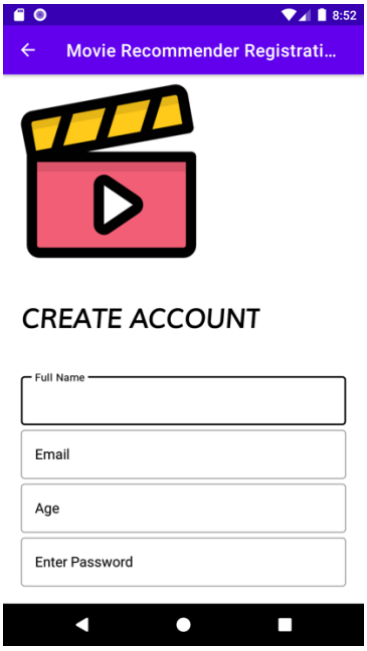
7.0 Appendix

A: User Manual

Step 1:

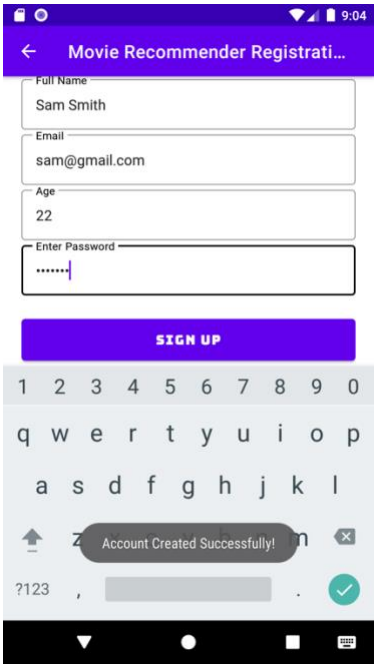
Register an account by scrolling down and by clicking “Not have an account”.





Step 2:

Once on registration page, enter user Account details and scroll to the bottom to click sign up. Once confirmation button says “Successful register” continue steps. If an “Unsuccessful register”, please follow the prompts and try again.

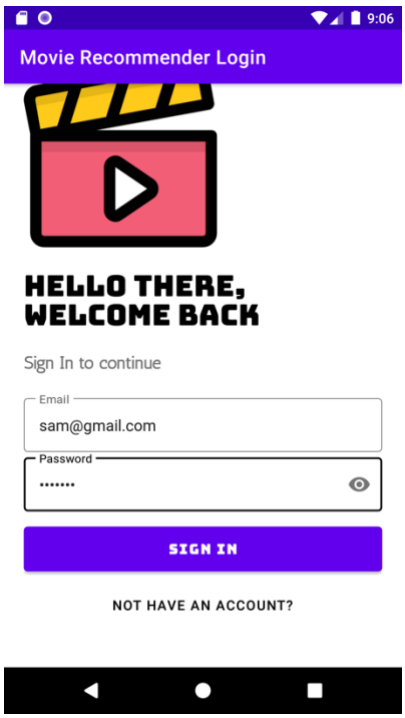


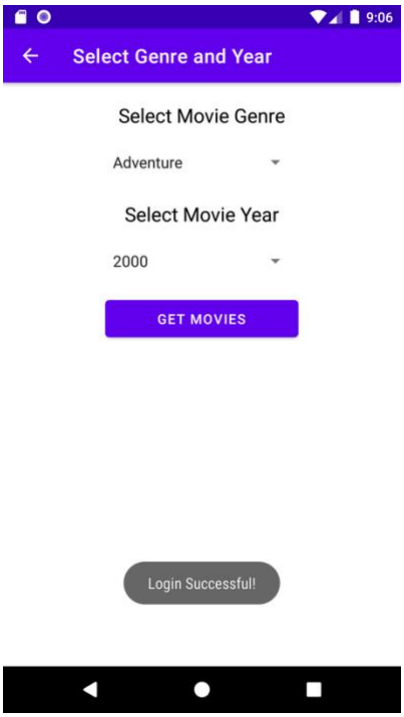
Step 3:

Click the back button on the top left of the screen and go back to sign in page.

Step 4:

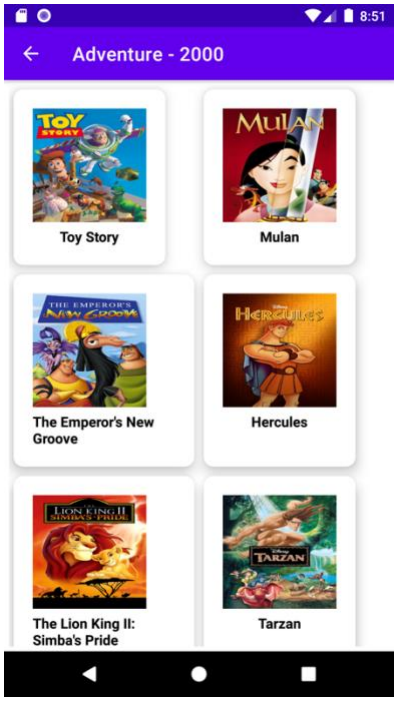
Enter login details on sign page and press sign in. If login unsuccessful follow prompts and try again.





Step 5:

Once logged in, select genre and year by using drop down menu and click “Get Movies”. This will display a movie view holder with a selection of movies for the given genre and year.



Step 6:

Click on desired movie choice and scroll down for more information and if you want to access trailer click “Show movie Trailer”.

Step 7

To change selection, use the apps back arrows or alternatively to close application press back on the emulator button or by turning off emulator.

B: Time Management plan

Week Number	Dates	Work to be done
1	17/06/21 - 25/06/21	Research existing mobile apps and movie recommendation literature
2	28/06/21 – 5/07/21	Research what technologies are used in mobile applications and planning and initial implementation ideas.
3,4,5,6,7,8	5/07/21 – 9/08/21	Software implementation: • Gathering requirements • System specification • Design of system

		• Coding
9	10/08/21 – 16/08/21	Testing and evaluating the software
10,11,12	16/08/21 – 13/09/21	Documentation
13	13/09/21 – 19/09/21	Proof reading

C: Pre-Scenario Questionnaire

1. Would you consider yourself technically savvy?
2. What is your job title?
3. What is your Age?
4. Do you use Android or iPhone?
5. Do you think an app could help users find their desired movie?
6. What percentage of the users would want to use the movie recommendation app?