MovieSearcher Application Report

University:	Aston University
Module Code:	CS3800
Module Name:	Advanced Database Systems

Student Name:	Lewis King
Student Number:	159025495
Student Contact:	kingl@aston.ac.uk

This is an individual project.

Word Count: 3069*

*Excluding pre-introduction, post-conclusion, and figure/table content.

List of Contents

Relevant Links

GitHub Repository
https://github.com/lewisbenking/ADB

The Github repository contains the source code for the application, along with this report, and a README for running the application.

YouTube Video
https://www.youtube.com/watch?v=eMKMEHbq2Yc

The YouTube Video serves as evidence of the application running on a laptop with Windows 10.

Introduction

The task given was to design and implement a useful data-driven application that is reliant on big data, or open source data. The application created did not have to be fully-fledged software, but should query the underlying dataset to obtain useful information and present this to the user (Jiang, Xiaorui. 2019).

The following sections of this report contain: *Area of Analysis*, *Design* and *Implementation*, then the report finishes with presenting the *Results and Reflections* on the project.

Area of Analysis

Project Purpose

The aim of this project is to create an application that will display information about a movie, based on an input from the user. This input could be either typing a movie title, selecting a “random” movie or choosing one of the top rated movies in the dataset. Users can also search based on an actors name, and the most recent titles for that actor will be presented.

The application is being created to eventually provide a central hub for searching for their favourite movies, and will provide detailed information including rating comparisons from different websites. The information will be displayed in a clear layout and an understandable format.

The type of information needed for this project will be about different movies, and specific data about each movie. Potential data for each movie could include the title, director, cast, runtime, age rating, plot, year of release and review scores.

Intended Application

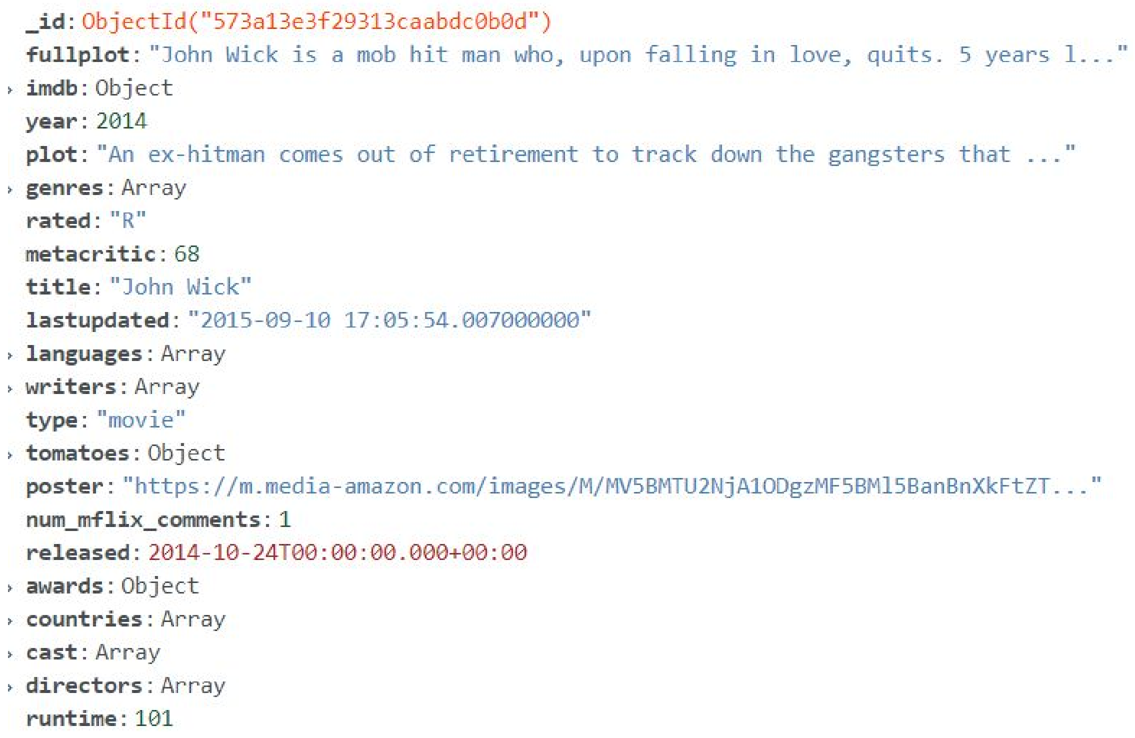
For this project, a C# windows form application will be created to provide a user interface for communication with the MongoDB document store. The application will be an executable file (.exe.), and is designed to run on Windows laptops. Different screens (forms) will be created for the different sections in the application, including the home page, viewing the information, and an about section. Information will be displayed in a clear and readable format to the user in order to provide a positive and seamless user experience.

The Dataset

The intended plan was to implement a dataset from IMDB, which contained over 2 million records, however when the dataset was downloaded, there were various data quality issues. For example, some entries only had a title of the movie, and no other fields, and at least 50% of the dataset contained data about YouTube videos and short episodes that weren't relevant to this project. Based on these discoveries, a decision was made not to use the IMDB dataset.

Fortunately when configuring MongoDB, which will be mentioned later, they provide sample datasets, including weather information and movies (MongoDB, 2019). As the project idea was to retrieve information about movies, it was decided that the MongoDB sample dataset would suffice. This dataset is much smaller than the IMDB dataset, however as a virtual cluster is being used for this project, a smaller dataset is more suitable, as recommended in the coursework description (Jiang, Xiaorui. 2019.) In addition to this, MongoDB limits the amount of storage on a free trial, meaning the IMDB dataset would have likely exceeded this free tier. When observing the dataset however, it became apparent that there was far more detail for individual items than in the IMDB dataset. The flexible schema of the MongoDB dataset means that attributes don't have a fixed order and are optional in each document. An example document on MongoDB can be seen below in *figure 1*.

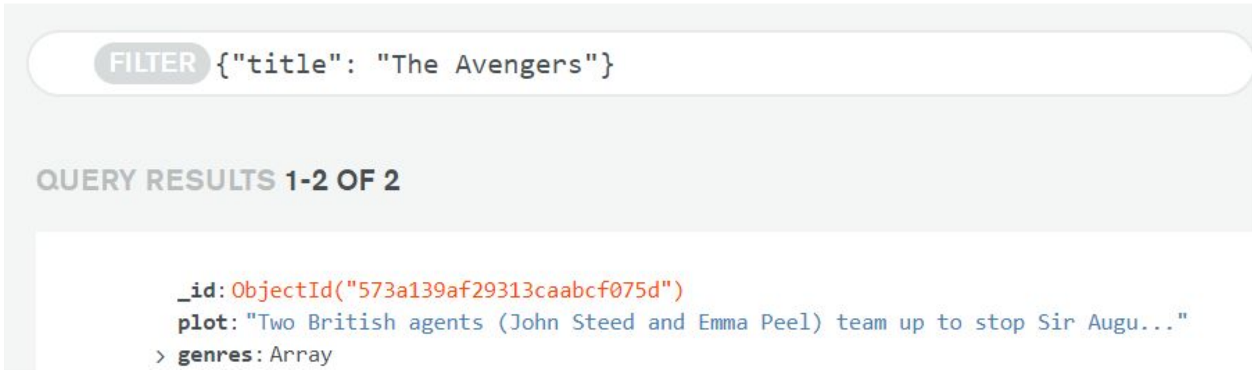
Figure 1: John Wick movie document on MongoDB document store.



```
  _id: ObjectId("573a13e3f29313caabdc0b0d")
  fullplot: "John Wick is a mob hit man who, upon falling in love, quits. 5 years l..."
› imdb: Object
  year: 2014
  plot: "An ex-hitman comes out of retirement to track down the gangsters that ..."
› genres: Array
  rated: "R"
  metacritic: 68
  title: "John Wick"
  lastupdated: "2015-09-10 17:05:54.007000000"
› languages: Array
› writers: Array
  type: "movie"
› tomatoes: Object
  poster: "https://m.media-amazon.com/images/M/MV5BMTU2NjA1ODgzMF5BMl5BanBnXkFtZT..."
  num_mflix_comments: 1
  released: 2014-10-24T00:00:00.000+00:00
› awards: Object
› countries: Array
› cast: Array
› directors: Array
  runtime: 101
```

From initially querying this dataset, as it has under 25,000 documents, it was clear that not every movie would be included. There were no outliers or anomalies in this dataset in contrast with the IMDB dataset. In addition to this, different movies may have the same title. An example of this was “The Avengers”. There are 2 movies in the dataset called “The Avengers”, as shown in *figure 2*. These films are not the same and are from different years, therefore in the application it is important to consider cases such as this, and inform the user of this. From querying the dataset, the most exact matches for a title was 3, for the film “A Star Is Born”.

Figure 2: 2 documents are returned when querying title: “The Avengers”.



```
FILTER {"title": "The Avengers"}

QUERY RESULTS 1-2 OF 2

  _id: ObjectId("573a139af29313caabcf075d")
  plot: "Two British agents (John Steed and Emma Peel) team up to stop Sir Augu..."
› genres: Array
```

Design

Requirements Analysis

This section consists of gathering user stories and converting them into functional and non-functional requirements.

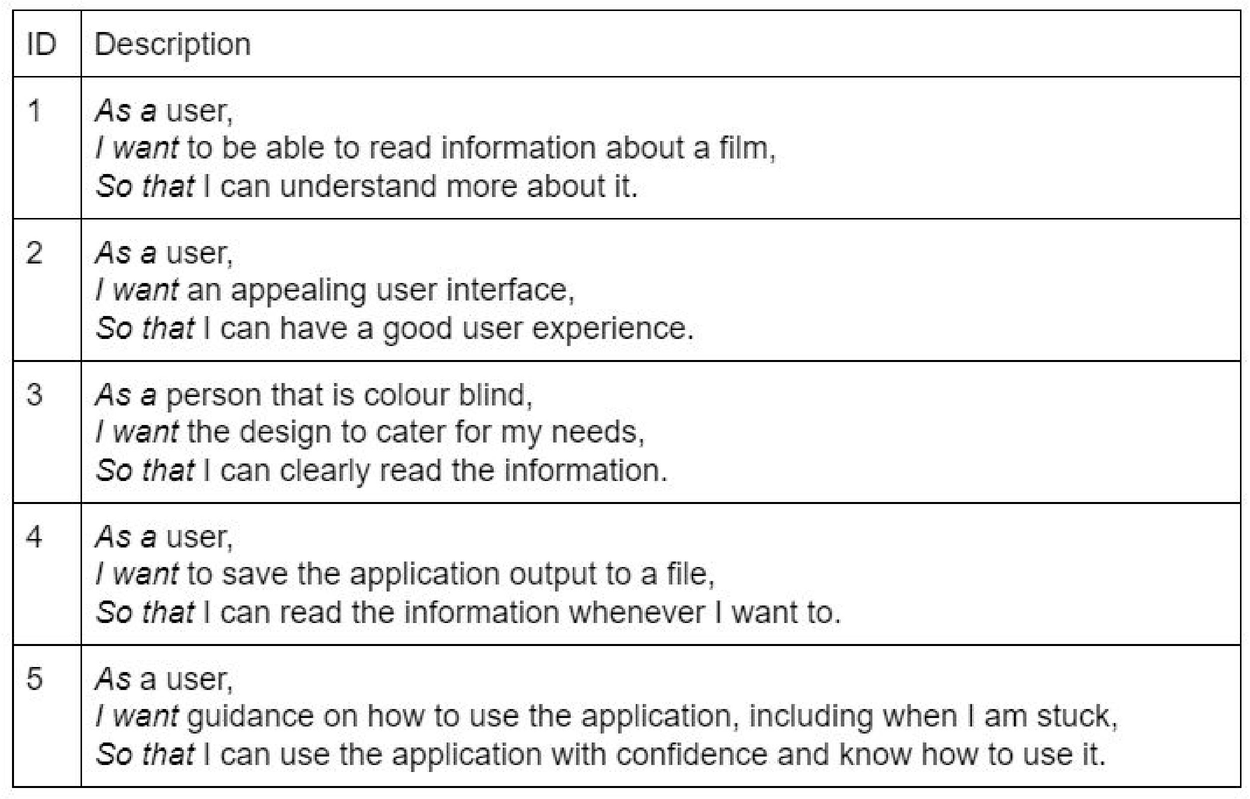
User Stories

Potential users were consulted to obtain their desired features from the software, and user stories were created, as shown in *table 1*.

The user stories followed a template of:

As a {user},
I want {feature},
So that {benefit}.

Table 1: User stories



ID	Description
1	*As a* user, *I want* to be able to read information about a film, *So that* I can understand more about it.
2	*As a* user, *I want* an appealing user interface, *So that* I can have a good user experience.
3	*As a* person that is colour blind, *I want* the design to cater for my needs, *So that* I can clearly read the information.
4	*As a* user, *I want* to save the application output to a file, *So that* I can read the information whenever I want to.
5	*As* a user, *I want* guidance on how to use the application, including when I am stuck, *So that* I can use the application with confidence and know how to use it.

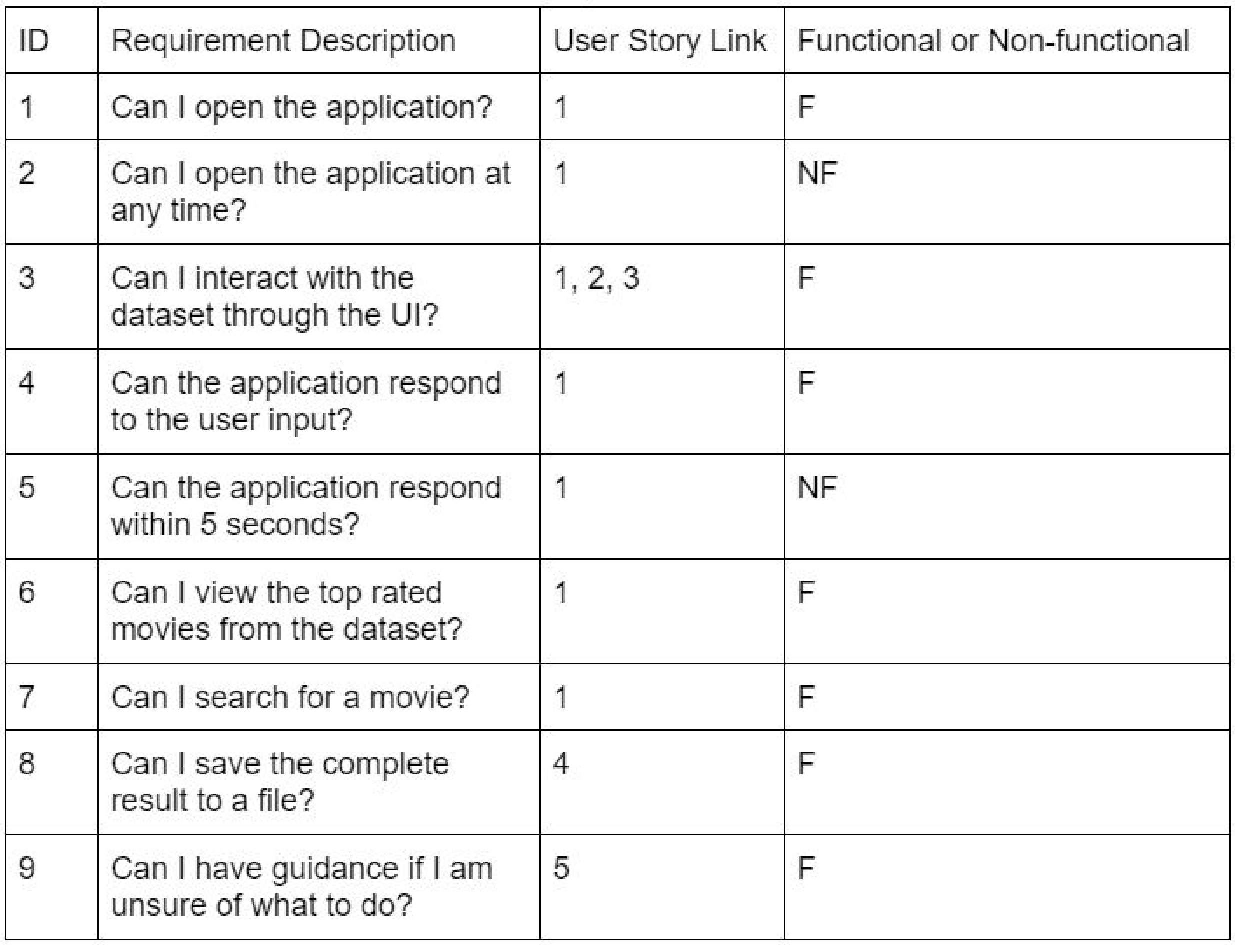
Requirements Specification

Next, user stories were converted into functional and non-functional requirements, as shown in *table 2*.

Functional Requirements describe what the program can perform, and what the user can do on the software. An example of this is: *“Can I log in”?*

However, non-functional requirements describe how well a program should behave. An example of this is: *“Can I log in within 5 seconds?”*

Table 2: Functional and non-functional requirements



ID	Requirement Description	User Story Link	Functional or Non-functional
1	Can I open the application?	1	F
2	Can I open the application at any time?	1	NF
3	Can I interact with the dataset through the UI?	1, 2, 3	F
4	Can the application respond to the user input?	1	F
5	Can the application respond within 5 seconds?	1	NF
6	Can I view the top rated movies from the dataset?	1	F
7	Can I search for a movie?	1	F
8	Can I save the complete result to a file?	4	F
9	Can I have guidance if I am unsure of what to do?	5	F

Summary

Based on the above requirements analysis, many users wanted information to be displayed clearly on the application and to cater to their needs such as colour blindness, with options to save the output to a text file if they wanted.

Database

MongoDB was chosen to store the dataset. MongoDB is an open source NoSQL database and document store. The dynamic schema allows easier representation of semi-structured data. Data is stored in a lightweight and fast Binary JSON format, has rapid scalability, and it supports ACID transactions from version 4.0 onwards. SQL Injection, which can occur in relational databases, is not an issue for the NoSQL MongoDB. MongoDB also supports many different programming languages and is designed to serve an OLTP workload (MongoDB, 2019).

Despite these advantages, there are some disadvantages to using MongoDB / NoSQL databases. Complex queries, including queries involving joins, are more suited to relational databases than NoSQL databases, and MongoDB needs more memory allocated. MongoDB is a fairly recent platform, meaning it is not extensively documented compared to SQL databases.

For this project and the types of queries required, MongoDB is a suitable choice, and connections to MongoDB from the application will be discussed under *Implementation*.

Wireframes

Wireframes were created to illustrate the intended design of the system. These wireframes do not contain specific images (the presence of images are shown with an X through a box), colours and fonts, but provide a general layout for the application and the software appearance and layout will have evolved from these wireframes. *Figures 3,4,5,6,7,8 and 9* show the wireframes for different sections in the software.

Figure 3: Wireframe for home screen.

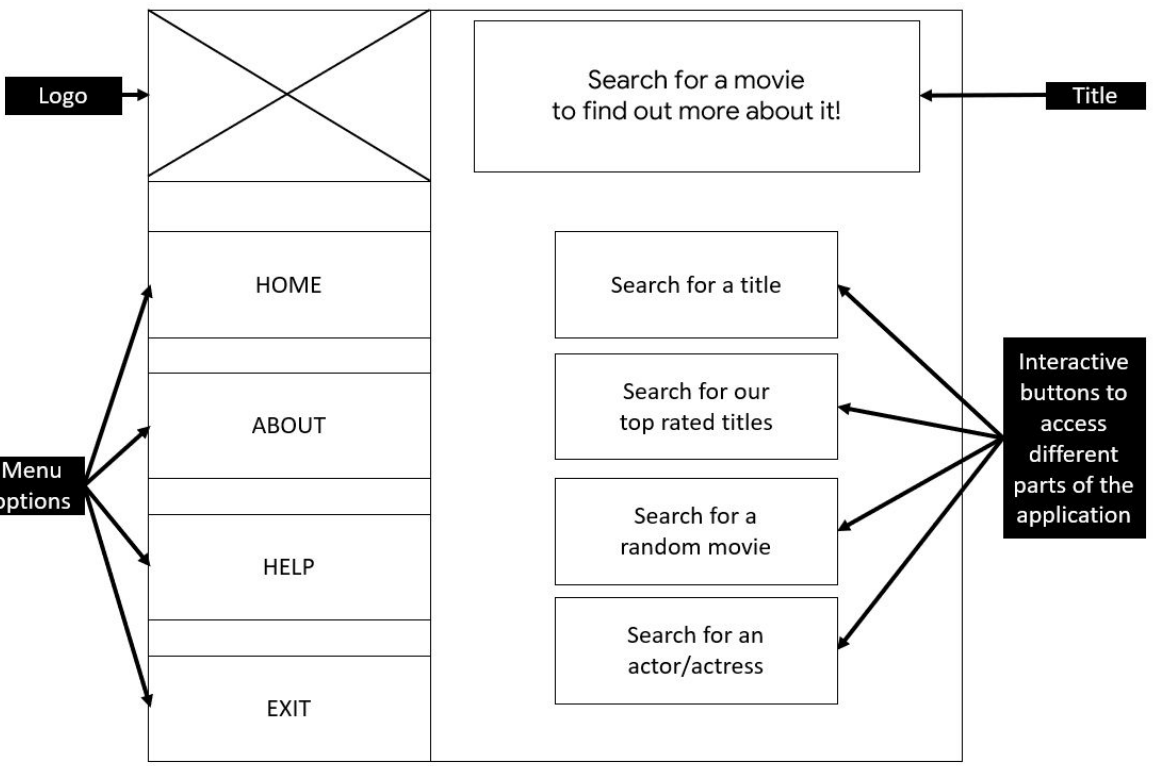

Logo
Search for a movie to find out more about it!
Title
HOME
ABOUT
HELP
EXIT
Menu options
Search for a title
Search for our top rated titles
Search for a random movie
Search for an actor/actress
Interactive buttons to access different parts of the application

Figure 4: Wireframe for "Search for a movie" screen.

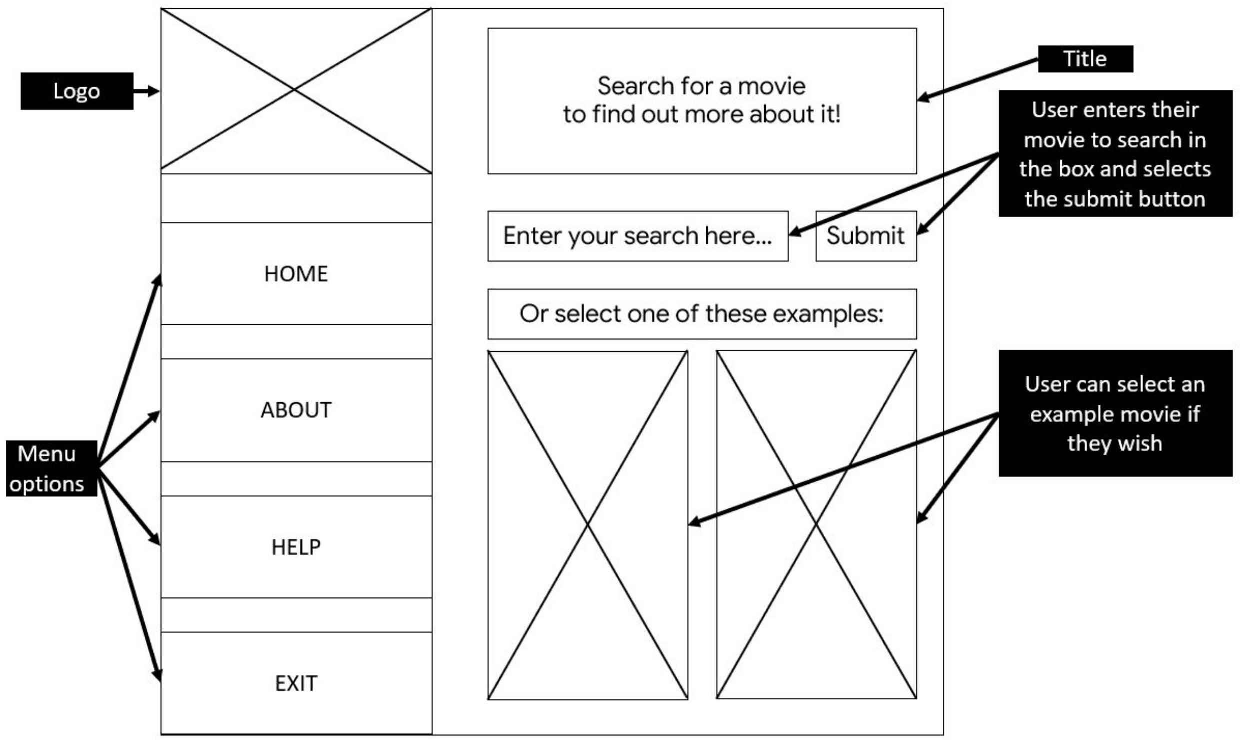
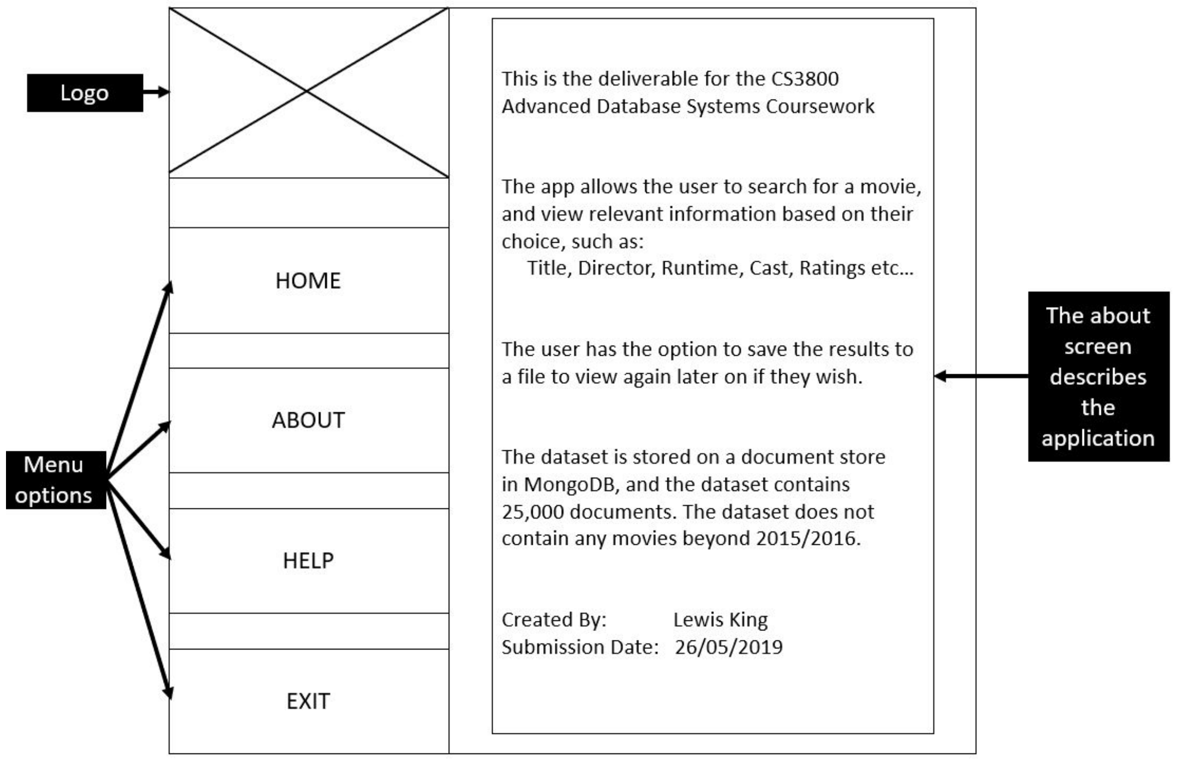

Logo
Search for a movie to find out more about it!
Title
User enters their movie to search in the box and selects the submit button
Enter your search here...
Submit
Or select one of these examples:
User can select an example movie if they wish
HOME
ABOUT
HELP
EXIT
Menu options

Figure 5: Wireframe for about screen.



This is the deliverable for the CS3800 Advanced Database Systems Coursework

The app allows the user to search for a movie, and view relevant information based on their choice, such as:
Title, Director, Runtime, Cast, Ratings etc...

The user has the option to save the results to a file to view again later on if they wish.

The dataset is stored on a document store in MongoDB, and the dataset contains 25,000 documents. The dataset does not contain any movies beyond 2015/2016.

Created By: Lewis King
Submission Date: 26/05/2019

Figure 6: Wireframe for "Top Rated Titles" screen.

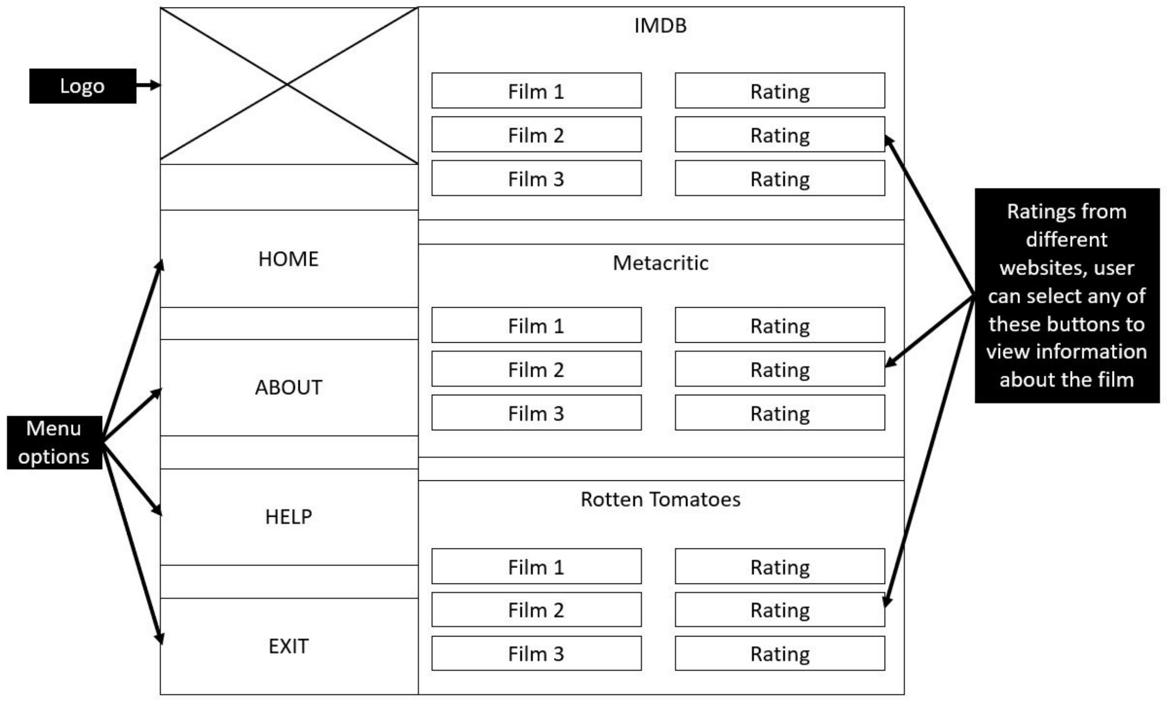
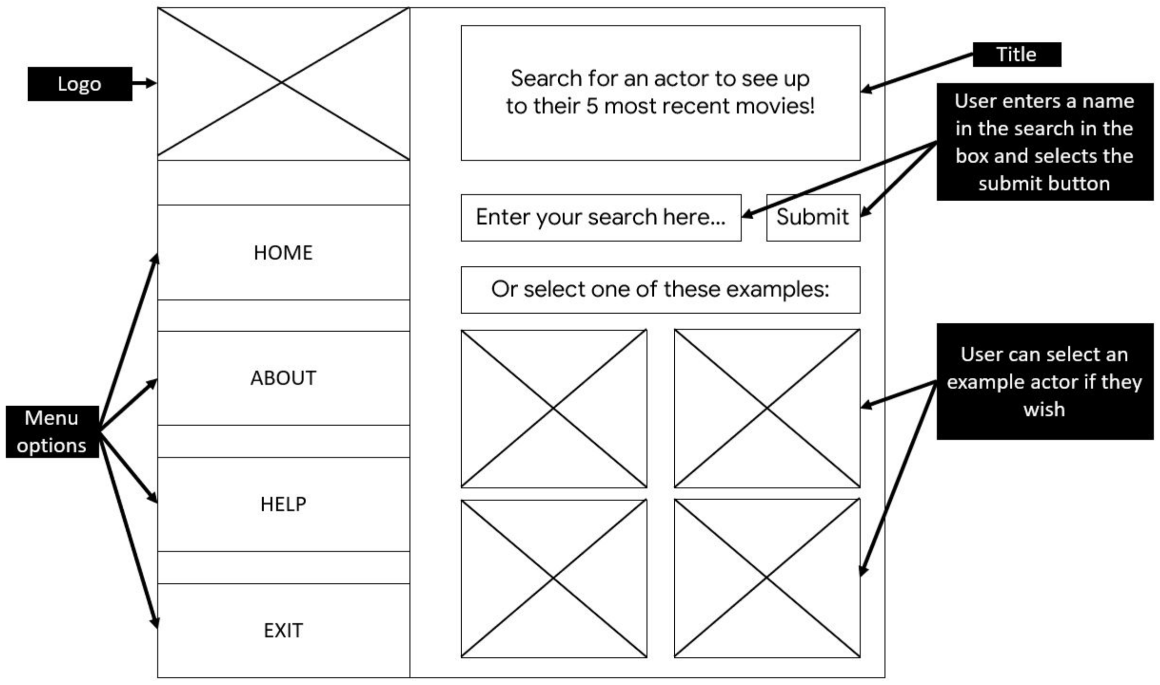


Figure 7: Wireframes for choosing an actor to search, and screen for when user searches by actor and displays up to their 5 most recent movies.


Logo
Search for an actor to see up to their 5 most recent movies!
Title
User enters a name in the search in the box and selects the submit button
Enter your search here...
Submit
HOME
Or select one of these examples:
ABOUT
User can select an example actor if they wish
Menu options
HELP
EXIT

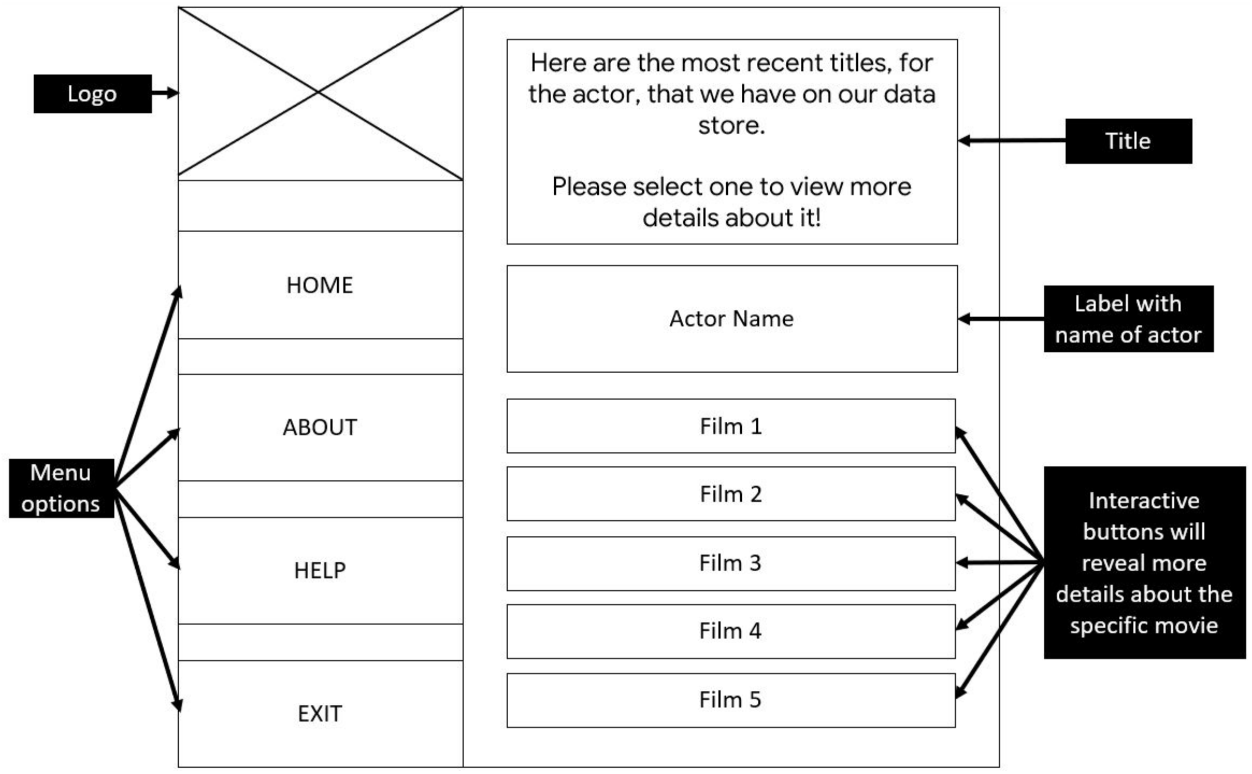

Logo
Here are the most recent titles, for the actor, that we have on our data store.
Please select one to view more details about it!
Title
HOME
Actor Name
Label with name of actor
ABOUT
Film 1
Film 2
Film 3
Film 4
Film 5
Menu options
HELP
Interactive buttons will reveal more details about the specific movie
EXIT

Figure 8: Wireframe for individual movie details screen.

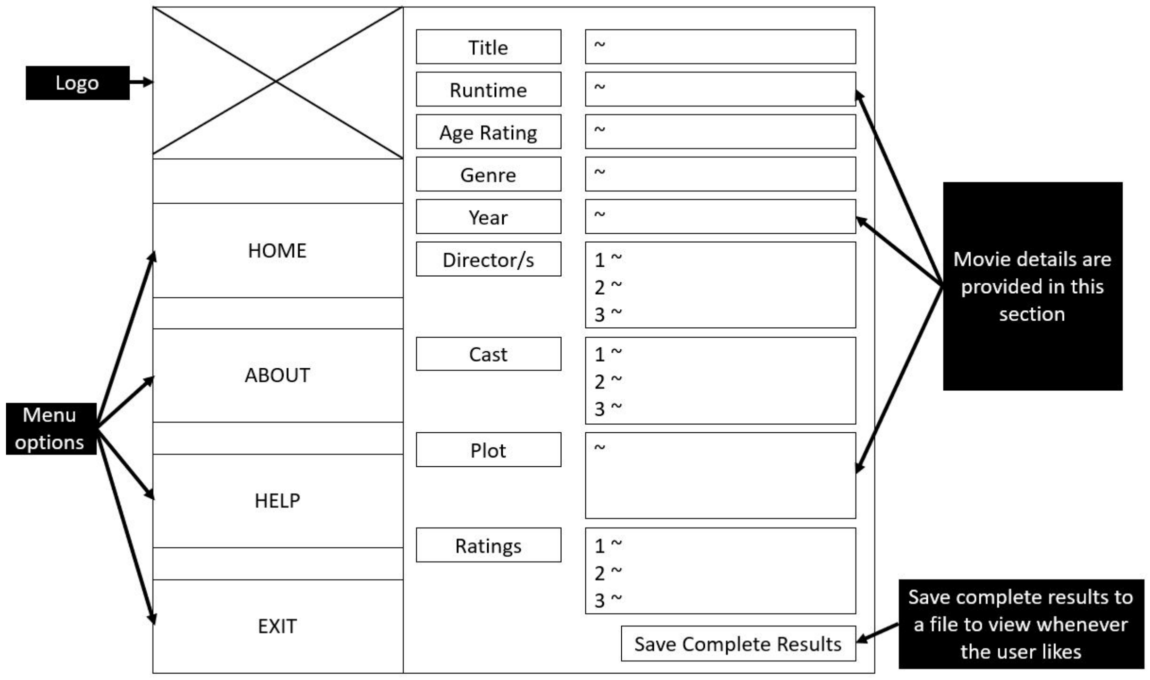
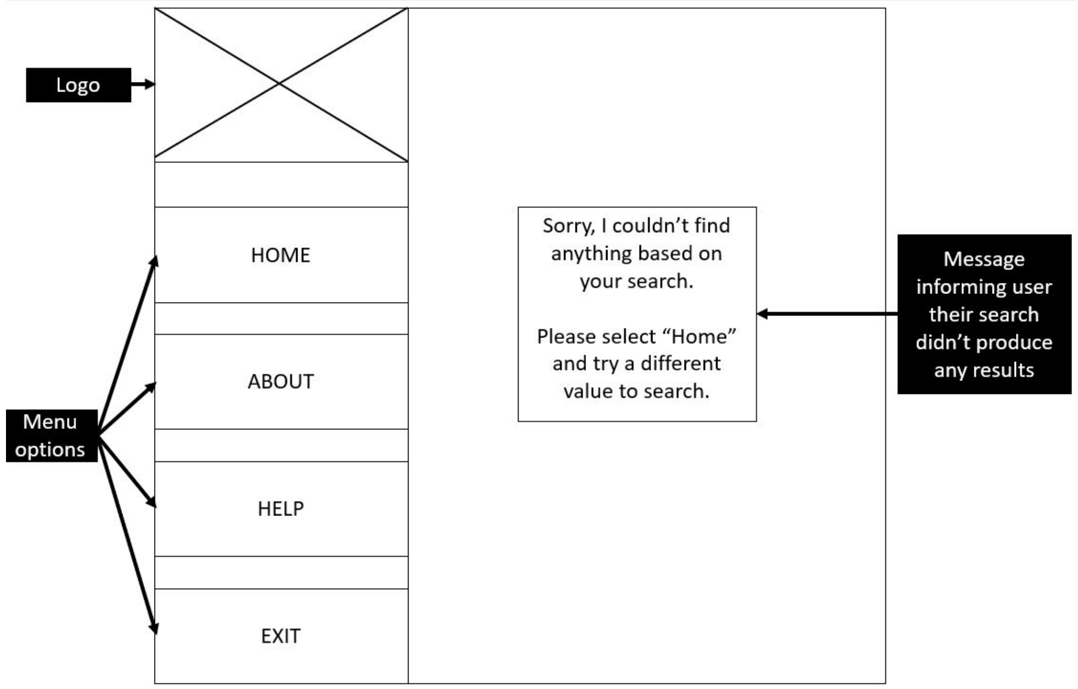


Figure 9: When the input doesn’t generate a result (e.g. title doesn’t exist in dataset).


Logo
HOME
ABOUT
HELP
EXIT
Menu options
Sorry, I couldn't find anything based on your search.
Please select "Home" and try a different value to search.
Message informing user their search didn't produce any results

Implementation

Brief Overview

The application is a Windows Form Application, and was made using Visual Studio 2017 (Community Edition), and in the C# programming language. The application contains 4 main sections, with each section containing varying numbers of subsections.

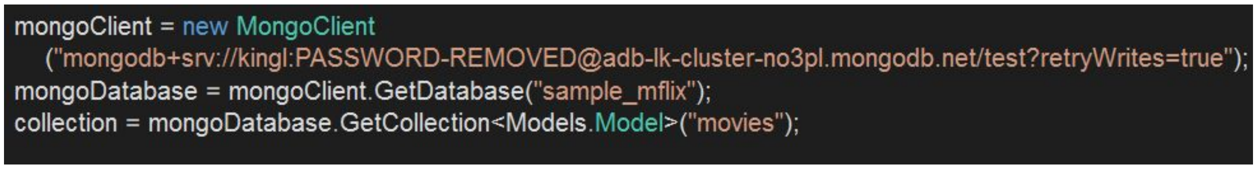
Connecting to MongoDB

To establish a connection to MongoDB, packages were installed from the NuGet package manager. These packages enable conversion to and from BSON types, and allow communication between .NET and MongoDB. The packages installed include:

MongoDB.Driver
MongoDB.Driver.Core
MongoDB.Bson

The connection was established by creating a MongoClient and then using the GetDatabase and GetCollection methods from MongoDB (MongoDB, 2019), as seen in *figure 10*.

Figure 10: Connecting to MongoDB



```
mongoClient = new MongoClient
    ("mongodb+srv://kingl:PASSWORD-REMOVED@adb-lk-cluster-no3pl.mongodb.net/test?retryWrites=true");
mongoDatabase = mongoClient.GetDatabase("sample_mflix");
collection = mongoDatabase.GetCollection<Models.Model>("movies");
```

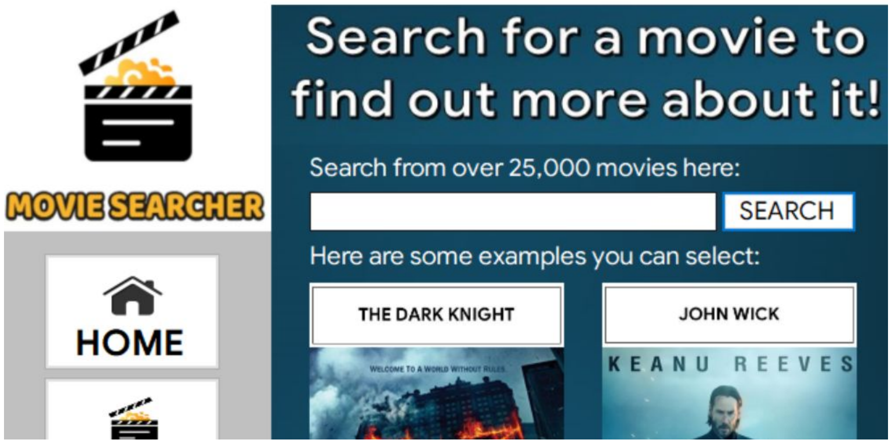
Now the connection to MongoDB is established, queries against the underlying data can be performed.

Functionality

Searching By Title

The first way of interacting with the dataset is to search based on a title. Figure 11 shows the screen the user will see when they choose to search by a title. They input the value into the textbox and select the search button.

Figure 11: Search by title screen.

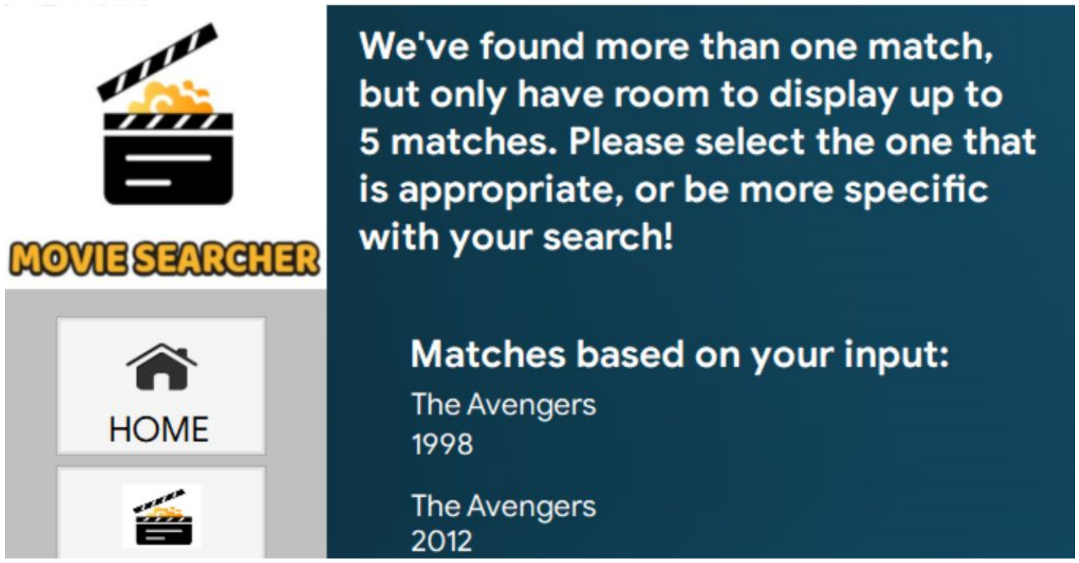

MOVIE SEARCHER
HOME
Search for a movie to find out more about it!
Search from over 25,000 movies here:
SEARCH
Here are some examples you can select:
THE DARK KNIGHT
JOHN WICK
KEANU REEVES

The application attempts to find an exact match for the phrase entered by the user, using the following code:

exactInputMatchModel = (from item in collection.AsQueryable()
where item.title == movieToSearch orderby item.title select item).Take(5);

The application can also handle an input from the user that generates more than one result, for example when searching by title for the phrase "The Avengers". As mentioned above, the dataset has 2 exact titles for this phrase so the application will return both results to the user and ask which one the user had intended to search, as shown in *figure 12*.

Figure 12: More than one match for the phrase "The Avengers".



We've found more than one match, but only have room to display up to 5 matches. Please select the one that is appropriate, or be more specific with your search!

Matches based on your input:

The Avengers
1998

The Avengers
2012

If there is no exact match for the user input, it will attempt to find a partial match (if the title contains the phrase), then if a match is now found, the application continues as normal. For this partial match, the following code is used:

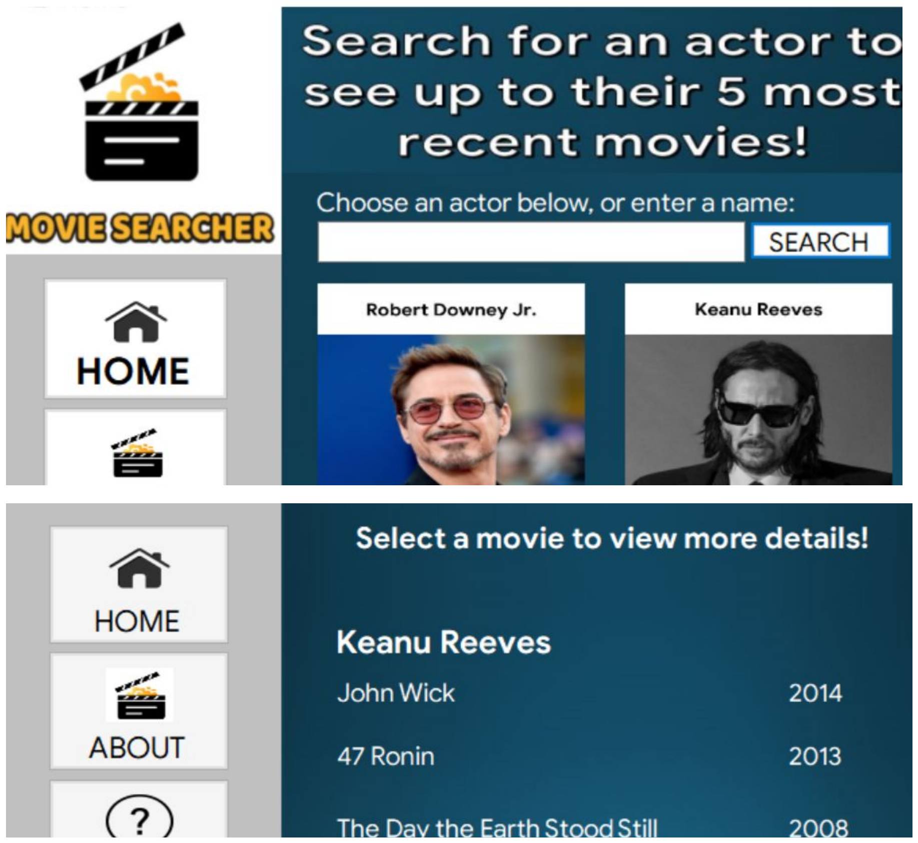
```
if (exactInputMatchModel.Count () == 0)
partialInputMatchModel = (from item in collection.AsQueryable()
where item.title.Contains(movieToSearch) orderby item.title select item).Take(5);
```

If there is still no match from the user input, the user will be informed of this.

Searching By Actors Name

Users can search by an actors name to view their most recent titles from the dataset (up to 5 due to the space available). There are some example actors presented to the user, including Keanu Reeves and Emma Watson, and users can select these if they want to. If a match is found for the user input, they will be able to view the recent titles for that actor, and this can be seen in *figure 13*. These titles are interactive, so the user can select them to view more details about the individual movie.

Figure 13: The different screens when searching for an actor.



Search for an actor to see up to their 5 most recent movies!

Choose an actor below, or enter a name:

Select a movie to view more details!

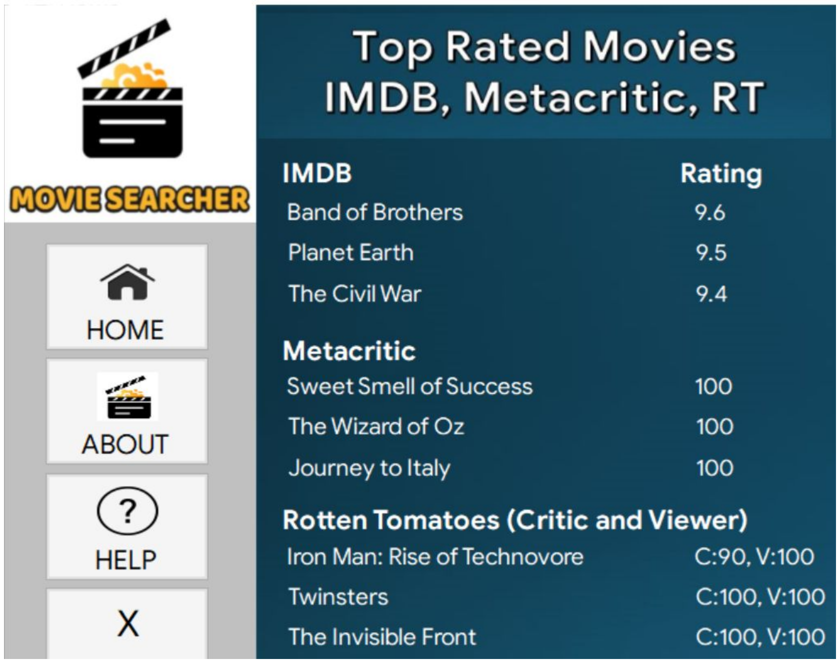
Keanu Reeves

John Wick	2014
47 Ronin	2013
The Day the Earth Stood Still	2008

Searching Top Rated Titles

Users can also view the highest rated titles in the dataset, from three different organisations (IMDB, Metacritic and Rotten Tomatoes). This feature is useful because the different organisations will rate titles differently, so users can compare titles and how the organisations have rated them. *Figure 14* shows the screen displayed to the user with each organisation having their own equal portion of the screen.

Figure 14: Top rated titles screen.



Top Rated Movies
IMDB, Metacritic, RT

IMDB	Rating
Band of Brothers	9.6
Planet Earth	9.5
The Civil War	9.4
Metacritic	
Sweet Smell of Success	100
The Wizard of Oz	100
Journey to Italy	100
Rotten Tomatoes (Critic and Viewer)	
Iron Man: Rise of Technovore	C:90, V:100
Twinsters	C:100, V:100
The Invisible Front	C:100, V:100

Searching a Random Title

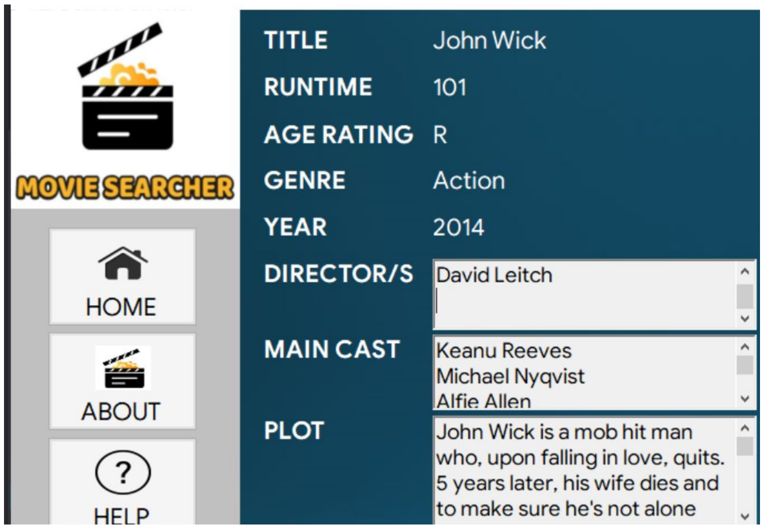
This option selects a sample document from the MongoDB collection, using the following code:

```
collection.AsQueryable().Sample(1);
```

When the document is retrieved, it populates the values on the results screen in the same way as searching for a title would do. With this feature, users may learn about movies they hadn't previously been aware of.

Figure 15 shows details for an individual movie (John Wick). This layout will be consistent regardless of how the user decides to search for a title, either by a name, through the top rated screen, a random title, or using an actor. The consistent layout will give the user a sense of familiarity.

Figure 15: Portion of individual movie details layout.



TITLE	John Wick
RUNTIME	101
AGE RATING	R
GENRE	Action
YEAR	2014
DIRECTOR/S	David Leitch
MAIN CAST	Keanu Reeves Michael Nyqvist Alfie Allen
PLOT	John Wick is a mob hit man who, upon falling in love, quits. 5 years later, his wife dies and to make sure he's not alone

Saving Complete Result

Due to the available screen space, it wasn't possible to display every field from the movie document from MongoDB. Users have the option to save the complete result to a text file and then they can view this at any time. *Appendix 1* shows the full document saved to a text file for the movie "John Wick". The code to save to a file is the following:

```
if (sfd.ShowDialog() == DialogResult.OK)
{
    using (StreamWriter writer = new StreamWriter(sfd.FileName))
    {
        writer.Write(movieInJson);
        writer.Close();
    }
}
```

Results and Reflections

This section will compare the outcomes with objectives, and reflect on the project by discussing limitations and future improvements.

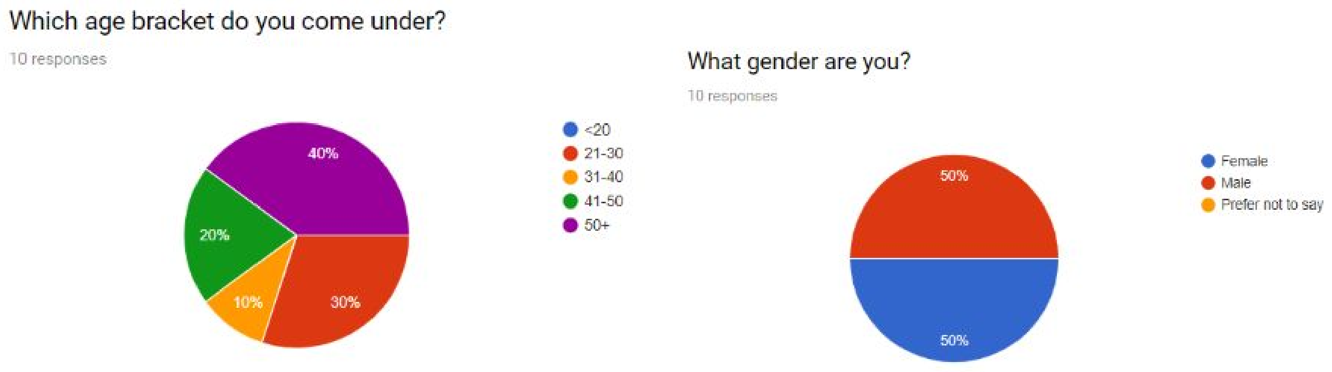
Comparing Outcomes With Objectives

For obtaining feedback through user acceptance testing, a questionnaire was handed out to the people that assisted in creating the user stories, in order for them to see the project progression and to provide unbiased feedback. The questions vary from usability and accessibility questions to their overall experience and effectiveness of the application in responding to their input.

The results gathered from this questionnaire can relate to the user stories defined above, so will be used in viewing how well the deliverable has met the objectives.

The first two questions are included to identify any patterns relating to age groups or genders. As shown in *figure 16*, the 10 testers are from different age groups and a 50/50 split of males and females, and this will provide a diverse set of results.

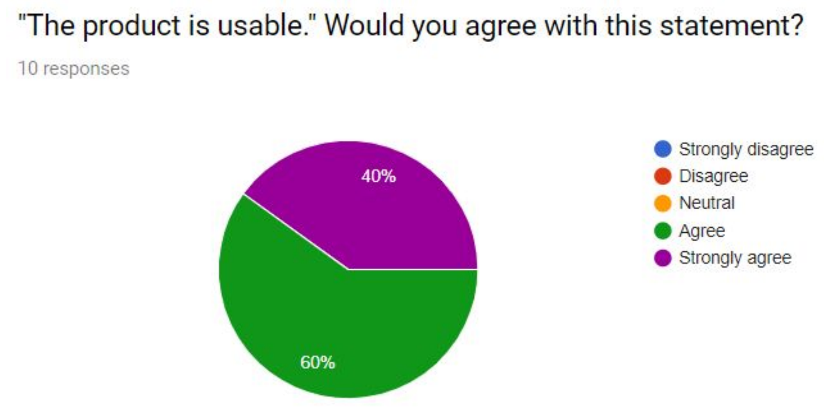
Figure 16: Question 1 & 2 responses.



Usability and Accessibility

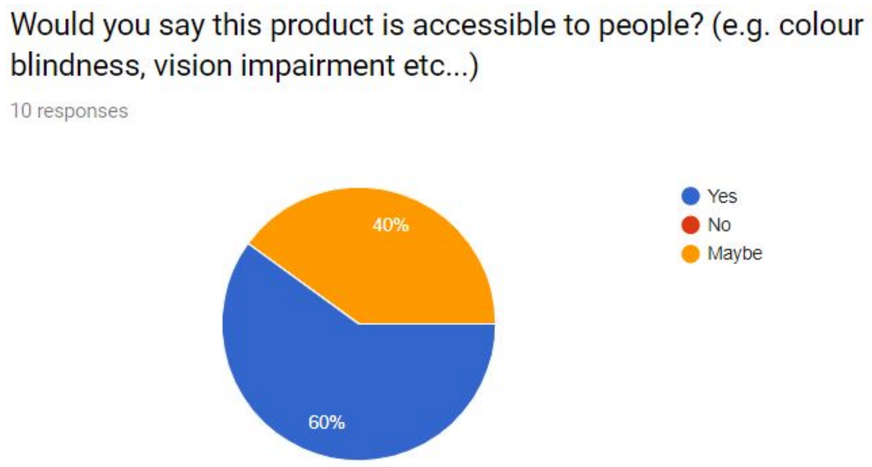
Figure 17 shows the responses to a question regarding usability. The results have shown that 100% of the testers either responded with “agree” or “strongly agree”, and this indicates the product is usable.

Figure 17: Question 3 responses.



Question 5, as shown in *figure 18*, asks for opinions on the product being accessible to people of varying abilities, e.g. colour blindness or visual impairments. The responses to this question were either “maybe” or “positive”, demonstrating the product can be accessible to people. This was expected because the colour scheme may not be suitable for everyone, however this colour scheme was chosen to make the UI more aesthetically pleasing to the user instead of using a plain white background and black text. Fortunately none of the testers responded with “no”. Unfortunately this application does not cater for people with severe visual impairments such as blindness, however a future iteration could incorporate audio output to solve this. Another possible reason for the “maybe” responses could be due to uncertainty of running the application on mobile devices (not laptops).

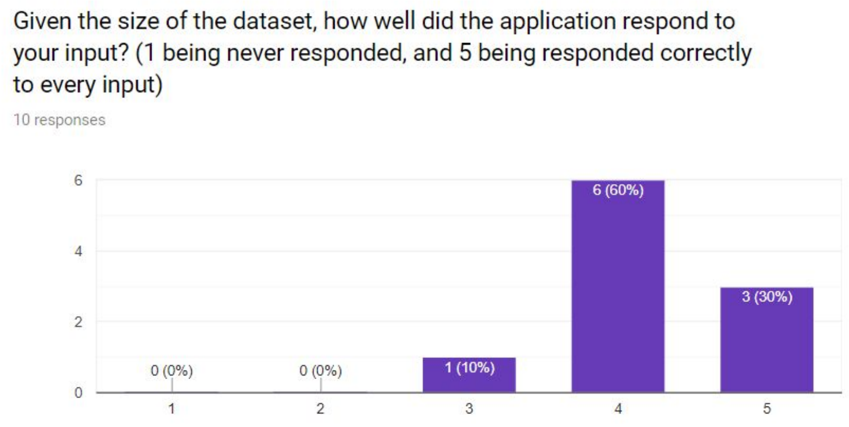
Figure 18: Question 5 responses.



Other

Question 6 asks for their view on how well the application responded to their input, this includes how quickly/slowly the response time was. Based on the results shown in *figure 19*, the users gave on average a 4/5 score, showing that the application responded promptly to their input, and responded as expected, i.e. the correct screen was shown based on their input.

Figure 19: Question 6 responses.



Questions 7 and 8, shown in *figures 20 and 21,* ask whether the users could envisage people using this software, or a future iteration. Given the current state, neutral or negative responses were expected, because there is certainly room for improvement, however it was encouraging to see the users believe people could use a future iteration of the software, given some modifications.

Figure 20: Question 7 responses.

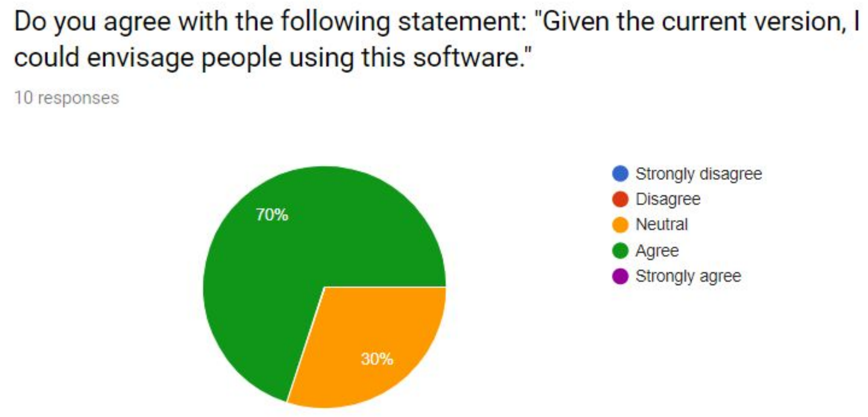
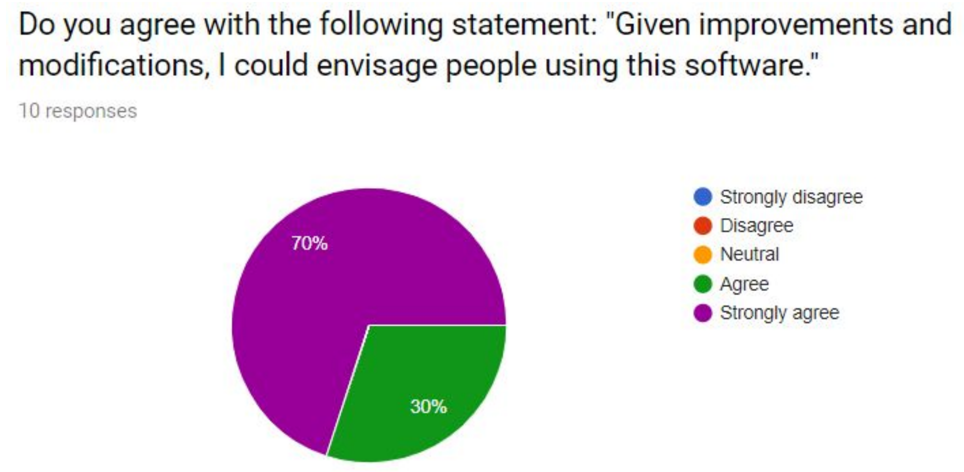
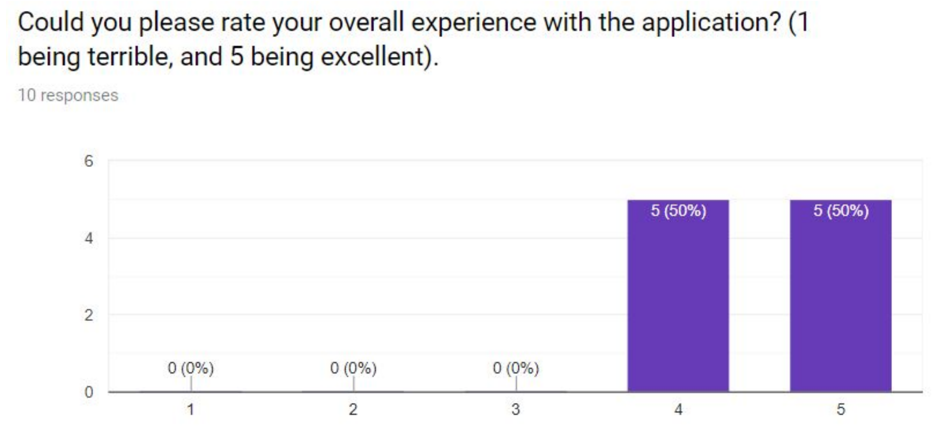


Figure 21: Question 8 responses.



The final question, shown in *figure 22*, asks for an overall rating based on their experience, and fortunately all of the responses were positive being 4/5 and 5/5, indicating the user experience is more than satisfactory.

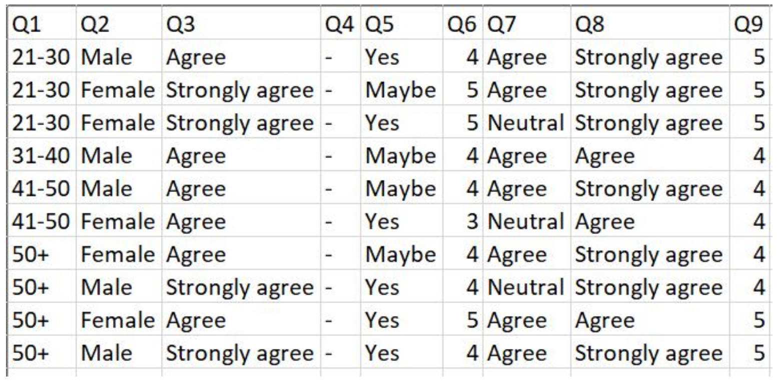
Figure 22: Question 9 responses.



Patterns Identified

From observing the raw response data extracted from Google Forms, as shown in *figure 23*, it was interesting to see the shared opinions in the age groups and difference of opinion between different age groups. Similar opinions within the age groups can be seen for Q9, with every person aged 21-30 responding with 5/5. Differing opinions between groups can be seen with Q6 with people under the age of 30 generally responding with 5/5 but older age groups responded more consistently with 3/5 or 4/5. This suggests different age groups can share differing opinions on the application.

Figure 23: Raw response data extracted from Google Forms.



Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
21-30	Male	Agree	-	Yes	4	Agree	Strongly agree	5
21-30	Female	Strongly agree	-	Maybe	5	Agree	Strongly agree	5
21-30	Female	Strongly agree	-	Yes	5	Neutral	Strongly agree	5
31-40	Male	Agree	-	Maybe	4	Agree	Agree	4
41-50	Male	Agree	-	Maybe	4	Agree	Strongly agree	4
41-50	Female	Agree	-	Yes	3	Neutral	Agree	4
50+	Female	Agree	-	Maybe	4	Agree	Strongly agree	4
50+	Male	Strongly agree	-	Yes	4	Neutral	Strongly agree	4
50+	Female	Agree	-	Yes	5	Agree	Agree	5
50+	Male	Strongly agree	-	Yes	4	Agree	Strongly agree	5

Reflection, Limitations and Future Improvements

This section will discuss the overall thoughts on the project, along with mentioning its merits, limitations and potential future improvements.

Dataset

The original IMDB dataset was abandoned due to issues mentioned previously, and whenever an attempt was made to fix the dataset, e.g. by removing unnecessary records, the computers in the Aston labs would crash. Reflecting on the IMDB dataset, given more time it could be eventually fixed and then imported into MongoDB to replace the existing data. Unfortunately, whilst the IMDB dataset contains far more up-to-date records than the current dataset, the individual records aren't as detailed. However, as the coursework specification stated, as a virtual cluster is being used, a smaller dataset would suffice for this first iteration. To expand and update the existing dataset, more time and resources would be required.

Application

Due to the dataset being stored externally online, the application requires an internet connection to interact with the dataset. The performance in terms of response times from MongoDB will vary depending on the quality of the internet connection. It was noted during testing that a higher quality connection means a reduced response time, however even with a poorer quality internet connection, the response times were still subsecond, showing the performance speeds were high.

The first iteration of the software has been solely designed for laptops that run the Windows operating system (ideally Windows 10). Upon reflection, users may prefer an application designed for their mobile devices, so this should be considered for a future iteration.

Regarding specific improvements to the application, when the user is searching for a title or actor, suggested (autocomplete) options could appear whilst they're typing for them to select if they wanted to. This feature could also save the user the time of searching for a value that doesn't exist in the dataset.

Another beneficial feature in a future iteration could be user profiles. The user profile could contain a list of their favourite movies/actors. When the user signs in successfully they could automatically view the list of their favourites. This feature could work alongside the existing "save to file" feature, as this will contain more data.

User Response

Upon reflection, the results from UAT could be used to prove that the software has met the aforementioned requirements, and the diverse test group could represent real future users of the application. The UAT results were generally positive, and it was interesting to identify patterns, using the raw data above in *figure 23*, for different genders and age groups. Users thought the application was usable and gave high scores for their overall experience, and that the application would respond to their every input. Despite the general positive outlook from UAT, it must be acknowledged that there is room for improving this software, and potential improvements for expansion have been identified above.

Conclusion

This report has covered the documentation of the development process of a data-driven application against an underlying dataset. It has discussed analysis, design, implementation and reflected on the results obtained against the project objectives.

References

These were explicitly mentioned in this report.

Jiang, Xiaorui., 2019. *Part I. Coursework Description*, Birmingham: N/A.

MongoDB, 2019. *FAQ*. [Online]
Available at:
https://www.mongodb.com/faq
[Accessed 01 05 2019].

MongoDB, 2019. *Sample Mflix Dataset.* [Online]
Available at:
https://docs.atlas.mongodb.com/sample-data/sample-mflix/#mflix-movies
[Accessed 01 05 2019].

MongoDB, 2019. *Welcome to the .NET MongoDB Driver Documentation.* [Online]
Available at:
http://mongodb.github.io/mongo-csharp-driver/2.8/
[Accessed 01 05 2019].

Bibliography

These were not explicitly mentioned in the report, but assisted in creating the software.

Hatchful, 2019. *Create stunning logos in seconds.* [Online]
Available at:
https://hatchful.shopify.com/
[Accessed 01 05 2019].

Kilroy, 2018. *101 Reasons Sandra Bullock is the Goddamn Best.* [Online]
Available at:
https://medium.com/@jakekilroy/101-reasons-sandra-bullock-is-the-goddamn-best-b221
44d55438
[Accessed 10 05 2019].

MongoDB, 2019. *Welcome to the .NET MongoDB Driver Documentation.* [Online]
Available at:
http://mongodb.github.io/mongo-csharp-driver/2.8/
[Accessed 01 05 2019].

Panganiban, 2017. *16 Encouraging Emma Watson Quotes.* [Online]
Available at:
http://mentalfloss.com/article/61464/16-encouraging-emma-watson-quotes
[Accessed 10 05 2019].

Pappademas, 2019. *The Legend of Keanu Reeves.* [Online]
Available at:
https://www.gq.com/story/the-legend-of-keanu-reeves/
[Accessed 10 05 2019].

Preston, 2019. *'Avengers: Endgame': How Robert Downey Jr.'s Biggest Line Almost Didn't Happen.* [Online]
Available at:
https://www.cheatsheet.com/entertainment/avengers-endgame-how-robert-downey-jr-s-biggest-line-almost-didnt-happen.html/
[Accessed 10 05 2019].

Appendices

Appendix 1: John Wick full results saved to a text file.

```
[{"_id":"573a13e3f29313caabdc0b0d",
"released":"2014-10-24T00:00:00Z",
"awards":{"wins":5,
"nominations":3,
"text":"5 wins & 3 nominations."},
"imdb":{"rating":7.2,
"votes":190907,
"id":0},
"tomatoes":{"critic":{"rating":6.9,
"numReviews":180,
"meter":85},
```

"viewer":{"rating":3.9,
"numReviews":71591,
"meter":80},
"fresh":153,
"rotten":27,
"lastUpdated":"2015-08-27T18:05:50Z"},
"runtime":101,
"num_mflix_comments":1,
"year":2014,
"metacritic":68,
"cast":["Keanu Reeves",
"Michael Nyqvist",
"Alfie Allen",
"Willem Dafoe"],
"countries":["USA"],
"directors":["Chad Stahelski",
"David Leitch"],
"genres":["Action",
"Thriller"],
"languages":["English",
"Russian",
"Hungarian"],
"writers":["Derek Kolstad"],
"lastupdated":"2015-09-10 17:05:54.007000000",
"plot":"An ex-hitman comes out of retirement to track down the gangsters that took everything from him.",
"fullplot":"John Wick is a mob hit man who, upon falling in love, quits. 5 years later, his wife dies and to make sure he's not alone she arranges for a dog to be brought to him after her death. Later, some men wanting his car break in and beat him up and kill his dog. When he recovers, he sets to get the ones who killed his dog. He learns that the leader is the son of his former employer. And the man wanting to protect his son, tries to take care of Wick but he's still as good as he was.",
"rated":"R",
"type":"movie",
"poster":"https://m.media-amazon.com/images/M/MV5BMTU2NjA1ODgzMF5BMl5BanBnXkFtZTgwMTM2MTI4MjE@._V1_SY1000_SX677_AL_.jpg",
"title":"John Wick"}]

Appendix 2: Tutorial / ReadMe

Pre-requisites

1) Laptop running Windows 10.
2) Visual Studio 2017 Community Edition (only for option 2 below).
3) Get the **entire** project source code files from either:
    a) The GitHub repository (link is near the start of the report).
    b) The .ZIP submission contains the files from the GitHub repository.

Installing the application

Option 1 - Run the .exe from the folder.

1) Either:
    a) Run the "Coursework - Shortcut" file in the ADB → Coursework → Coursework - Shortcut.lnk.
    b) Run the "Coursework" file in ADB → Coursework → Coursework → Bin → Debug → Coursework.exe.

Option 2 - Run the application in Visual Studio.

1) Open the project (.sln) in Visual Studio.
2) Go to Build → Build Solution, wait for it to finish.
3) Press Start button with green arrow to run the application.

Interact with the application

Interact with the application through a mouse to click buttons on the screen, and use the keyboard to type your input. Please note pressing the enter key will not act as a button click, so to interact with buttons you need to hover the mouse over the button and left click once on it. The application functionality is described in the report under *Implementation*. Try using the different features from the menu, i.e. search by title, search top rated titles, search random title and search by an actor. You can also try saving the results to a text file if you wish.