OCR GCE A

COMPUTER SCIENCE PROJECT

H446-03

Name : Aaryaa Awasthi

Candidate Number : <INSERT CANDIDATE NUMBER>

<Institution Name> : Reading Bluecoat School

Title of Project : Movie Matcher

H446-03 – Project CONTENTS

Table of Contents

[A. Analysis 3](#_Toc452555018)

1. **Problem Definition**
2. **Computational Methods to develop the solution**
3. Initial method on developing solution
4. **Stakeholders**
5. **Specifications**
6. **Possible Limitations**
7. **Product Research**

**a) Current solutions**

**b) Reviews on current solutions**

**c) Survey**

1. Project Requirements

a) Development Hardware Requirements

b) Development Software Requirements

1. Evaluation and success criteria
2. **Agile methodology**

[B. Design 3](#_Toc452555019)

1. Decompostion tree
2. Key data structures
3. Key algorithms needed
4. Usability features

[C. Developing the coded solution (“The development story”) 3](#_Toc452555021)

[D. Evaluation 3](#_Toc452555022)

[Project Appendixes 4](#_Toc452555023)

# A. Analysis

Problem Definition

With the vast number of films available on streaming platforms, users often struggle to choose what to watch next. Many rely on general rating systems or random browsing, which can lead to wasted time and unsatisfying viewing experiences. Current recommendation systems often lack personalization, and users don’t always trust anonymous or generic reviews. Additionally, it’s difficult for users to find reviewers with similar tastes, making it hard to identify films they are likely to enjoy.

A platform which can achieve these goals for the best user experience:

1. **Help users make informed decisions about what to watch next.**
2. **Provide personalized recommendations based on reviews from people with similar movie preferences**
3. **Allow reviewers to build credibility, so users can identify and follow people whose opinions they trust.**
4. **Save users time by reducing the trial-and-error process of finding enjoyable movies.**
5. **Will change film and media browsing forever**

This is where Movie Matcher comes in. Movie Matcher is a movie recommendation platform that helps users decide what to watch next by curating suggestions based on community reviews. Users rate and review films they have watched, and these reviews are publicly visible, allowing others to read and compare opinions before deciding to watch a film.

The program uses these reviews to recommend movies tailored to a user’s tastes by prioritizing feedback from users with similar preferences. Reviewers who consistently provide detailed and reliable opinions can gain credibility, building a following of users who trust their recommendations. By combining personalized suggestions with community-driven reviews, Movie Matcher saves users time by filtering out movies they are unlikely to enjoy and helps them discover hidden gems recommended by like-minded reviewers.

Computational Methods to develop this solution

Abstraction

I will remove unnecessary detail from my idea to make my program easier to code, and the problem easier to solve. Examples of the unnecessary detail including the images of the movies, input/output formatting etc. For the beginning stages of my development I will ignore the raw data storage details of the movies, such as whether they are stored I a list, database JSON, or API. My goal of the Movie Matcher is to match movies based on preferences such as genre, actors, producers, rating etc, so I can immediately abstract Internal IDs such as file paths or URLs, technical metadata and exact timestamps or storage format. The important details of my program includes: the users’ preferences (input), movie list (data model) and matching results (output).

Decomposition

My problem must be broken down into smaller, easier to solve, problems to guarantee my problem is easier to solve. Examples of the smaller problems include:

1. Data handling
   1. Loading movie data (from a file, database or API)
   2. Present a class/dictionary to represent the movies
   3. Store and Retrieve Movies
2. User Input Processing
   1. Create a user profile
   2. Gather user preferences (genre, producers etc.)
3. Movie Matching Logic
   1. Filter Movies (genre, ratings etc.)
   2. Calculate a similarity score, visualising how similar the movie is to the users preferences
   3. Sort/ Rank matches (best-to-worst order based off similarity rating)
4. Output and User interaction
   1. Display results aesthetically
   2. Allow more searches
   3. Save search history
5. Program debugging
   1. Error handling
   2. Loggin and debugging

Before I begin my iterative development, I must gather information via surveys on potential users and research effectively on current solutions.

Pattern Recognition

Pattern recognition is a vital aspect of my Movie Matcher program because it identifies trends in the user’s previous activity and preferences. By analysing these patterns, the program can detect similarities between movies, such as shared genres, ratings, or actors, and use this information to recommend films that closely match the user’s interests. I must recognise and identify common movie features, that hare recurring attributes (such as genre, rating, actors). Pattern recognition will help me spot common features so I can create consistent data structure as a class or a ditionary. Pattern recognition also helps group and categorise data by identifying which genres or ratings appear most frequently, making it easier to organise and filter results. Additionally, by spotting predictable attributes, such as popular genres, the program can optimise searching by pre-grouping movies, improving both efficiency and accuracy when generating recommendations.

Algorithmic Thinking

Algorithmic thinking is important in my Movie Matcher program as it allows me to design a clear, step-by-step process to solve the problem of recommending movies. First, the program will create a profile for the user, so their ratings can be saved and potentially create a trusted profile which others can use for their own movie preferences. Next, the user can search the dictorary/class to find a specific movie to rate, the rating will be published and then recommended filns will be outputted to the user. This algorithm will remember the rating, Genre, producers, actors and other key details in the user’s history. This approach uses selection, as the program will decide whether to include or exclude a movie based on conditions (e.g., if the rating is below the user’s minimum, it will be ignored), and iteration, as it will loop through each movie in the dataset to perform these checks. By breaking the problem into smaller, logical steps, the program ensures that the recommendations are generated efficiently and accurately.

## Stakeholders

The stakeholders I am looking for this project are movie fanatics looking for reliable recommendations, critics who want to publish their reviews, and users who are dissatisfied with the current available platforms for movie reviews. Data collectors would largely benefit from the reviews published on Movie Matcher further being an important stakeholder in the program.

My first stakeholder is Mr Yogesh Agarwal, who represents an example of my projected end user. Mr Agarwal is an experienced Advertisement and Short Movie Director with extensive knowledge of the film industry. He is directly impacted by movie rating platforms, as these systems allow his productions to gain greater visibility and reach a wider audience. I will maintain close communication with Mr Agarwal throughout the development of the recommendation system. By carefully designing the questions asked to users, I aim to generate meaningful data that contributes to an overall percentage rating of each film. With this solution in place, Mr Agarwal will be able to clearly identify what aspects of his movies resonate most with viewers, helping him to refine future projects and maximise audience engagement.

First name:

Yogesh

Last name:

Agarwal

Email:

[toyogesh@icloud.com](mailto:toyogesh@icloud.com)

Occupation:

Movie and advertisement director

The next stakeholder of Movie Matcher is my classmate, Mr Munraj Bhatal, who represents another example of a projected end user. Mr Bhatal is a passionate movie enthusiast who frequently leaves reviews after watching films. However, he is dissatisfied with the current review systems, as he feels that excessive filtering of posts often results in dishonest and less genuine feedback.Following my survey, Mr Bhatal expressed that a personalised recommendation system would be highly beneficial, as it would help him make better use of his valuable watch time. His feedback will be useful in understanding the best user interface to use, making the browser easy to access and use, so I will keep in close contact with him through testing of the user interface and user experience. While I recognise that his input reflects the perspective of a single user and that other individuals may have different preferences, his feedback will provide a valuable indication of whether the system is meeting the needs of end users.

First name:

Munraj

Last name:

Bhathal

Email:

munrajbhathal@outlook.com

Occupation:

Student

My final stakeholder is Mrs Vandana Awasthi, a software engineer with professional experience in product coding. Her expertise will be highly valuable in providing technical validation for my project. In addition to her testing background, she also recognises the importance of personalised recommendations and appreciates how they can enhance user engagement. During the testing stage, Mrs Awasthi will evaluate the accuracy of the recommendation algorithm and assess whether the system produces meaningful results for users. She will also provide feedback on the graphical user interfaces (GUIs), ensuring that they are intuitive, accessible, and user-friendly. Her input will help identify any technical or usability issues, allowing me to refine the system and improve its overall quality.

First name:

Vandana

Last name:

Awasthi

Email:

Avandanaa@yahoo.co.uk

Occupation:

Software Developer

Product Research

Current Solutions

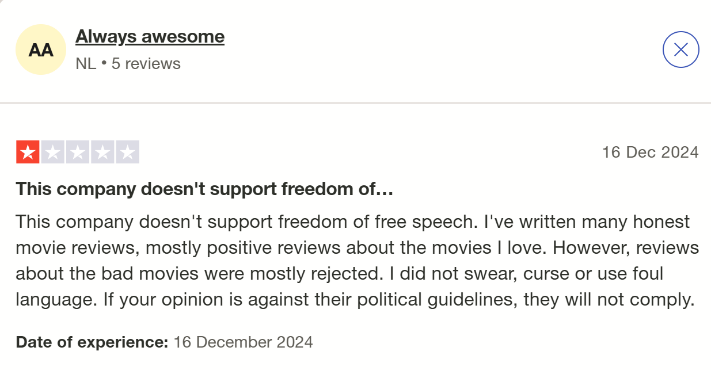
There are many websites that allow users to rate movies and publish reviews; however, there is a gap in the market as no existing platform specifically focuses on matching users with movies based on their preferences. Examples include :

1. **Imdb** - User ratings, critic reviews, and movie details.
2. **Rotten Tomatoes** – Critic and audience scores, aggregated reviews
3. **Meta Critic** – Weighted average critic and user reviews
4. **Letterboxd** - Social platform for movie fans to log, rate, and review films.
5. **FilmAffinity** – User ratings and reviews, popular in some regions.
6. **AllMovie** – Detailed reviews, cast, and production information.
7. **RogerEbert.com** – Professional critic reviews and film analysis.
8. **The Guardian / Variety / IndieWire (Film Sections)** – Professional critic reviews.
9. **Fandango** – Movie ratings and user reviews (mainly for cinema releases).
10. **Screen Rant / Collider (Review Sections)** – Online critic and editorial reviews.

The top 3 examples are the most used with approximately 250 million monthly users combined. I will incorporate features from these successful platforms to use in my project.

Reviews on current solutions

Although Imdb, Metacritic and RottenTomatoes are very successful platforms they are deemed ineffective by these reviews:

 A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a post

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

As displayed above the prime complaint from clients is the overuse of filtering of comments and reviews. This defeats the point of the program therefore I must take this into account and find a suitable method on filtering comments so users can remain honest without being hateful.

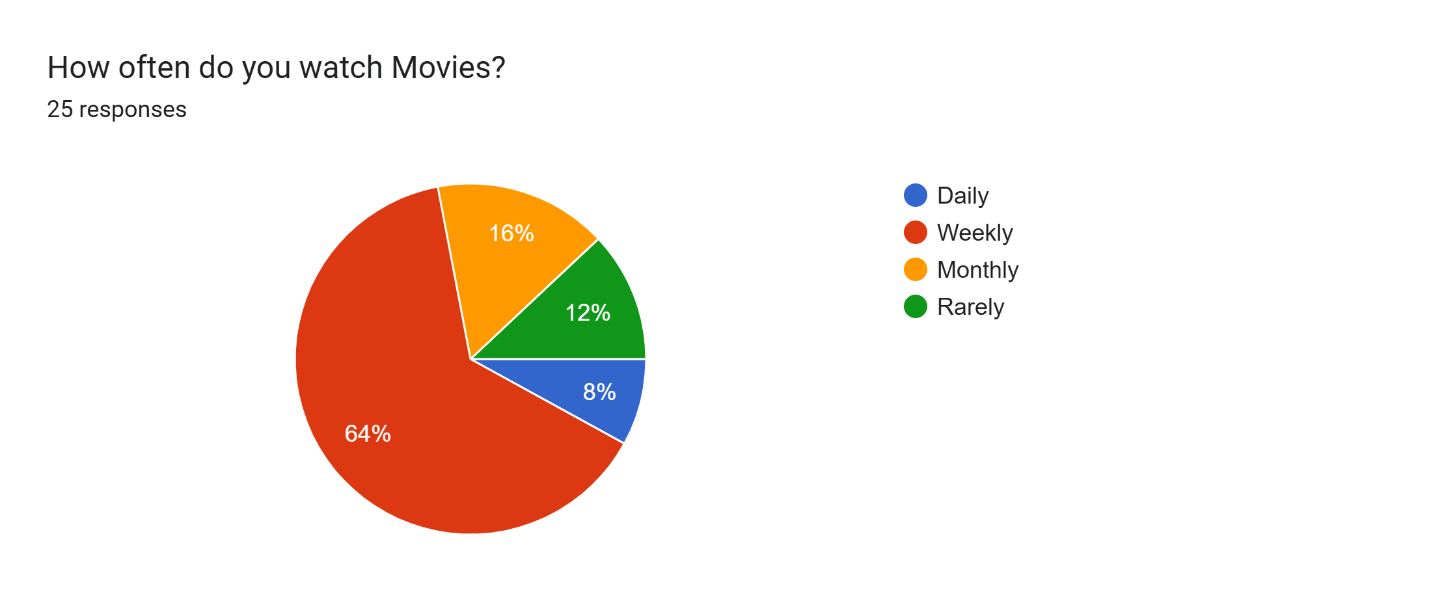
|  |  |
| --- | --- |
| **MetaCritic**    **RottenTomatoes**    **IMDb** | The common theme of the positive reviews on these platform is the neat layout and presentation of movies, information and reviews, which I will definitely incorporate in my design. A quote from a review for IMDb “there are lists of top movies or tv series by years. it’s very convenient.” displays a key attribute I want Movie Matcher to present. The negative reviews are often very similar, as stated above, which is why I am determined to make Movie Matcher a trustworthy and reliable source. It will provide honest and constructive opinions while avoiding the spread of unnecessary negativity or hate. |

Survey

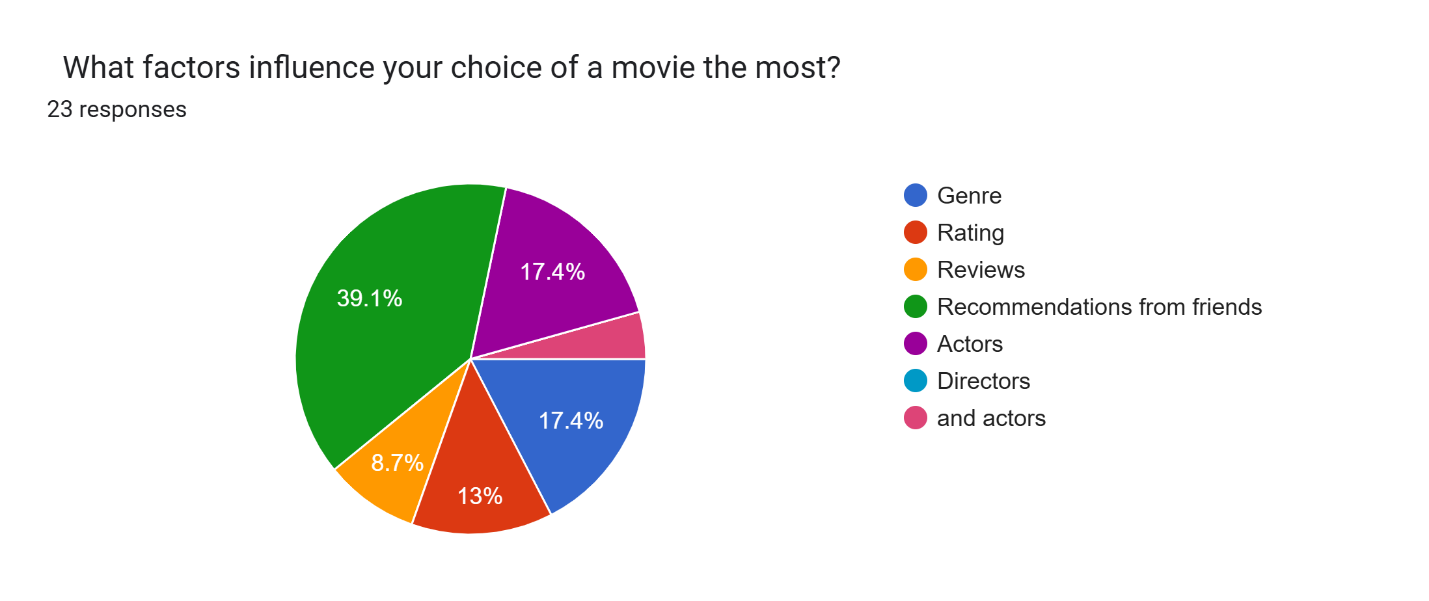
I conducted a questionnaire for the possible clients of Movie Matcher. The questions included:

1. **How often do you watch movies?** (Daily / Weekly / Monthly / Rarely)
2. **What factors influence your choice of a movie the most?** (Genre / Rating / Reviews / Recommendations / Actors / Director / Other)
3. **Which platforms do you currently use to find movie reviews or ratings?** (IMDb / Rotten Tomatoes / Metacritic / Letterboxd / Other)
4. **How satisfied are you with these platforms for finding new movies to watch?** (Very Satisfied / Satisfied / Neutral / Unsatisfied / Very Unsatisfied)
5. **What do you dislike about current movie review or recommendation websites?** (Open-ended)
6. **Would you find it useful if a website suggested movies based on your past preferences?** (Yes / No / Maybe)
7. **Which features would you find most helpful in a movie-matching site?** (Personalised recommendations / Honest user reviews / Filter by genre & rating / Save watchlist / Other)
8. **What device would you mainly use to access Movie Matcher?** (Phone / Tablet / Laptop / Desktop)
9. **How do you prefer movie recommendations to be displayed?** (Simple list / Grid with posters / Detailed review cards / Other)
10. **How important is it for the interface to look visually appealing compared to just being functional?** (Very Important / Somewhat Important / Not Important)

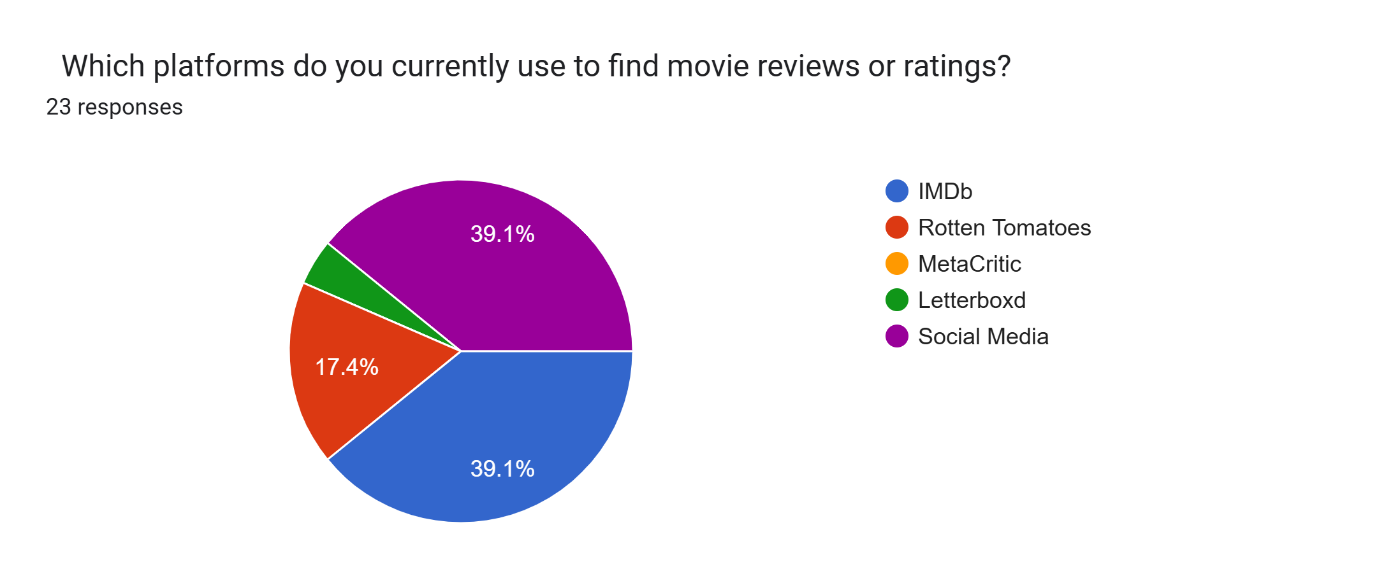
The results were as follows:

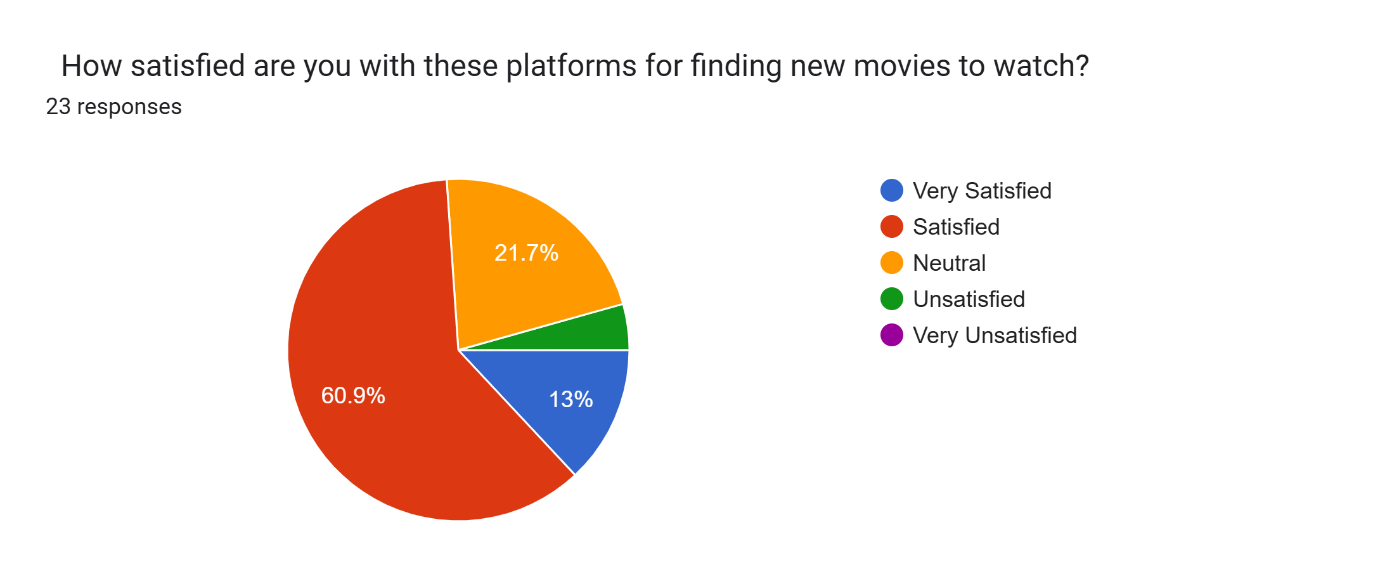


This question indicates how often people will use Movie Matcher, with a majority result of weekly.



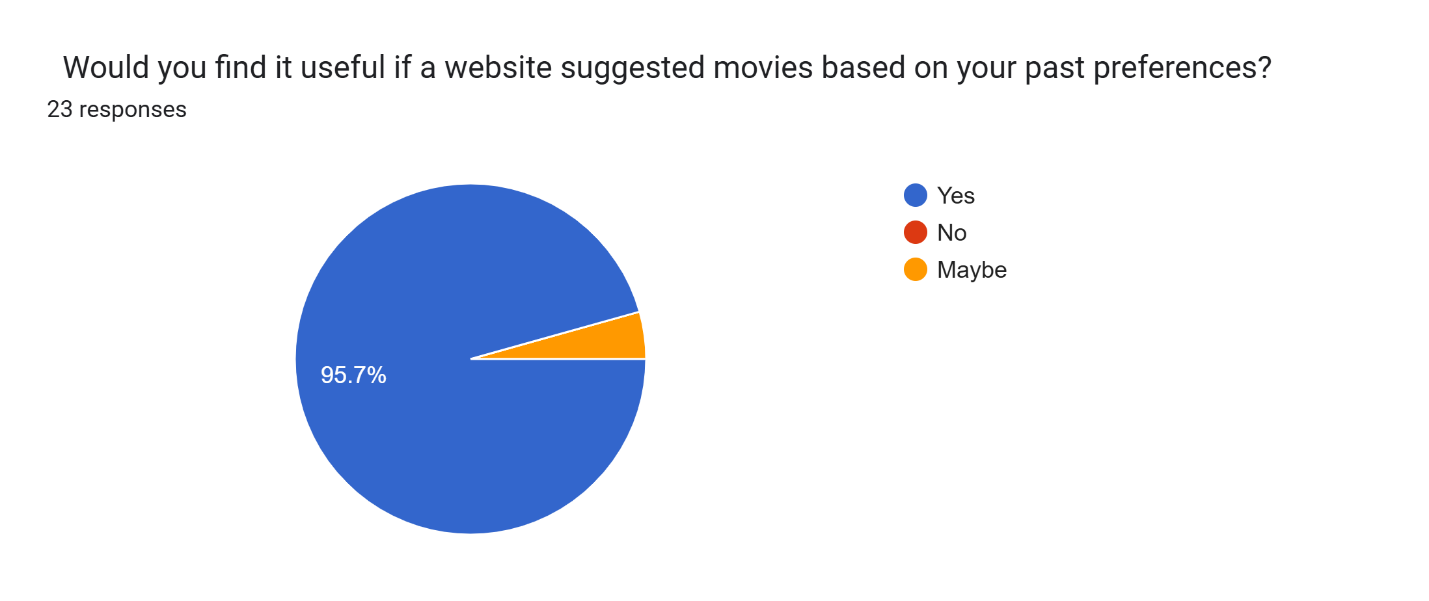
This suggests that users value personal and trusted recommendations over critic reviews or ratings. For Movie Matcher, this means the recommendation system should prioritise personalised suggestions that feel similar to friend recommendations, possibly by comparing users’ preferences with others who have similar tastes. Genre and actor-based filtering should also be key features, as these were rated highly. Meanwhile, rating and review features can be included but should be secondary to personalised recommendations.

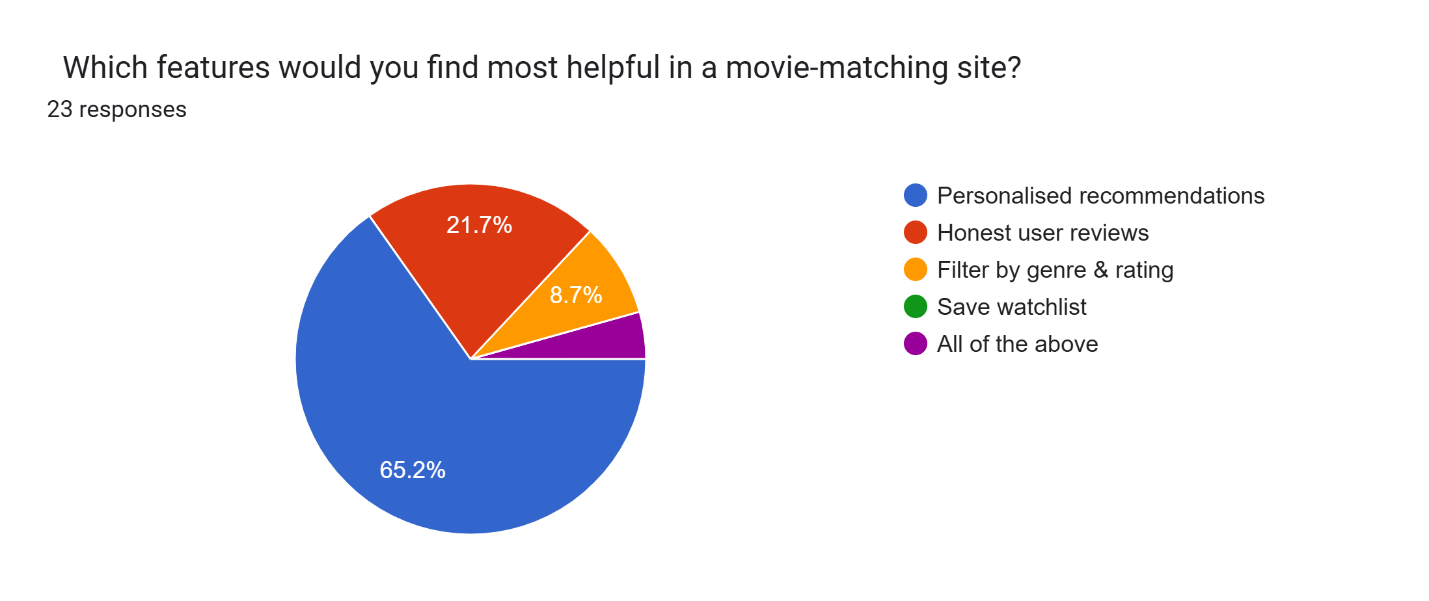


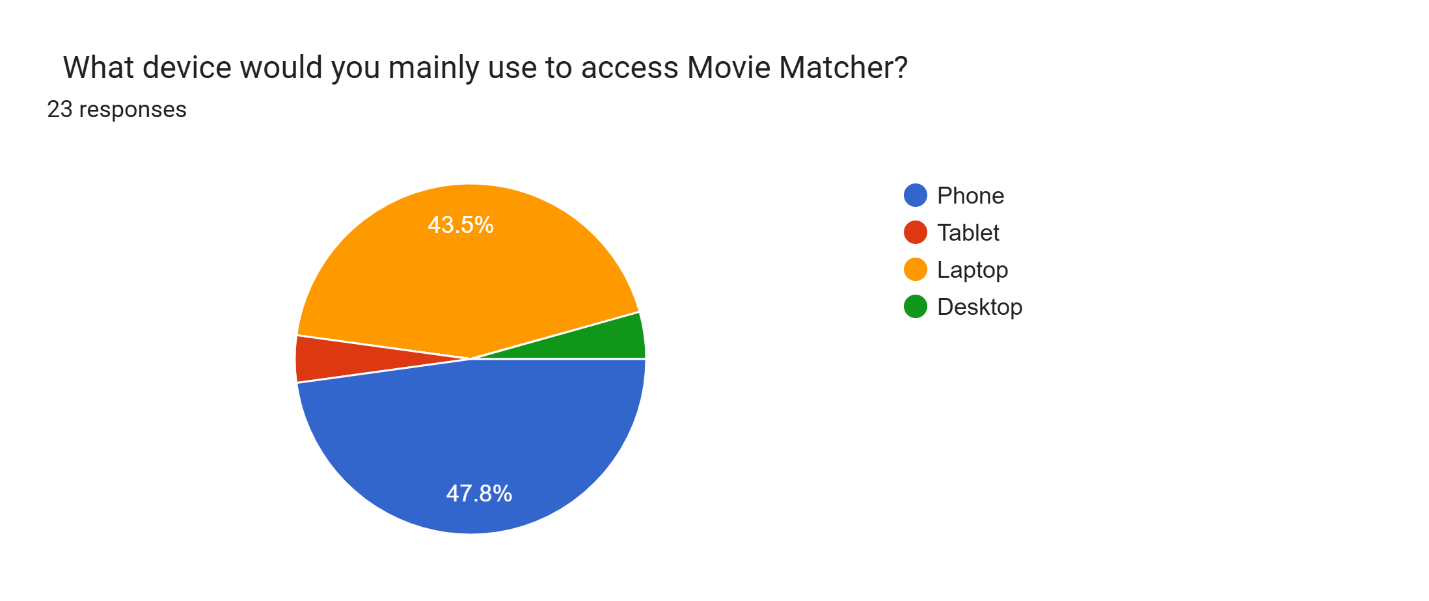
IMDb was rated the most used platform along with social media, proving that users value both professional databases (like IMDb, which provides ratings, cast information, and reviews) and more casual, friend-based recommendations found on social media. For Movie Matcher, this highlights the importance of combining a structured movie information while also embodying the personalised and social feel of friend recommendations. A screenshot of a computer

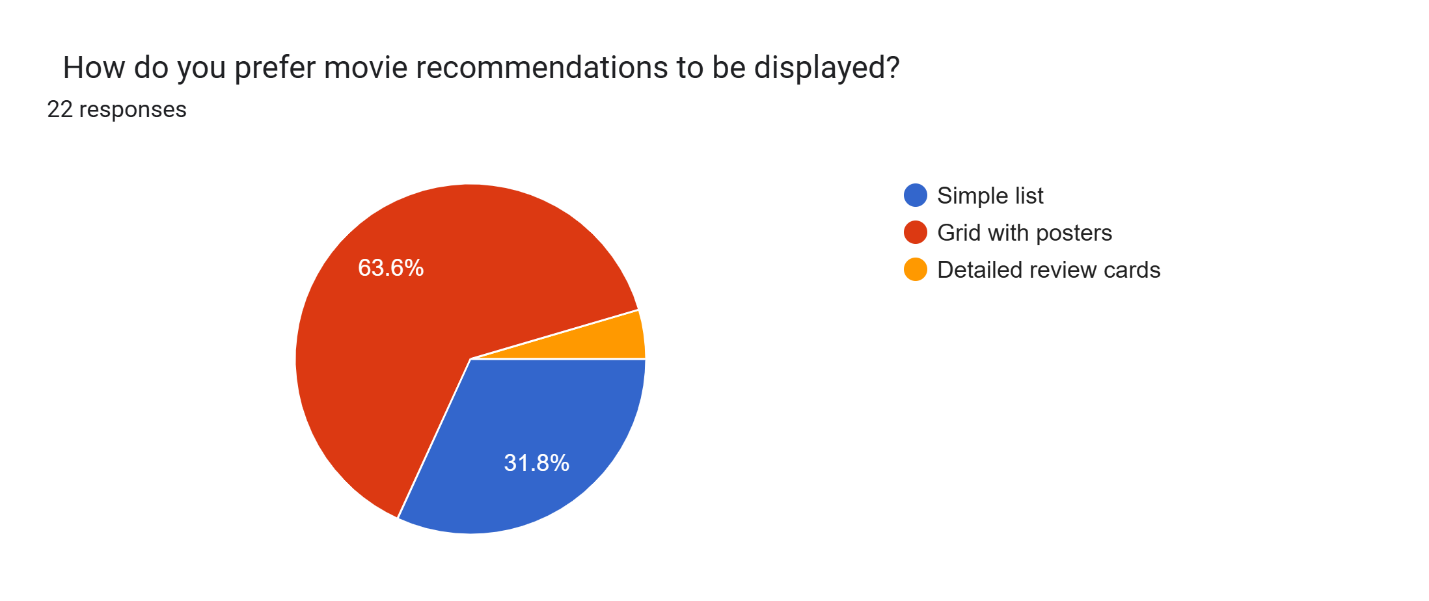
AI-generated content may be incorrect.A screenshot of a computer

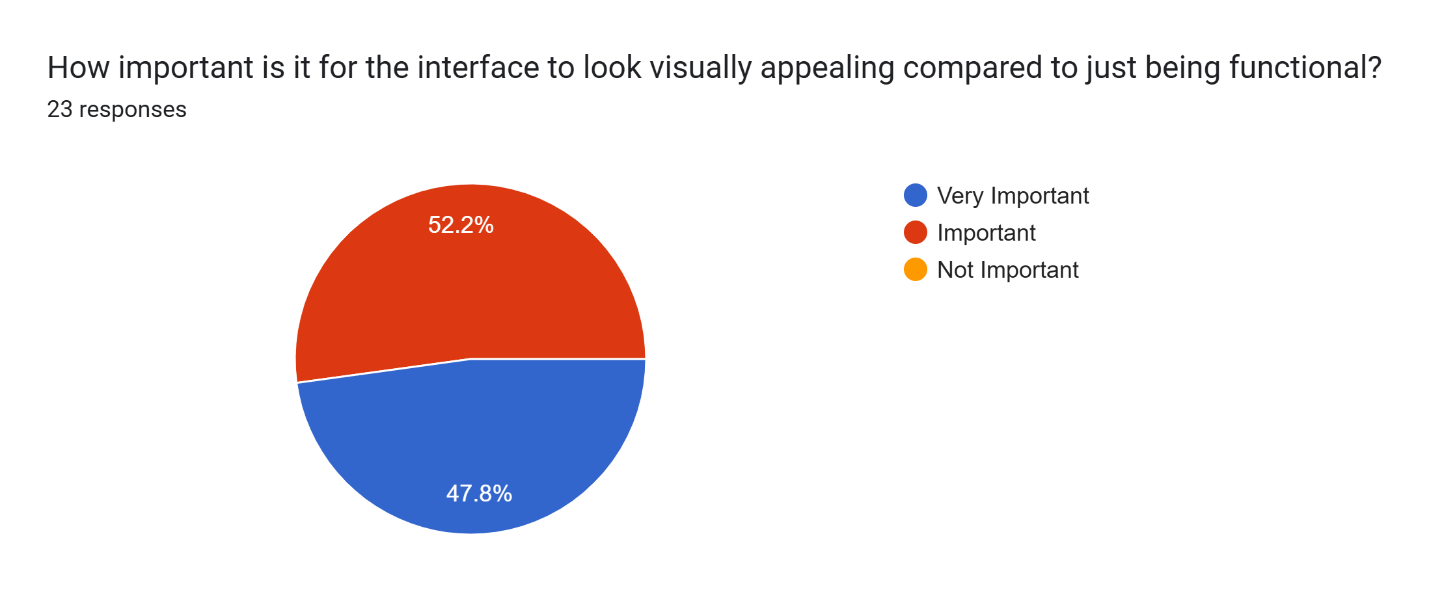
AI-generated content may be incorrect.

The written responses provided valuable insights into additional features for Movie Matcher. Many users highlighted the importance of displaying the reviewer’s age, as this can influence how others interpret the review, particularly if it is targeted towards a different age group. This will be incorporated into the design to allow users to better judge the relevance of a review. Additionally, I will ensure that both positive and negative reviews are presented to avoid bias, giving users a more honest and balanced experience when using Movie Matcher.

This clearly supports the core functionality of Movie Matcher, confirming that personalised recommendations should be the primary feature.

When asked which features would be most helpful, the majority selecting personal recommendations making it the highest priority feature to implement. This indicates that Movie Matcher should focus on recommendations first but still integrate other features to enhance user experience.The user interface for movie matcher must be fit with a responsive design optimized for smaller screens, touch navigation, and fast loading times, while still functioning well on laptops.

 This suggests that users favour a visual and image-based layout, as movie posters allow for quick recognition and make browsing more engaging. Movie Matcher should therefore prioritise a poster grid view, with optional text-based details available on click or hover for users who want more information.



This clearly indicates that users value a well-designed, aesthetically pleasing interface almost as much as functionality. As a result, Movie Matcher should incorporate clean layouts, high-quality images, and a modern design style to attract and retain users, rather than focusing solely on functionality.

Specifications

After I have conducted my research, I will now label the essential features my program should have.

|  |  |
| --- | --- |
| **Essential Features** | **Justification** |
| A clear tab/comma separated database which holds the title of the film, producers, actors, yearn of the film | It is easier for the programmer to read, understand the database as well as output the appropriate information when the user searched for a movie. |
| A detailed user profile including the user’s age, username, strong password, favourite genres and a place to store future reviews and information. There should be a clear home where the user can make any changes to their profile. | The profile ensures that all of their reviews and recommendations are stored neatly under their profile making it easily accessible. The profile provides authenticity and for piracy issues. They must also enter their age and gender as due to my survey results these factors can play a big factor in the possible recommendations for other users. |
| A distinct search bar for users to be able to search for movies using an OOP approach, where movies are stored as a class. | A distinct search bar is essential because it improves **usability and efficiency**, letting users quickly find movies without scrolling. Using an**OOP approach,** movies as class objects make searching more structured and scalable, enabling clean filtering, retrieval, and future expansion (e.g., by genre, rating, or actor). |
| The user should have a create review function where they can publicly post their review and rating of a film. This can be seen by other users too. | Allows users share opinions and ratings publicly, fostering engagement and community interaction while enhancing the credibility and usefulness of the platform. |
| The rating given to each user will be based off answering 5 questions, where the user will rank their answer out of 5. The questions are as follows:   1. On a scale of 1 to 5, how likely are you to recommend this movie to a friend (1- definitely, 5-not a chance)? 2. Would you watch this film again (1- definitely, 5- impossible)? 3. Were you engaged the entire film (1-the entire time, 5- not once)? 4. How would you rate the acting performance of the cast (1-spectacular 5-terrible)? 5. How would you rate the visual elements (cinematography, special effects, set design) (1- spectacular, 5- terrible)?   These results will be added together to generate a score out of 25, these results will be converted into a percentage which will represent the overall review from the user. | This approach makes it easier to create a percentage rating for each film, and it also makes ratings comparable to other movies as they have been assessed against the same criteria. By converting individual question scores into a standardised percentage, the system ensures that every film is judged fairly and consistently. This not only improves clarity for users when interpreting ratings but also enhances the reliability of comparisons across different films. Furthermore, using structured questions provides a balanced evaluation of multiple aspects of the movie, such as engagement, acting, and visuals, resulting in a more comprehensive and meaningful overall score |
| The key USP of Movie Matcher is the personalised recommendation system for the user. This will be achieved by using the ratings and information gathered from the previous questions to determine similar films the user may enjoy.  These recommendations will be directly linked to the specific aspects rated highly by the user. For example, if the user gives a high rating in Question 4 (acting performance), the system will suggest films featuring a similar cast or the same lead actor. Likewise, if the user rates Question 5 (cinematography and visual elements) highly, movies with similar producers or cinematographers will be recommended. This targeted approach ensures that recommendations are personalised based on the specific elements the user values most in a film. | This feature is essential because it provides users with **personalised recommendations tailored to their individual preferences**, making the platform unique and engaging. By linking suggestions directly to the aspects a user values most—such as acting, cinematography, or overall engagement—the system ensures that recommendations are **relevant, meaningful, and user‑centric.** This targeted approach not only enhances satisfaction by helping users discover films they are more likely to enjoy, but also differentiates Movie Matcher from generic recommendation systems, establishing its **key USP** |

Limitations

There are a few limitations to my program which prevents me from carrying out a few key features to their full potential.

One limitation of Movie Matcher is the size of its movie database. If the dataset is small, the range of recommendations will be limited, which could lead to repetitive or less relevant suggestions for users. Similarly, if new movies are not added frequently, the system may quickly become outdated, reducing its long-term usefulness.

Additionally, for my project, I cannot update the database when new films are released, which could potentially make the program outdated relatively quickly.

The recommendation mechanism itself will be relatively straightforward due to the nature of the project. It will depend significantly on comparing attributes such as genre, actors, and cinematography, rather than depending on advanced algorithms. As a result, recommendations will find a note of familiarity or otherwise become overly simplistic, especially for users with varied tastes.  
  
The accuracy of the program also depends on user input. Unless users rate movies truthfully and consistently, the quality of recommendations will be poor. Additionally, the system will not learn nor observe a user's shifting preferences over time since this would take much more complex programming and room for storage than the scope of this project allows for

A great addition to Movie Matcher would be a social media link from the user profiles to their accounts such as TikTok, Instagram, and Facebook. This would be helpful in determining appropriate matches and recommendations as the algorithm could intertwine with recent searches and activity. However, this feature is currently a limitation of the programme, as integrating external social media platforms requires complex APIs, strict privacy considerations, and user consent, which are beyond the current scope of development.

Finally, since time and resource constraints are an issue, the interface will be working but far less polished and visually rich than commercial websites. While I will make it responsive on laptops and phones, it may not have accessibility features, animations, or community features such as commenting and sharing.

## Hardware and Software Requirements

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hardware/Software Requirements** | **Reason for user** | **Reason for developer** | **Necessity** | |
| Movie Data API (IMDb) | Accurate movie information giving users confidence in the app and improves their experience | Reliable structured data makes it easy to fetch titles, genres etc | Essential for core functionality and user interface | |
| Search API | Fast, accurate search results let users find movies effortlessly | Simple query endpoints reduce complexity in building the search bar | Essential for users too search for a movie to match. | |
| RAM | To run browser smoothly for streaming/searching movies. | To run IDEs, databases, and test servers efficiently. | 8 GB recommended | |
| Processor (CPU) | For smooth browsing and video playback. | For compiling code, running servers, handling multiple processes. | Minimum dual-core, recommended quad-core | |
| Memory (RAM) | To store cookies, cache, and local preferences. | To save code, libraries, logs, and database files. | At least 20–50 GB free space | |
| Database | Stores user reviews, ratings, and login details securely. | Required to implement user authentication, recommendations, and content storage. | Very necessary | |
| Version Control | - | To track changes in code, collaborate with other developers. | Necessary for development | |
| APIs | Lets users fetch real-time movie info, trailers, recommendations. | Developer integrates external APIs (e.g., TMDb/IMDb). | Very necessary if pulling external data | |
| Internet Connection | To browse, stream trailers, and access recommendations. | To fetch libraries, push/pull code, test APIs, and deploy website. | Essential | |
| (Optional) GPU | Smoother visuals if animations are added | Optional acceleration for ML-based recommendation experiments | Optional |
| Dual-Core CPU | Fast response times when searching or loading recommendations | Efficient code execution for API calls, image handling and packaging | Essential, python and GUI frameworks rely on CPU. |

Visual Studio Code

I am using Visual Studio Code (VS Code) for writing, testing, and debugging code. This IDE, developed by Microsoft, is the most important software requirement for Movie Matcher as it provides a professional environment to code in Python with built‑in shortcuts and productivity features. VS Code is highly beneficial because it offers extensive developer support, including error handling, integrated debugging tools, and quality‑of‑life enhancements that streamline the development process. Features such as syntax highlighting, auto‑completion, and multiple coding tabs make it easier to identify errors, navigate between different sections of code, and maintain clarity in larger projects. Additionally, VS Code supports extensions for version control, database integration, and UI design, making it a versatile tool that ensures the Movie Matcher programme can be developed efficiently and to a high standard.

Database

The database is a key software requirement for Movie Matcher as it stores all movie information, user reviews, and ratings in a structured format. By using an online/downloaded database, the programme can efficiently handle queries from the search bar, generate percentage ratings, and support the personalised recommendation system. The database ensures data is organised**, scalable, and accessible**, making it possible for multiple users to interact with the system simultaneously while maintaining accuracy and consistency. My databases are stored as tab‑separated values (TSV) files, which provide a clear and organised structure. This format makes the data easy to read and access, while also simplifying error handling and ensuring consistency across records. My databases are stored as tab‑separated values (TSV) files, which provide a clear and organised structure. This format makes the data easy to read and access, while also simplifying error handling and ensuring consistency across records.

Operating system QUESTION

API

APIs are a vital software requirement for Movie Matcher, as they allow the programme to connect with external services and fetch real-time data such as movie information, trailers, and recommendations. For users, this enhances the experience by providing up-to-date content directly within the platform. For developers, APIs are essential for integrating third-party sources like TMDb or IMDb, enabling dynamic content delivery and expanding the functionality of the recommendation system.

I am using Windows OS

Evaluation and Success Criteria

To ensure that my application performs its intended function while providing an enjoyable user experience, I have created a clear success criteria. This outlines the key goals I want my program to achieve by the end of development. In addition, I designed a short questionnaire to gather user feedback, helping me evaluate whether the program is effective and enjoyable to use. These steps help confirm that my program is successfully addressing the problem it was designed to solve.

Success Criteria

Below are the key success criteria points/objectives my program must achieve by the end of its function in order to provide the efficient solution. This allows me to gain a better understanding of the key objectives of my application.

|  |  |  |
| --- | --- | --- |
| Number | Criteria | Justification |
| 1. | A menu page with a home button, user profile, clear display of movies based off new in, genres, recommended | Easy accessibility for the user |
| 2. | Make a user profile with important credentials | Allow personalised recommendations and secure access to user-specific features |
| 3. | Application must close/exit on demand | Ensures user control |
| 4. | Setting Menu where the user can adjust their user profile settings | Empowers users to manage preferences and update personal details easily. |
| 5. | Application must allow user to select a movie and give their review | Enables user engagement and contribute to community feedback |
| 6. | Application must allow user to select a movie and give their review | Enables user engagement and contributes to community feedback |
| 7. | Application must display a short questionnaire for the user to generate a review percentage | Reinforces core functionality and supports multiple review entries. |
| 8. | Application must allow user to see other user reviews | To create a community between users, trusting each other’s honest reviews |
| 9. | Application must recommend movies they thin k the user will enjoy | To perform the application’s purpose, if accurate can lead to attraction of more users. |
| 10 | Application must save the users data | To save history, for future recommendations |
| 11. | Recommendations are based off genre, actors, producers etc | Ensures the movie matches are appropriate and relevant |
| 12. | User interface is clean, responsive and intuitive | Improves user experience and accessibility |
| 13. | Data is stored securely in a relational database | Ensures scalability and integrity |
| 14. | No crashes or errors during normal use | Stability is a basic expectation from stakeholders and users |
| 15. | All foreign key relationships are correctly implemented | Ensures data integrity |
| 16. | App loads within 3 seconds on standard hardware | Measurable performance target that reflects real-world usability |

User experience objectives

Below is a table for testers and stakeholders to complete after using the program. It is designed to assess whether the user experience was enjoyable and whether the program successfully met the objectives outlined in the success criteria.

**Name:**

|  |  |  |
| --- | --- | --- |
| Question: | Descriptive Answer: | Rating out of 5 (1, best, 5, worst): |
| Did the program’s algorithm return accurate and relevant movie matches based on your input? |  |  |
| How easy was it to navigate and use the features of the program? |  |  |
| Did the program meet its functional requirements as outlined in the success criteria? |  |  |

Agile methodology

For the development of Movie Matcher, I will follow an approach similar to the Agile methodology by building the program in small, manageable stages and continuously improving it based on testing and feedback. Instead of completing the entire program in one go, I developed it in iterations, first forming the basic movie database, then adding the user profile, and finally implementing the user rating/recommendation feature. After completing each stage, I will test the functionality and made adjustments before moving on to the next part.

# B. Design

<See H446-03 Project Advice Booklet for help and guidance of what must go here.>

## Decomposition tree

My decomposition tree clearly outlines how I will approach the development of the program by breaking it down into smaller, manageable sub-problems. This method allows each individual component to be developed and tested separately. By focusing on one part at a time, I can solve problems more efficiently and integrate each feature smoothly into the final system.

1. User profile -register, edit info on user, and delete, login, storage and ui personalisation
2. Movie database- adding movies, editing movies, name, title, producers, year, cast,
3. Ratings, link users to movies with ratings
4. Recommendations, personalised from algorithms
5. Ai if possible, driven

Menu Screen

**User profile**

**User Profile Creation**

* The system will provide a clear interface for creating a new user profile.
* Users will be required to enter key credentials (name, username, and password).
* New users will also have the option to select their uuioi. preferred movie genres.
* Theyse preferences will be stored and used by recommendation algorithms to personalize movie suggestions.

**Login Functionality**

* Users can log in with their username and password.
* The system will retrieve and validate login data from a CSV file containing all registered users, including their name, username, password, and genre preferences.

**Validation and Error Handling**

* The login system will provide clear, user-friendly outputs when credentials are invalid.
* Validation rules include:
* Password must be at least 6 characters long.
* Password must contain at least one digit.
* Password must include both uppercase and lowercase letters.
* Username must be at least 6 characters long.
* If either the username or password fails to meet these criteria, the system will display specific error messages to guide the user in correcting their input.

**Movie Database**

* Once the user has logged in, a search bar should appear at the top of the screen available to access anytime the user wants to search for a film
* The available online databases are downloaded and used for this function:
* [name.basics.tsv.gz](https://datasets.imdbws.com/name.basics.tsv.gz)
* [title.akas.tsv.gz](https://datasets.imdbws.com/title.akas.tsv.gz)
* [title.basics.tsv.gz](https://datasets.imdbws.com/title.basics.tsv.gz)
* [title.crew.tsv.gz](https://datasets.imdbws.com/title.crew.tsv.gz)
* [title.episode.tsv.gz](https://datasets.imdbws.com/title.episode.tsv.gz)
* [title.principals.tsv.gz](https://datasets.imdbws.com/title.principals.tsv.gz)
* [title.ratings.tsv.gz](https://datasets.imdbws.com/title.ratings.tsv.gz)
* The information on these files are copied into excel and stored as a well differentiated database, with clear divisions
* These databases will be stored into clear columns which are easy to look through when debugging.

**Ratings**

* Rating Categories: Break down the 5 questions into
* If a user wants to rate a film they will answer a 5 question questionnaire as explained in the analysis section which will give the program a good understanding on how much the user likes the movie
* Each question is scored out of 20 and this generates a percentage rating for the user which is published, for other users to see

**B**

## Key data strucures needed

Movie Matcher is made up of 3 key tables in the movie\_matcher\_db database: tbl\_users, tbl\_movies, tbl\_reccomendations and tbl\_reviews.

**Database FileName:** movie\_matcher\_db

**TableName :** tbl\_users

**Description :** This table stores all registered user information, including their unique ID, age, preferences, and login details. This table is used to personalise movie recommendations and link users to their reviews and activity in the system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **DataType** | **Description** | **Example** |
| user\_id  primary key  foreign key | Integer | Unique identifier for each user | 1023 |
| username | String | User’s login name or display name | aaryaa\_awasthi |
| age | Integer | User's age (for age-based filtering) | 17 |
| email | String | For validation, and if the user forgets their password | a.awashthi@gmail.co.uk |
| genre\_preferences | String/JSON | Stored preferences | ["Action", "Sci-Fi", "Drama"] |

**Database FileName:** movie\_matcher\_db

**TableName :** tbl\_movies

**Description :** This table Contains core information about each movie, including title, genre, cast, release year, and average rating. This is used as the main dataset from which recommendations are generated.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **DataType** | **Description** | **Example** |
| movie\_id  primary key  foreign key | Integer | Unique ID for each movie | 1001 |
| movie\_title | String | Movie Name | La La Land |
| movie\_genre | String | Main genre | Musical |
| release\_year | String | Year the movie was released | 2016 |
| main\_cast | String/array | Key actors | Emma Stone, Ryan Gosling |
| avg\_user\_rating | Float | Average rating from users | 8.0 |
| age\_certification | string | Legal age restrictions | PG-13 |

**Database FileName:** movie\_matcher\_db

**TableName :** tbl\_reviews

**Description :** This table Stores user-generated ratings and optional written reviews for each movie.  
Used to support honest, community-based feedback.

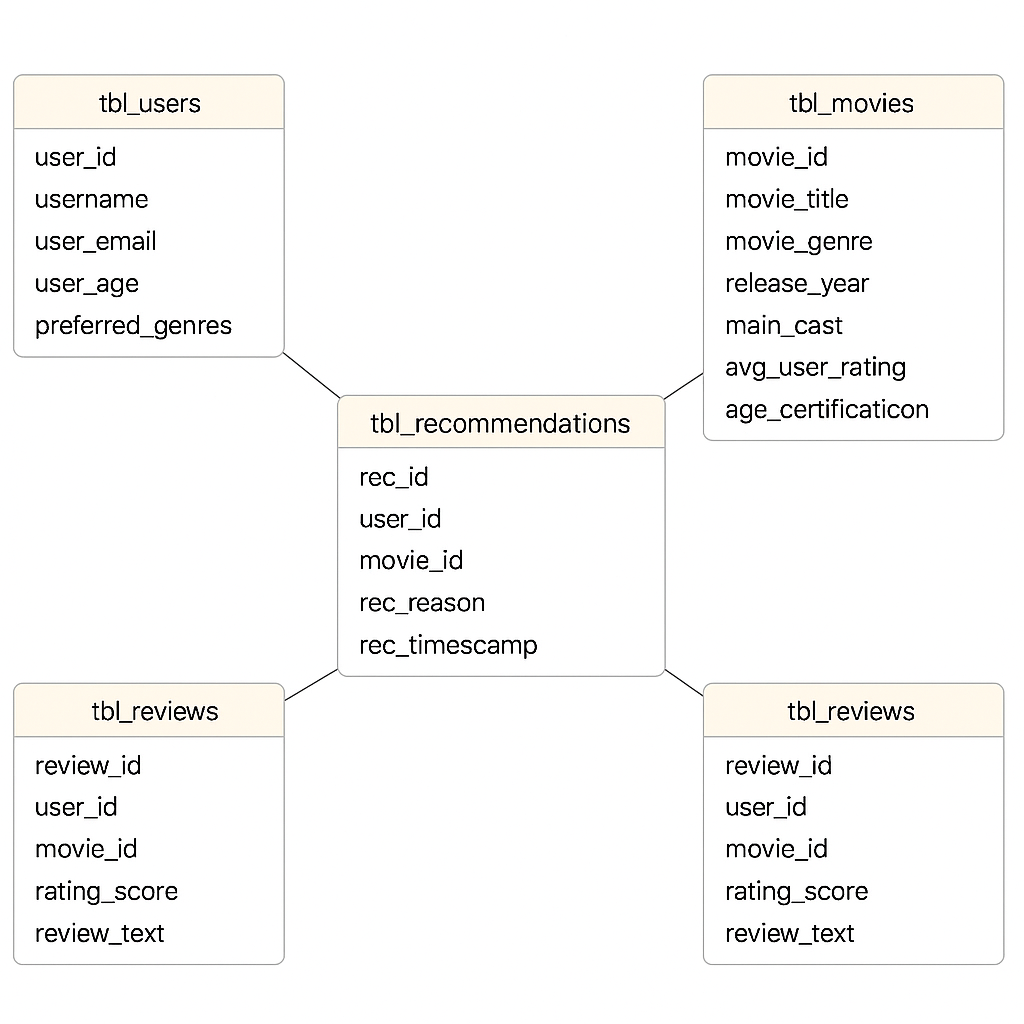
|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **DataType** | **Description** | **Example** |
| review\_id  primary key | Integer | Unique identifier for each review | 501 |
| user\_id\_fk | Integer | Foreign key from tbl\_users | 10001 |
| movie\_id\_fk | Integer | Foreign key from tbl\_movies | 1002 |
| rating\_score | Integer | User rating | 5 |
| review\_text | string | Optional review/comment from user | “An amazing performance by Heath Ledger. Must watch!” |
| review\_date | DateTime | Date and time the review was generated | 2025-07-28 |

**Database FileName:** movie\_matcher\_db

**TableName :** tbl\_reccomendation

**Description :** This table stores all registered user information, including their unique ID, age, preferences, and login details. This table is used to personalise movie recommendations and link users to their reviews and activity in the system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **DataType** | **Description** | **Example** |
| rec\_id  primary key | Integer | Primary key, unique ID for the recommendation | 9001 |
| user\_id | Integer | Foreign key linking to tbl\_users.user\_id | 10001 |
| movie\_id | Integer | Foreign key linking to tbl\_movies.movie\_id | 1003 |
| reason | String | Text explaining the reason for recommendation | “Same lead actor and high rating” |
| date\_generated | DateTime | Date and time recommendation was generated | 2025-07-28 |

Star Schema  The star schema in my Movie Matcher system uses a central fact table (tbl\_rec) to store recommendation events, linking to surrounding dimension tables (tbl\_users, tbl\_movies, tbl\_reviews). This structure supports efficient querying and clear separation of transactional data (recommendations) from descriptive data (user details, movie info, and reviews).

## relationship

|  |  |  |
| --- | --- | --- |
| **Diagram** | **Relationship** | **Reason** |
| A diagram of a diagram  AI-generated content may be incorrect. | One-to-many relationship | This is because each user can write multiple reviews, creating a one-to-many relationship between the user profile and the reviews. |
| make an image like this but replace the tbl_reviews with tbl_reccommendations | One-to-many relationship | One record in the tbl\_users table can correspond to many records in the tbl\_recommendations table. |
| Create an image showing a one-to-many relationship between tbl_movies and tbl_reviews, where one movie has many reviews. | One-to-many relationship | One movie can have many reviews, but each review is linked to only one movie. |
| Create a diagram showing a many-to-many relationship between tbl_users and tbl_movies. | Many-to-many relationship | One user can watch many movies and one movie can be watched by many users. |

## Algorithms needed

The flowchart below represents the abstracted algorithm for my entire program, incorporating the key elements identified in the decomposition tree above. A diagram of a flowchart

AI-generated content may be incorrect.

**The first step for users is to create their user profile for which their pseudocode is presented below:**

FUNCTION password\_check(password)

IF length(password) < 6 THEN

PRINT "Password length should be at least 6"

RETURN False

ENDIF

IF length(password) > 20 THEN

PRINT "Password length should not be greater than 20"

RETURN False

ENDIF

IF password does not contain a digit THEN

PRINT "Password should have at least one numeral"

RETURN False

ENDIF

IF password does not contain an uppercase letter THEN

PRINT "Password should have at least one uppercase letter"

RETURN False

ENDIF

IF password does not contain a lowercase letter THEN

PRINT "Password should have at least one lowercase letter"

RETURN False

ENDIF

RETURN True

END FUNCTION

SET existing\_profile ← False

SET user\_profile\_created ← False

WHILE existing\_profile = False

OUTPUT "Do you already have a MovieMatcher account? (Enter: yes or no)"

INPUT user\_response

CONVERT user\_response TO lowercase

IF user\_response = "yes" THEN

SET existing\_profile ← True

OUTPUT "Enter your username:"

INPUT username

OUTPUT "Enter your password:"

INPUT password

OUTPUT "Signing in as " + username

// Here: Add code to verify stored credentials

ELSE IF user\_response = "no" THEN

WHILE user\_profile\_created = False

OUTPUT "Please create a username (minimum 6 characters):"

INPUT username

IF length(username) < 6 THEN

OUTPUT "Username must be at least 6 characters long"

CONTINUE to next loop iteration

ENDIF

// Password creation loop

REPEAT

OUTPUT "Please create a strong password:"

INPUT password

IF password\_check(password) = False THEN

CONTINUE to next iteration of REPEAT

ENDIF

OUTPUT "Please re-enter your password to confirm:"

INPUT confirm\_password

IF password ≠ confirm\_password THEN

OUTPUT "Passwords do not match. Please try again."

ELSE

EXIT REPEAT

ENDIF

UNTIL password is valid AND matches confirmation

OUTPUT "Account created successfully! Welcome, " + username

SET user\_profile\_created ← True

END WHILE

ELSE

OUTPUT "Invalid input. Please enter 'yes' or 'no'."

ENDIF

END WHILE

**The following algorithmic pseudocode is for when the user has created an appropriate username and password, and they are now logging into MovieMatcher. This is done by comparing the details with the profiles in the created csv file.**

IF user\_response = "yes" THEN

SET existing\_profile ← True

OUTPUT "Enter your username:"

INPUT username

OUTPUT "Enter your password:"

INPUT password

OUTPUT "Signing in as " + username

// Open the CSV file containing stored credentials

OPEN "user\_credentials.csv" FOR reading AS file

SET login\_successful ← False

// Read each line (each record contains username, password)

WHILE NOT end\_of\_file(file)

READ line FROM file

SPLIT line BY "," INTO stored\_username, stored\_password

IF username = stored\_username AND password = stored\_password THEN

SET login\_successful ← True

BREAK

ENDIF

END WHILE

CLOSE file

IF login\_successful = True THEN

OUTPUT "Login successful! Welcome back, " + username

ELSE

OUTPUT "Incorrect username or password. Please try again."

ENDIF

**This next pseudocode models Movie Matcher’s review stage, guiding the user through a simple decision flow:**

1. **Browse & Select – The user is shown a list of available movies and chooses one.**
2. **Decide – They confirm whether they want to leave a review for the selected movie.**
3. **Review Input – If yes, they enter a rating (1–5) and a short written review.**
4. **Validation – The system checks the rating range and ensures the review text isn’t empty.**
5. **Submission – Valid reviews are stored in the database; invalid ones prompt an error.**
6. **No Review Path – If the user declines, the system simply returns them to the main menu.**

**The process keeps user interaction smooth while ensuring all stored reviews meet basic quality checks.**

FUNCTION main()

INITIALIZE reviews\_database as empty list

WHILE True

DISPLAY "1. Add Review"

DISPLAY "2. View Reviews"

DISPLAY "3. Search Reviews"

DISPLAY "4. Exit"

INPUT choice

//options for the user to choose if they want to leave a review

IF choice == 1

CALL add\_review(reviews\_database)

ELSE IF choice == 2

CALL display\_reviews(reviews\_database)

ELSE IF choice == 3

CALL search\_reviews(reviews\_database)

ELSE IF choice == 4

DISPLAY "Exiting... Goodbye!"

BREAK

ELSE

DISPLAY "Invalid choice. Please try again."

END WHILE

END FUNCTION

//if the user decides to add a review

FUNCTION add\_review(database)

INPUT reviewer\_name

INPUT product\_or\_service\_name

INPUT rating (1-5)

INPUT review\_text

review\_entry = {

"name": reviewer\_name,

"item": product\_or\_service\_name,

"rating": rating,

"text": review\_text,

"date": CURRENT\_DATE

}

IF validate\_review(review\_entry) == TRUE

APPEND review\_entry TO database

DISPLAY "Review added successfully!"

ELSE

DISPLAY "Review not added. Validation failed."

END FUNCTION

//users rating 1-5

FUNCTION validate\_review(review)

IF review.rating < 1 OR review.rating > 5

RETURN FALSE

IF LENGTH(review.text) == 0

RETURN FALSE

RETURN TRUE

END FUNCTION

FUNCTION display\_reviews(database)

IF LENGTH(database) == 0

DISPLAY "No reviews available."

RETURN

FOR each review IN database

DISPLAY "Reviewer:", review.name

DISPLAY "Item:", review.item

DISPLAY "Rating:", review.rating

DISPLAY "Review:", review.text

DISPLAY "Date:", review.date

DISPLAY "----------------------"

END FOR

END FUNCTION

//searching for an item in the database

FUNCTION search\_reviews(database)

INPUT search\_term

found = FALSE

FOR each review IN database

IF search\_term IN review.item OR search\_term IN review.text

DISPLAY review

found = TRUE

END FOR

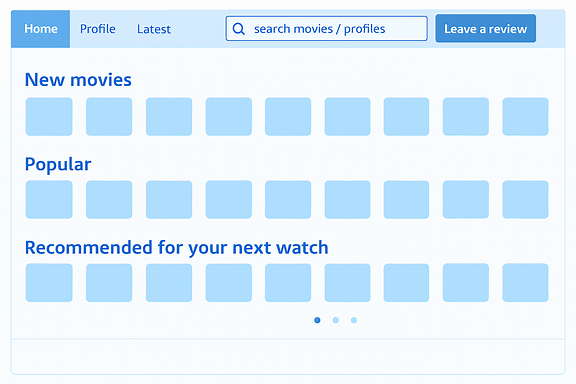
IF found == FALSE

DISPLAY "No matching reviews found."

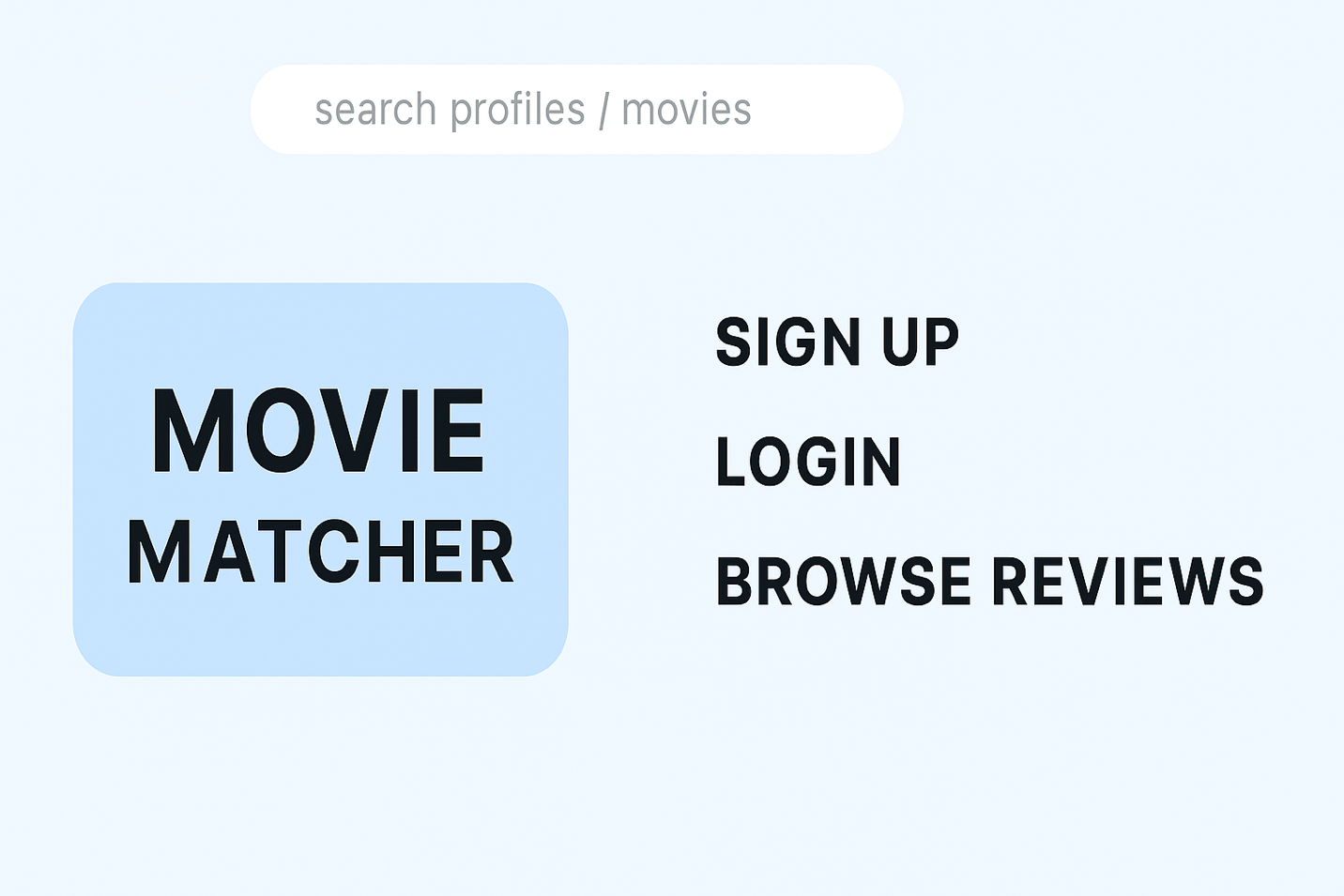
END FUNCTION

## USability features

Movie Matcher is an interactive program based soley on user experience therefore it is essential the program displays key usability features such as a menu screen, search bar, exit,

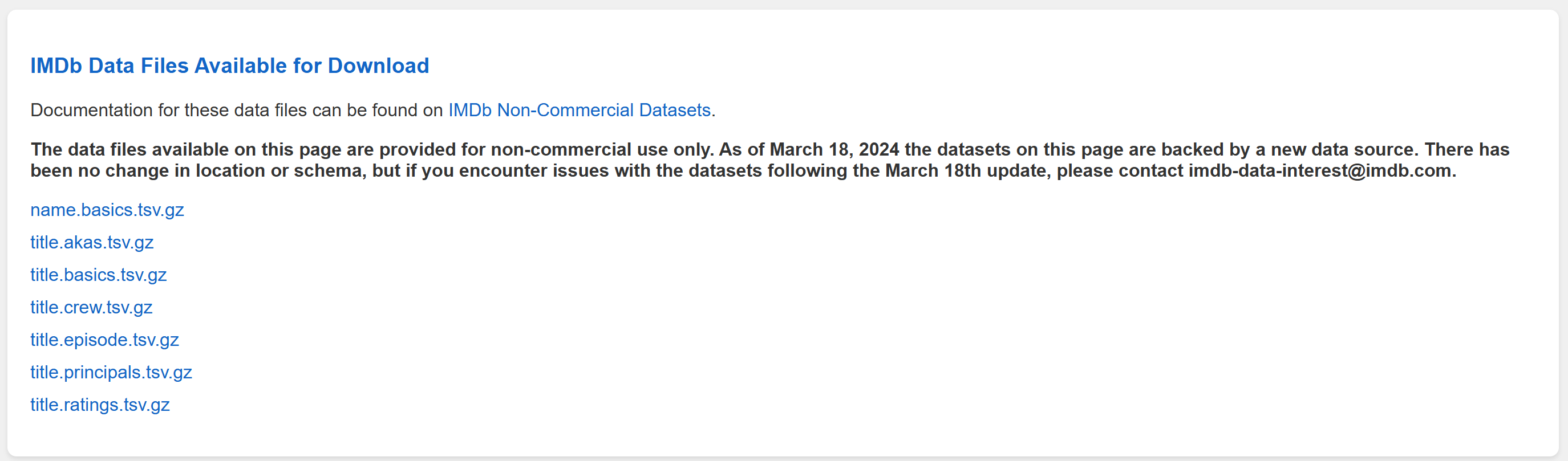


Above Is a clear display of the main home page on Movie Matcher’s website incorporate



# C. Developing the coded solution (“The development story”)

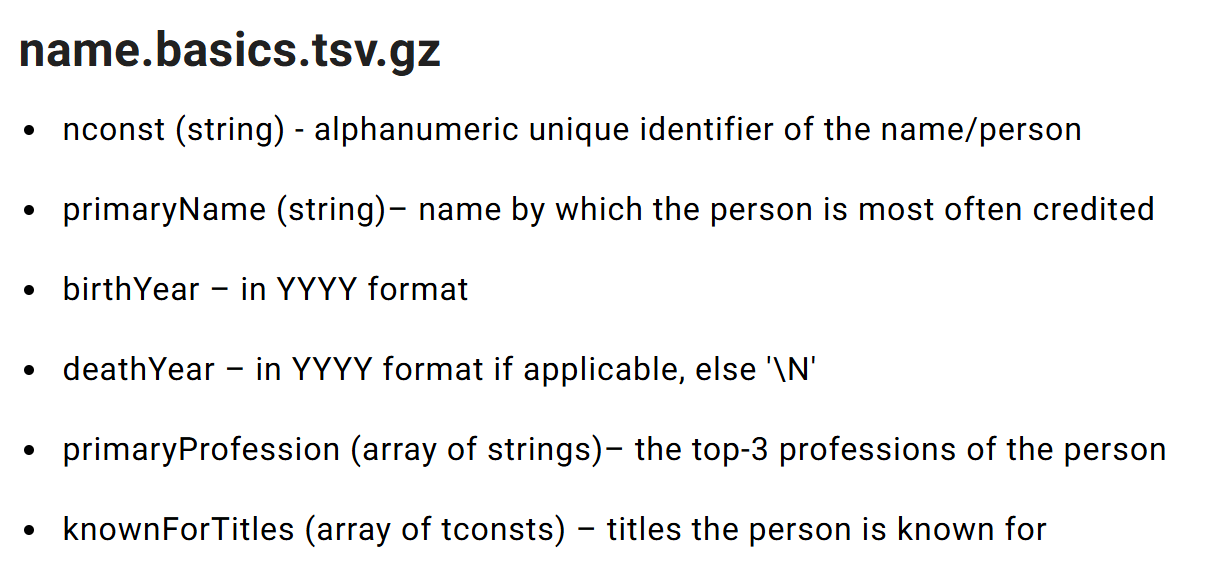
A key requirement of my Movie Matcher program is having access to a reliable and comprehensive movie database to power all of its matching features. To lay the groundwork for the project, one of my first steps was to obtain the official IMDb dataset, which is provided as a compressed gzip file. Downloading this dataset early in the development process ensures that the program has a solid foundation of accurate movie information to work with.

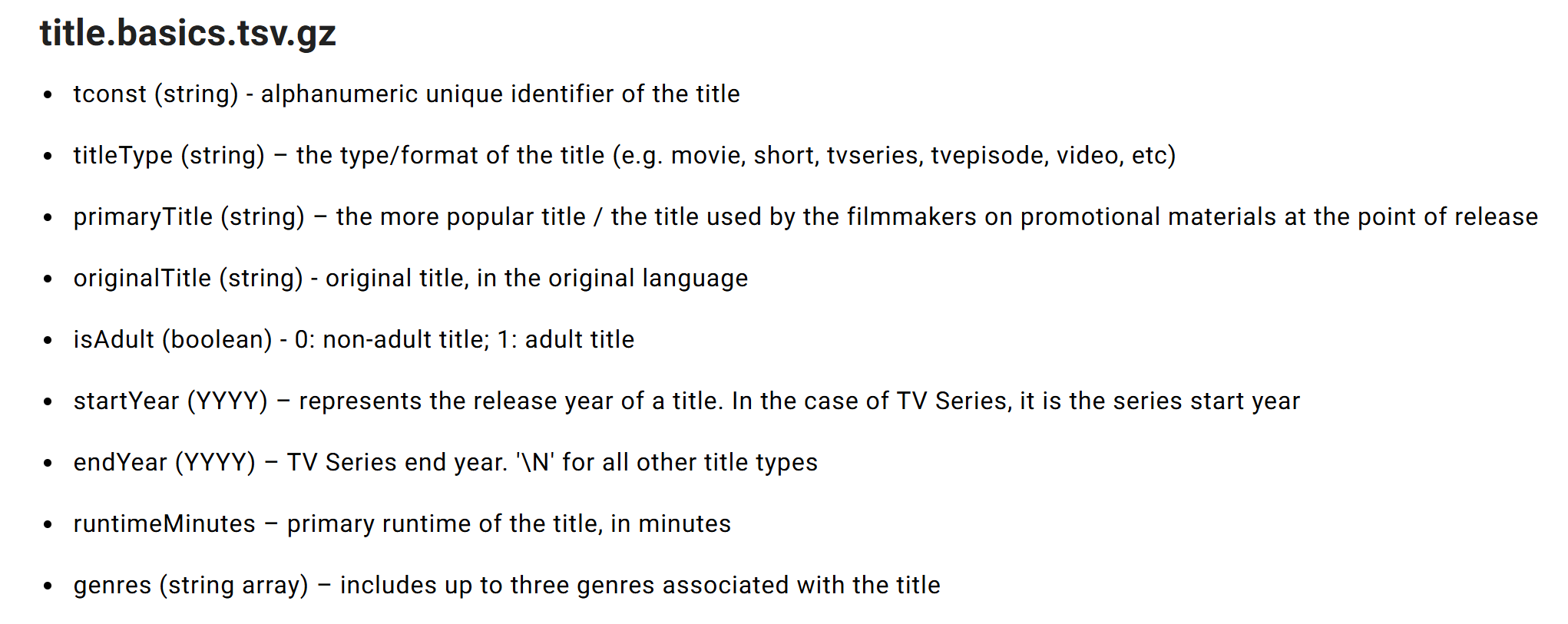


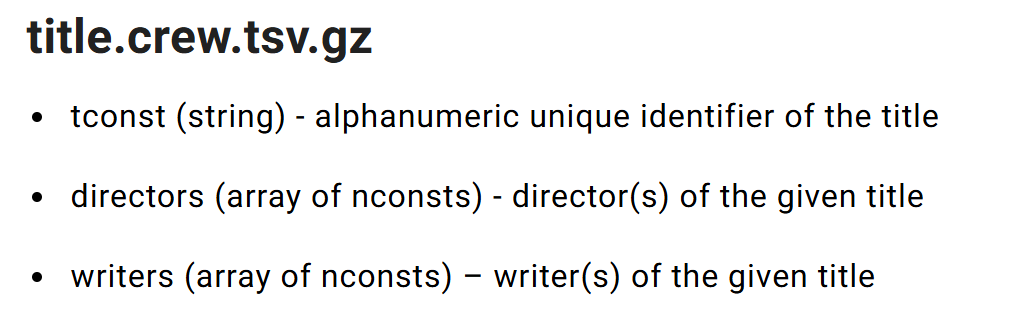
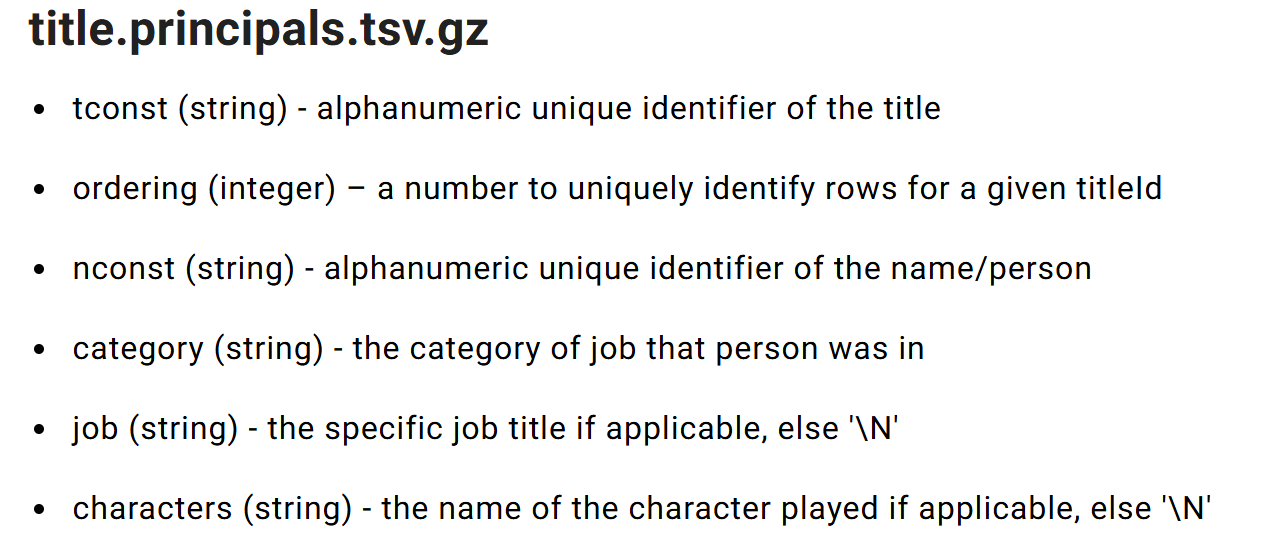
I downloaded the following files:

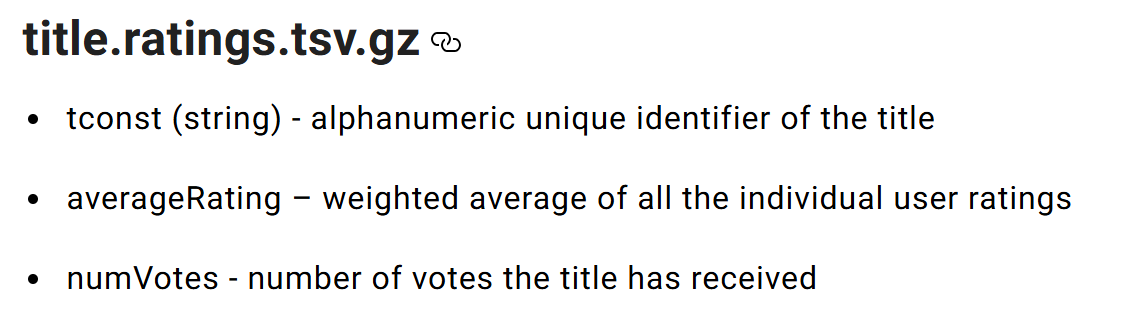
 These files are vital for the program’s recommendation feature. By analysing the information contained in the files, such as genres, release years, cast, and user‑rated attributes, the system can determine how closely a film aligns with a user’s preferences.

### Information on the following files:







A screen shot of a computer

AI-generated content may be incorrect.

A black screen with colorful text

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

**Error checking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test no.** | **Description** | **Test data** | **Expected outcome** | **Actual Outcome** |
| 1.1 | Sign up/creating a username | min | Expected 6 characters for username |  |
| 1.2 |  |  |  |  |
| 1.3 |  |  |  |  |
| 1.4 |  |  |  |  |
| 1.5 |  |  |  |  |

A computer screen shot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

# D. Evaluation

<See H446-03 Project Advice Booklet for help and guidance of what must go here.>

# Project Appendixes

Insert as many project appendixes as you need for your project.

These might include, but are not limited to:

* Complete Code Listing (ESSENTIAL)
* Interview Transcripts
* Meeting notes
* Observation notes or questionnaires