MoviePsychic
System Requirements Specification

Documentation by Brandon Addison, Matthew Rodriguez, and Jonah Schwab

1. Vision

1.1 Background

In the digital age, the online film community has exploded in popularity, with so-called "cinephiles" spilling much digital ink discussing and sharing films, both popular and niche. For those with specific tastes and interests who have seen dozens or hundreds of films, it can become an issue to find new experiences that they are likely to enjoy. While there are ways to find new films, such as on blogs, forums, and websites like IMDb and Letterboxd, there remains to be seen a website solely dedicated to recommending films to users based on an input of liked and disliked films.

1.2 Vision Statement

We aim to create a website that allows users to give previously watched films star ratings and be immediately presented with films that are similar and also well-regarded by users with similar taste. This would be a fast, immediate process without the distractions of similar websites, focused in its purpose: to recommend films the user should watch next. Furthermore, we aim to provide more fine-tuned recommendations as users rate more films, reflecting each user's specific taste and encouraging continued use among the user base.

1.3 References

The Movie Database, *The Movie Database API version 3*, https://developer.themoviedb.org

1.4 Features

FE-1: Search for films from a database
FE-2: Log in with a username and password and remember the user
FE-3: Change password
FE-4: Delete account
FE-5: Give films a star rating from 1-10 (or 1-5, with half-stars)
FE-6: Edit, alter, and delete previously given ratings
FE-7: Compare films with "positive" star ratings (i.e. above the halfway point) with not-yet-seen films based on shared characteristics
FE-8: Compare the catalog of rated films of multiple users to find high levels of similarity (i.e. >50% of shared films are matches in terms of positive and negative star ratings)
FE-9: Present the user with the ten films from the database that best match both methods of comparison

1.5 Assumptions and Dependencies

AS-1: Users will generally adhere to a combination of a rating spectrum and rating binary, wherein films they feel positively about will *nearly always* be rated above the midpoint and vice

versa. This etiquette will be shared with them in the “About” section (see User Documentation) before the user begins to rate films.
AS-2: Users will only rate films they have previously watched and give accurate and honest ratings to the best of their ability.
DE-1: The operation of MoviePsychic is dependent on The Movie Database API, from which the site’s movie catalog and all accompanying information about each movie within said catalog is being derived.

2. Scope and Limitations

2.1 Scope

Feature	**Demo 1 (10/22)**	**Demo 2 (11/12)**	**Final (12/5)**	**Priority**
FE-1	Database implemented	Searching implemented		High
FE-2	Not implemented	Basics implemented if time permits	Full implementation if time permits	Low
FE-3	“”			Low
FE-4	“”			Low
FE-5	Implemented			High
FE-6	Implemented			High
FE-7	Implemented			High
FE-8	Draft or unfinished algorithm in source code	Full implementation		Medium
FE-9	Implemented			High

2.2 Limitations

LI-1: Only films in the database used will be available for use with the various features
LI-2: Algorithm will be highly deterministic, leading it to feeling less personalized for users
LI-3: Algorithm will be heavily weighted against niche films that have not been seen by many users
LI-4: Due to the limited time, programming experience, and access to resources, some features may have to be stripped down or cut as the project goes on

LI-5: To cut down on runtime issues, the database of films may need to be cut down to the top X% most-popular films in our dataset, using parameters given by TMDB API

2. 3 User Classes and Characteristics

Basic User (placeholder name) - All users are expected to be of the same users class. Anyone who uses MoviePsychic will have access to all of the features that it offers.

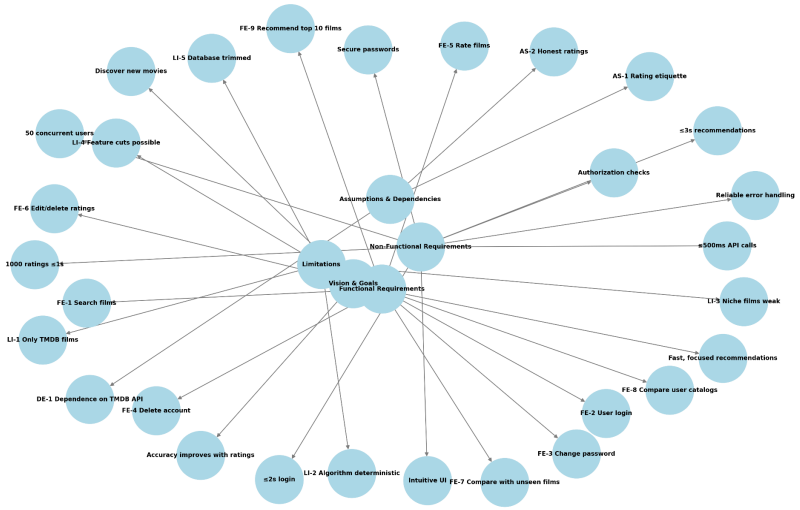

Fig. 1: Context Diagram

2.4 Operating Environment

OE-1: MoviePsychic will be accessible via any modern internet browser on a device with a stable connection

2.5 Design and Implementation Constraints

Programming Languages: The core of the recommendation engine and backend must be developed using Python. The frontend must be developed using JavaScript with the React library.
Frameworks: We are required to use Django for the backend API and React for the frontend.
Database: We must use PostgreSQL as the relational database for storing user data, movie metadata, and ratings.
Version Control: All code must be managed collaboratively using Git and hosted on GitHub.
Dataset: We must use a publicly available movie dataset, such as the MovieLens dataset, for our recommendation engine.

2.6 User Documentation

UD-1: The site will have an “About” page that provides instructions on how to utilize the features provided by the site. This will include information about creating and managing an account, searching for and rating films, and editing or deleting previously given ratings.

3. System Features

3.1 Search for films from a database

3.1.1 Description and Priority
A user will be able to search for a given movie by said movie's title. That search will return any movie(s) within the database with title(s) matching or containing the search input. This feature is of high priority.

3.1.2 Stimulus/Response Sequence
Stimulus: A user does a search via a search bar
Response: The site displays all movies within the database with titles matching or containing the text entered into the search bar by the user

3.1.3 Functional Requirements
- Search: The system will search in the database for films
- Search.Text: The system will compare the text in the query against tags and titles
- Search.Results: The system will display results ordered by popularity and how close they match the query

3.2 Log in with username and password

3.2.1 Description and Priority
A user will be able to create an account using a unique username and an accompanying password. The site will then store this user information and tie the user's ratings directly to their user profile. This feature is of low priority

3.2.2 Stimulus/Response Sequence
Stimulus: A user clicks "create account" or "log in"
Response: The site will prompt the user to enter credentials to create a new or find an existing account.

3.2.3 Functional Requirements
- Password: The system will be able to maintain and keep track of unique credentials pertaining to each user, securely
- Password.Validate: The system will be able to check a username and password to verify the credentials and grant access to user-specific pages and information
- Password.Validate.Correct: The system will grant the user access to the sensitive pages and information if correct credentials are provided

- Password.Validate.Incorrect: The system will revoke access and prompt an error message explaining what went wrong if incorrect credentials are provided
- Password.Storage: The system will be able to encrypt credentials securely so user data is not compromised by third-parties, nefarious and otherwise

3.3 Change password

3.3.1 Description and Priority
Once a user has created an account and logged in, they will be able to change their password. This feature is of low priority

3.3.2 Stimulus/Response Sequence
Stimulus: User changes password
Response: Account information is now updated on the server and will now only response to updated credentials.

3.3.3 Functional Requirements
User information should update in no more than 3 seconds.
List should adjust itself and recalculate placements.

3.4 Delete account

3.4.1 Description and Priority
Once a user has created an account and logged in, they will be able to delete their account. This feature is of low priority

3.4.2 Stimulus/Response Sequence
Stimulus: User clicks “delete account” on an account page
Response: All data associated with said account is deleted from MoviePsychic servers

3.4.3 Functional Requirements
- Delete: The system will be able to delete accounts
- Delete.Confirm: The system will ask the user to confirm their decision to prevent accidental deletion

3.5 Give films a star rating

3.5.1 Description and Priority
Once a user has selected a film from their search, they will be able to assign a rating to thefilm on a 1-5 star scale. This feature is of high priority

3.5.2 Stimulus/Response Sequence

Stimulus: User selects a film from an available list. Assigns rating
Response: Movie's priority in the list is adjusted according for the user's recommendation.

3.5.3 Functional Requirements

The recommendation list should start generating in no more than 3 seconds after a user rates a film.

3.6 Edit or delete previously given ratings

3.6.1 Description and Priority

If a user has previously rated a given film, they will be able to edit the rating given to it or delete that rating entirely. This feature is of high priority

3.6.2 Stimulus/Response Sequence

Stimulus: User decides to edit rating
Response: Movie recommendation list changes according to new ratings.

3.6.3 Functional Requirements

- Rate: Users can input ratings, which the system will update and save both as part of their profile and the overall average rating for a film

3.7 Compare films with positive ratings to unwatched films in the database

3.7.1 Description and Priority

When a user rates films positively, the system will recommend similar films after comparing. High priority.

3.7.2 Stimulus/Response Sequence

Stimulus: User rates movies, some of them positively, some negatively
Response: System recommends movies similar to positive-rated ones with few things in common with negative-rated ones.

3.7.3 Functional Requirements

User can search for movies or take them as provided and rate them. List should start generating to be presented to user in under 10 seconds.

3.8 Compare user catalogs

3.8.1 Description and Priority
The system will be able to meaningfully compare users' catalogs of rated films to find similarities, and recommend unwatched movies to paired users. Medium priority.

3.8.2 Stimulus/Response Sequence
Stimulus: User rates a certain number of films
Response: System automatically identifies users with similar film ratings and recommends unwatched films rated positively

3.8.3 Functional Requirements
- Catalog: System stores as part of user data the liked and disliked films a user has watched
- Catalog.Compare: System can look for users that have shared watched films and compare how similar their tastes are
- Catalog.Recommend: System can recommend films if a sufficient level of similarity is reached (~>75% similarity)

3.9 Recommendation

3.9.1 Description and Priority
System will give a user 10 films to watch next. High priority.

3.9.2 Stimulus/Response Sequence
Stimulus: User rates at minimum five films
Response: System lists 10 films to watch on the home page

3.9.3 Functional Requirements
- Recommend: System can display the titles and posters of ten films that best match the criteria of 3.7 and 3.8
- Recommend.Info: When films are hovered over, system will display the average rating of the film and why it was recommended (e.g. similar users liked it, in common with liked films, etc.)

4. Other Requirements

4.1 External Interface Requirements
UI-1: The MoviePsychic pages will be laid out with a clear, readable font and feature an easily accessible page select bar at the top of the home page
UI-2: Navigation will be accomplished with mouse alone, with the exception of using the keyboard to search

4.2 Performance Requirements

Can generate a recommendation list in 3 seconds. Will be verified with a unit test that measures response time of recommendation output.
User authentication and login shall be completed in less than 2 seconds. RESTFUL API will allows movie ratings and submissions within 500 milliseconds.
Testing:API load testing tools can be used to send a high volume of requests and measure the average and peak response times for each endpoint
The database shall be able to store and retrieve a user's complete rating history (up to 1000 ratings) within 1 second.
A single movie's metadata (title, genre, etc.) shall be retrieved from the database in less than 100 milliseconds
Testing:SQL queries can be timed directly against the database to measure the execution time. Database indexing will be a key factor in meeting these requirements.
The system shall be able to support up to 50 concurrent users without a noticeable degradation in performance (e.g., response times increasing by more than 20%).
Testing:A load testing framework can be used to simulate a large number of users simultaneously performing common tasks like logging in, browsing movies, and requesting recommendations.
The user interface shall load and render the main dashboard within **3 seconds** on a modern web browser.
Testing:Frontend performance tools, such as Google's Lighthouse, can be used to measure key metrics like First Contentful Paint (FCP) and Time to Interactive (TTI).

4.3 Security Requirements

The system shall use a secure (e.g., Argon2, scrypt, or a well-configured Bcrypt) to store user passwords in the database. Passwords shall never be stored in plaintext.The system shall implement a robust session management mechanism that assigns a unique, cryptographically secure session token upon successful login.

Testing:The system shall enforce an authorization check for all user-specific data access. A user shall not be able to view, modify, or delete another user's profile information or ratings.
All user-provided input, including usernames, emails, and movie ratings, shall be validated on both the client-side and server-side to conform to expected data types and formats.

4.4 Other Software Quality Attributes

Intuitive:A new user should be able to register and receive their first set of recommendations within 5 minutes without needing a tutorial.The interface will be clean, responsive, and provide clear feedback to the user on their actions (e.g., a visual confirmation when they rate a movie).
Reliable: The system must gracefully handle all potential errors (e.g., database connection failures, external API errors) and provide informative, user-friendly messages rather than crashing.

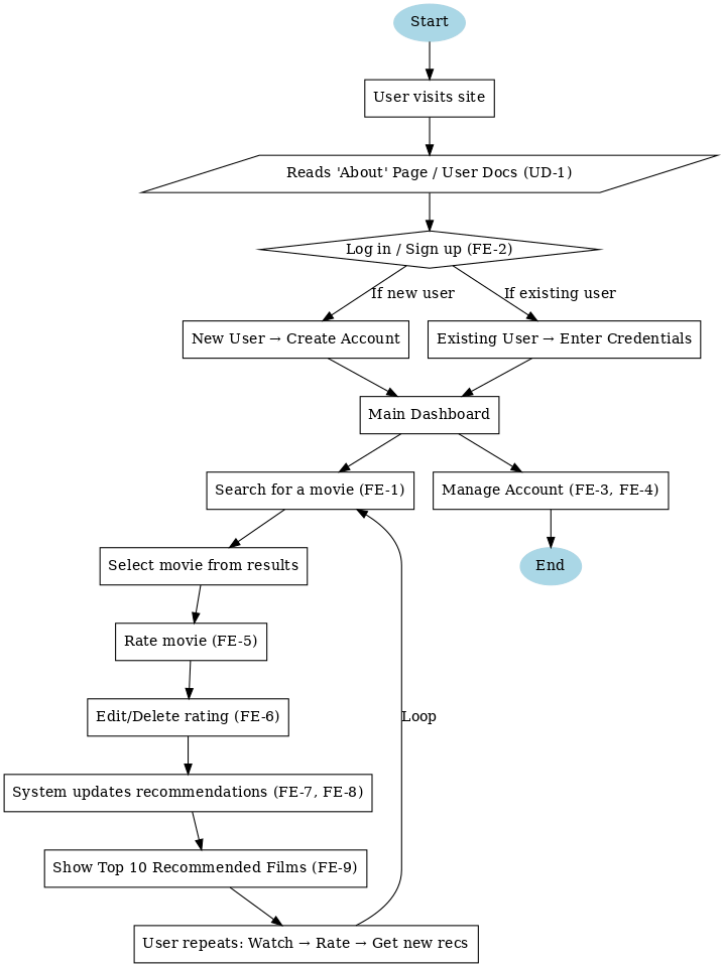

Fig. 2: System Use Flowchart