**Christopher Straka**

**Student ID:** 011551615

**Project Proposal:** Movie Recommender System

**Problem Summary:** Our streaming service is experiencing a decline in user engagement due to poor content discovery. There is no existing recommendation system, so our users struggle to find movies that match their preferences. This causes people to leave our platform and not recommend it as much, which leads to lower subscription counts as well. Finally, there is poor visualization and data insight into the ratings that users have left us. This leads to a limited understanding of customers’ preferences and hinders our ability to excel as a business.

**In Scope:** The items that are in scope include the recommendation system and the UI for showing those recommendations. Data visualization and documentation are also important items in scope for this project.

**Out of Scope:** Excluded from this project's scope is the implementation of a review or comment system for users to provide feedback on movies. While seemingly related, this feature requires a different set of functionalities and moderation resources, which diverges from the project’s primary aim of developing a recommendation algorithm based on existing ratings. Also out of scope is the personalization of user interfaces based on individual user preferences. While this can enhance user experience, it's a separate domain from the algorithmic recommendation engine we're focusing on.

**Application Benefits:** The recommender system will enhance user experience by suggesting movies that are more to their liking. This targeted approach aims to increase user engagement, retention, and subscription rates by making content discovery more personalized and efficient. The user will be given multiple suggestions as well so they can have multiple options to pick from. There will also be data visualizations to show key insights into the ratings data, which helps the company understand the ratings data.

**Application Description:** The application will be a desktop app that accepts a user ID and returns all the movies the user has rated and the movies that are recommended to a user. The movies will be in “imdb ID” format, where the user can click on an id to open that movie’s page on imdb. The desktop app will have a python model providing suggestions in the background. The model will use collaborative filtering, a technique that recommends movies based on what ratings the user has given and on what ratings similar users have given. The application will also have images of data visualization to show key insights of the ratings data collected by users.

**Data Description:** The system will utilize a dataset from Kaggle, specifically two of the dataset’s CSV files, ratings\_small.csv and links\_small.csv. The former has two nominal ids (user\_id and movie\_id), a timestamp, and a real number rating between 0 and 5. The latter csv file has three nominal ids, (movie\_id, imdb\_id, and tmbd\_id). The independent variables are user\_id, movie\_id, and rating from the ratings\_small.csv file. The dependent variable is the predicted movie rating for each movie, personalized for each user, based on their ratings and the ratings of other users who rate movies similarly. One limitation of this data is that movies with higher ratings tend to get more reviews. This makes it harder to recommend lesser-known movies, and so there is more bias for more popular films. Another limitation is that this is the smaller version of the dataset. This could limit the diversity and variety of movie recommendations.

**Objective:** Develop a recommender system that improves user engagement by providing personalized movie suggestions. Visualize the ratings data and grab key insights for stakeholders.

**Hypothesis:** If we implement a recommender system using collaborative filtering, then we will see an increase in user engagement as it provides more relevant movie suggestions based on user ratings and the ratings of similar users. If we visualize the ratings data in different ways, we will gain valuable insights that can help the business later.

**Methodology:** I've chosen the waterfall methodology for its clear, sequential structure, which aligns well with the defined and stable requirements of my movie recommendation system project. This approach ensures that each phase, from data collection to development and testing, is thoroughly completed before moving to the next. It also simplifies project management and planning, as each phase has predictable timelines and deliverables. Additionally, Waterfall's emphasis on documentation at each stage will be crucial for maintaining and auditing the system post-development.

**Funding Requirements:** Total estimated cost is $1,600, calculated at 32 hours of development at $50 per hour working 9:00 – 5:00. This aligns with the letter of transmittal and covers all stages of development. The dataset is public domain and the software (Tauri, Python, Rust, JS, Surprise, Jupyter) follows permissive licenses, leading to no other costs outside of labour.

**Impact on Stakeholders:** Users benefit from improved personalized movie recommendations, enhancing their viewing experience. Content creators have increased viewership of their content due to more effective recommendations and higher user engagement. The marketing team now has better insights into targeted advertising and promotional campaigns. The developers now have opportunities for professional growth in data science. Investors now see a potential increase in platform profitability and customer base.

**Data Precautions:** User ids are anonymous and have no personal information tied to them (name, age, payment information, etc.). Ratings for each user id are omitted to better prevent people from building profiles on users. HIPAA, FERPA and PCI DSS don't apply in this case, because I don't use health data, educational data, or credit card information. The data has largely been cleaned and prepared by the Kaggle dataset creator. The data is also public domain, meaning I’m not working with any private information that could get someone in trouble.

**Developer’s Expertise:** The developer possesses advanced expertise in JavaScript, Rust, and Python, honed through developing several high-traffic web applications and complex data processing systems. They also have experience in UI development and data science building ML full stack applications in the cloud. This background, coupled with academic training in computer science, makes him well-equipped to execute this project effectively and efficiently.