

# CPS 842 Movie Recommender System Project Report

## How to Run the Program?

Live Website: <https://movie-recommender-program.herokuapp.com/>

Or

```
pip install -r requirements.txt
```

```
python main.py
```

Your browser should automatically open, if not on your browser open: <http://localhost:5000>

## Detailed Design of System

### Dataset

There are 100,000 ratings ranked from (1-5) from 943 users on 1682 movies.

Each user has rated at least 20 movies.

In u.data, I use the following fields (user\_id | movie\_id | rating).

In u.item, I use the following fields (movie\_id | movie\_title).

<https://grouplens.org/datasets/movielens/100k/>

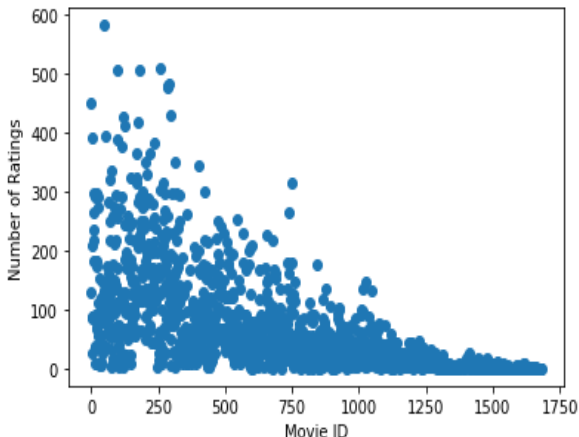
I generated the user similarity matrix using generate\_user\_similarity\_matrix.py which reads data from u.data and u.item and outputs the results to user\_similarity\_matrix.txt. Please open user\_similarity\_matrix.txt using Notepad++ for proper formatting.

### Data Analysis (From 'Data Analysis.ipynb')

Median number of ratings for movies: 27

Average number of ratings for movies: 59

data.png



The top ten most rated movies are:

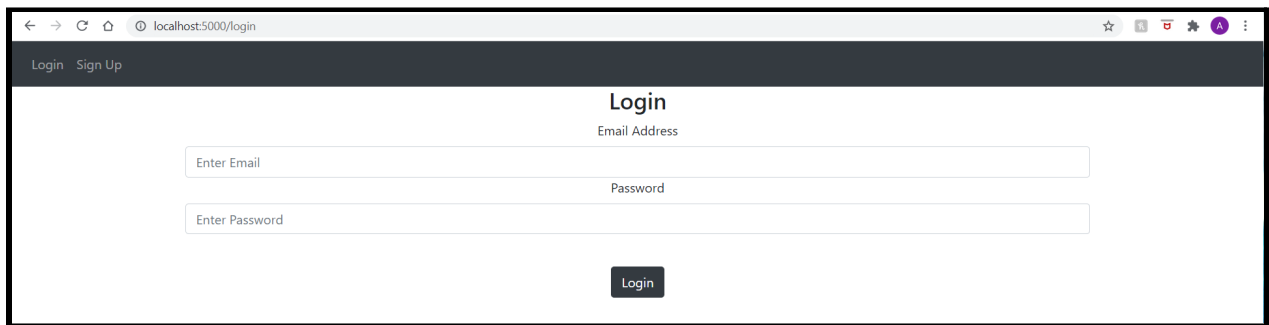
1. Star Wars (1977)	583 ratings
2. Contact (1997)	509 ratings
3. Fargo (1996)	508 ratings
4. Return of the Jedi (1983)	507 ratings
5. Liar Liar (1997)	485 ratings
6. English Patient, The (1996)	481 ratings
7. Scream (1996)	478 ratings
8. Toy Story (1995)	452 ratings
9. Air Force One (1997)	431 ratings
10. Independence Day (ID4) (1996)	429 ratings

## **Frontend**

Includes basic user management functions that allow users to sign up for an account, login to their account, and logout. I have implemented error handling to these functions.

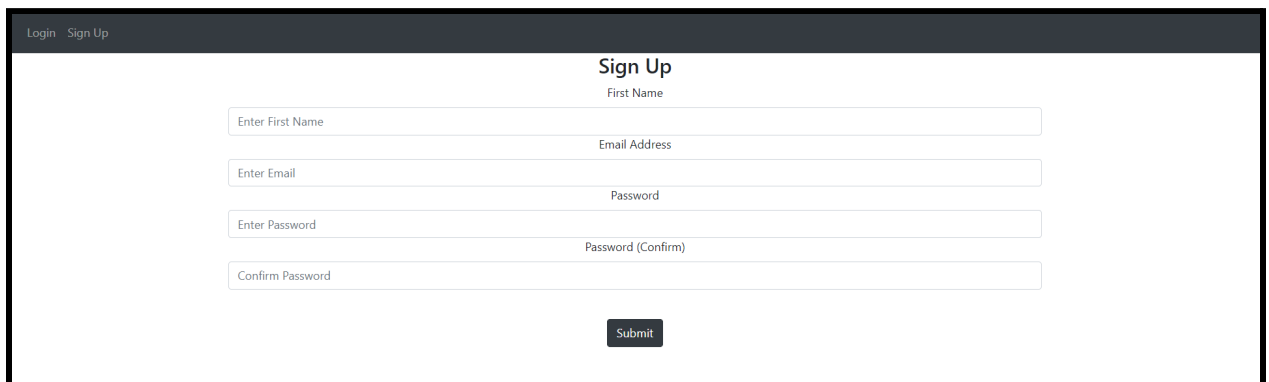
When users sign up for a new account, their first name and email characters must be greater than a certain number. Also, their passwords must match, and a user cannot use an email that already exists in the database. Passwords are hashed using the SHA-256 algorithm.

Users cannot access movies.html until they login to their account, which means they have to sign up for an account to login.



The screenshot shows a web browser window with the address bar displaying 'localhost:5000/login'. The page has a dark header with 'Login' and 'Sign Up' links. The main content area is titled 'Login' and contains two input fields: 'Email Address' with a placeholder 'Enter Email' and 'Password' with a placeholder 'Enter Password'. A 'Login' button is positioned below the fields.

**login.html**



The screenshot shows a web browser window with the address bar displaying 'localhost:5000/login'. The page has a dark header with 'Login' and 'Sign Up' links. The main content area is titled 'Sign Up' and contains five input fields: 'First Name' with a placeholder 'Enter First Name', 'Email Address' with a placeholder 'Enter Email', 'Password' with a placeholder 'Enter Password', 'Password (Confirm)' with a placeholder 'Enter Password', and 'Confirm Password' with a placeholder 'Confirm Password'. A 'Submit' button is positioned below the fields.

**sign\_up.html**

On movies.html, a user can rate up to 20 movies.

Division by zero errors can happen when a user gives the same rating for all movies handled in the backend by assigning a zero value so the program does not crash.

If a user rates the same movie more than once, the last rating will be used in the program's backend.

If a user enters no input, no recommendations will be given.

If a user enters a movie with no corresponding rating or vice versa, an error message will be displayed asking them to enter the movie/rating correctly.

[Movies](#) [Logout](#)

Logged in successfully!

## Movies

Movie Options	Your Ratings
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen
Please Pick a Movie	Haven't Seen

**movies.html**

Please Pick a Movie

Haven't Seen

Please Pick a Movie

Haven't Seen

Click Here for Recommended Movies

Clear Results

Your Inputs:

You gave Toy Story (1995) a score of 5

You gave GoldenEye (1995) a score of 3

You gave Four Rooms (1995) a score of 4

You gave Get Shorty (1995) a score of 3

You gave Copycat (1995) a score of 3

You gave Shanghai Triad (Yao a yao dao waipo qiao) (1995) a score of 5

You gave Twelve Monkeys (1995) a score of 4

You gave Babe (1995) a score of 1

You gave Dead Man Walking (1995) a score of 5

You gave Richard III (1995) a score of 3

Your Results:

Star Wars (1977) is recommended for you because based on your inputs we believe you would give it a score of 4.0 out of 5

Fargo (1996) is recommended for you because based on your inputs we believe you would give it a score of 3.08 out of 5

Raiders of the Lost Ark (1981) is recommended for you because based on your inputs we believe you would give it a score of 2.84 out of 5

**movies.html**

## **Backend**

The backend ([website/collaborative\\_filter.py](#)) uses the user-based collaborative filtering approach to predict the rating of movies the user has not rated based on items previously rated by other users.

For the user similarity calculation, I used the Pearson correlation coefficient which gives a result from -1 to +1 when comparing the movie ratings of each user in the dataset to the user's input. Where -1 is a perfect negative correlation between two users' ratings, 0 indicates no relationship between the two users' ratings, and +1 is a perfect positive correlation between two users' ratings.

I considered only similarity calculations greater than or equal to 0.5 in the weighted-sum aggregate function for this project.

For each movie the user has not watched, I calculated the weighted-sum which predicts the movie's rating between 0 to 5 using the similarity scores and movie ratings of the users considered. I returned the top 3 movies with the highest rating.

## **Functions in the System**

In ([website/collaborative\\_filter.py](#))

```
def main(selected_movies, ratings, movie):
```

The driver code which calls each of the following functions in sequential order. Then the top three scores and their corresponding movies are returned and displayed on movies.html if recommendations have been found; in other words, similarity scores > 0.5 were calculated.

```
def create_matrix(selected_movies, ratings, movies):
```

Returns a matrix of movie scores from u.data where each row is a user id, and their rating for each movie represents the columns.

```
def calculate_average(matrix, user_ratings):
```

Calculates and returns the average rating between the users' input and each user in the dataset for movies both users have rated.

```
def calculate_similarity_scores(matrix, user_ratings, averages):
```

Calculates and returns the Pearson correlation coefficient (a score between -1 and +1) for the user and each user in the dataset using their average ratings and the movie scores both users have rated.

```
def calculate_weighted_sum_rating(matrix, similarity_scores):
```

Calculates and returns the predicted movie scores the user has not rated using similarity scores > 0.5.

## **References:**

For the user interface: <https://github.com/techwithtim/Flask-Web-App-Tutorial>

For the dataset: <https://grouplens.org/datasets/movielens/100k/>