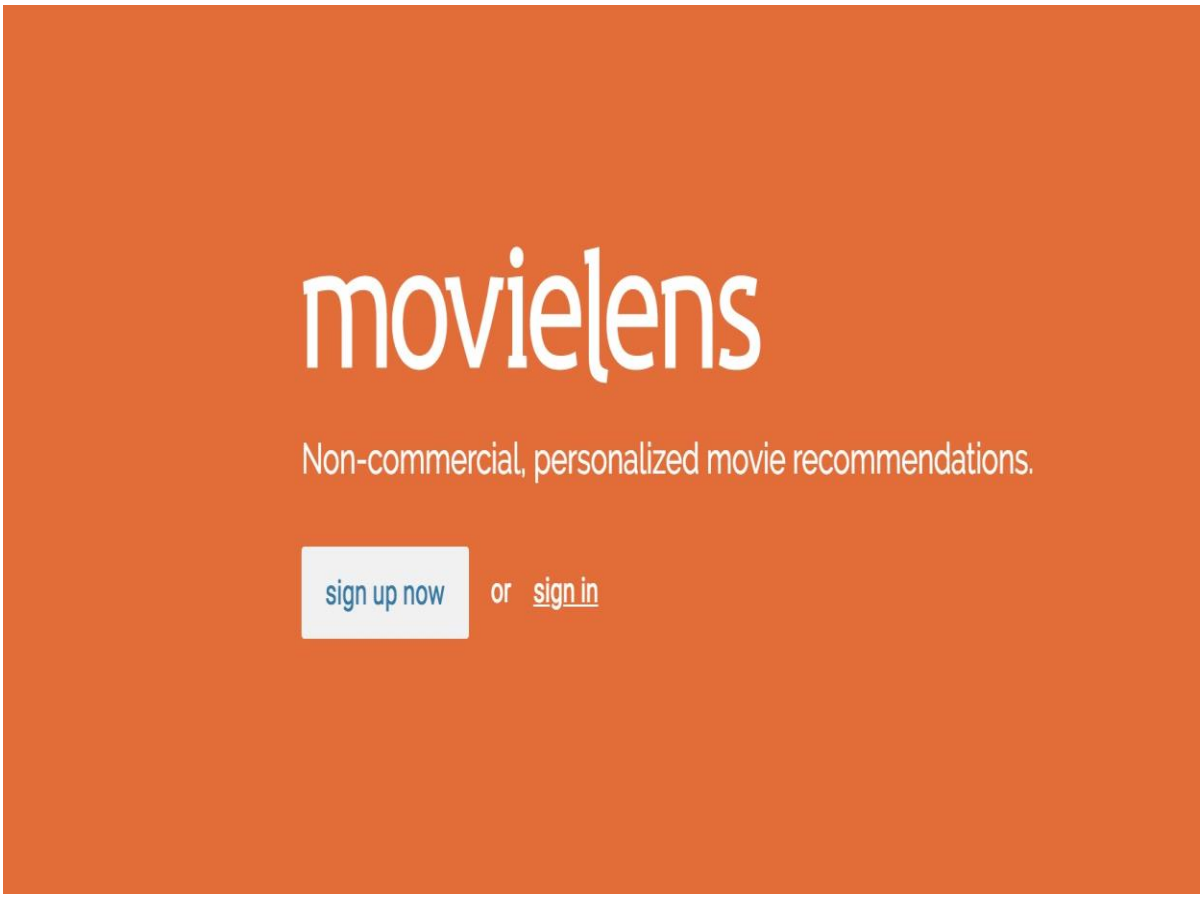



# MOVIE RECOMMENDATION SYSTEM

TEAM 6:
RISHABH JAIN
GUNJAN LALWANI
ANN SAJEE

# OVERVIEW:

The Movie Recommendation System offers generalized recommendations on movie popularity and (sometimes) genre. The basic idea behind this recommender is that movies that are more popular and more critically acclaimed will have a higher probability of being liked by the average audience.

# GOALS:

To provide some helpful information - things that we wish we could know - to the general people.
Recommending movies based on genre.
To offer a simple web application to let people see the recommendation of movies with the help of machine learning

# USE CASES:

**Case 1: Recommend the most popular items**

A simple approach could be to recommend the items which are liked by most number of users. This is a blazing fast and dirty approach and thus has a major drawback. The thing is, there is no personalization involved with this approach. Basically, the most popular items would be same for each user since popularity is defined on the entire user pool. So, everybody will see the same results. It sounds like, 'a website recommends you some pool of movies just because it's been liked by other users and doesn't care if you are even interested in watching or not'.

**Case 2: Using a classifier to make recommendation**

We already know lots of classification algorithms. Classifiers are parametric solutions, so we just need to define some parameters (features) of the user and the item. The outcome can be 1 if the user likes it or 0 otherwise. This might work out in some cases because of following advantages:

- Incorporates personalization
- It can work even if the user's history is short or not available

**Case 3: Recommendation Algorithms**

These are the special class of algorithms which are tailor-made for solving the recommendation problem. There are typically two types of algorithms – Content Based and Collaborative Filtering.

1. **Content based algorithms:**
    - **Idea:** If you like an item then you will also like a "similar" item
    - Based on similarity of the items being recommended
2. **Collaborative filtering algorithms:**
    - **Idea:** If a person A likes item 1, 2, 3 and B like 2,3,4 then they have similar interests and A should like item 4 and B should like item 1.
    - This algorithm is entirely based on the past behavior and not on the context. This makes it one of the most commonly used algorithm as it is not dependent on any additional information.
    - Further, there are several types of collaborative filtering algorithms:
        1. **User-User Collaborative filtering:** Here we find look alike customers (based on similarity) and recommend movies which first customer's look alike has chosen in past. This algorithm is very effective but takes a lot of time and resources. It requires to compute every customer pair information which takes time. Therefore, for big base platforms, this algorithm is hard to implement without a very strong parallelizable system.
        2. **Item-Item Collaborative filtering:** It is quite like previous algorithm, but instead of finding customer look alike, we try finding movie choice alike.

**Case 4: Evaluating Recommendation Engines**

For evaluating recommendation engines, we can use the concept of precision-recall. Let me define them in terms of recommendations.

- **Recall:**
    - What ratio of movies that a user likes were recommended.
    - If a user likes to say 5 movies and the recommendation decided to show 3 of them, then the recall is 0.6
- **Precision**
    - Out of all the recommended movies, how many the user actually liked?
    - If 5 movies were recommended to the user out of which he liked say 4 of them, then precision is 0.8

# DATA:

**The MovieLens Dataset:**

We will be using the MovieLens dataset for this purpose. It has been collected by the GroupLens Research Project at the University of Minnesota. MovieLens 100K dataset can be downloaded from here. It consists of:

- **100,000 ratings** (1-5) from 943 users on 1682 movies.
- Each user has rated **at least 20 movies.**
- Simple demographic info for the users (age, gender, occupation, zip)
- Genre information of movies

| | |
|---|---|
| Action | Film-Noir |
| Adventure | Horror |
| Animation | Musical |
| Children's | Mystery |
| Comedy | Romance |
| Crime | Sci-Fi |
| Documentary | Thriller |
| Drama | War |
| Fantasy | Western |

# PROCESS OUTLINE:

1. Data Preprocessing
    - Data Cleaning
    - handling missing values
2. Exploratory Data Analysis
3. Study of Supervised approaches and select the best model for prediction
5. Design of a pipeline and system to implement this approach and discussion on the system's capabilities
6. Deploy the Model on AWS
7. Build a flask web application to demonstrate the prediction and recommendation results.

## MILESTONES:

| Timeframe | Delivery |
|---|---|
| Day 1-3 | Data Preprocessing and Exploratory Data Analysis |
| Day 4-6 | Model Building, Training, Selection |
| Day 7-8 | Deployment of models on cloud and build web application |
| Day 9-10 | System integration and documentation |

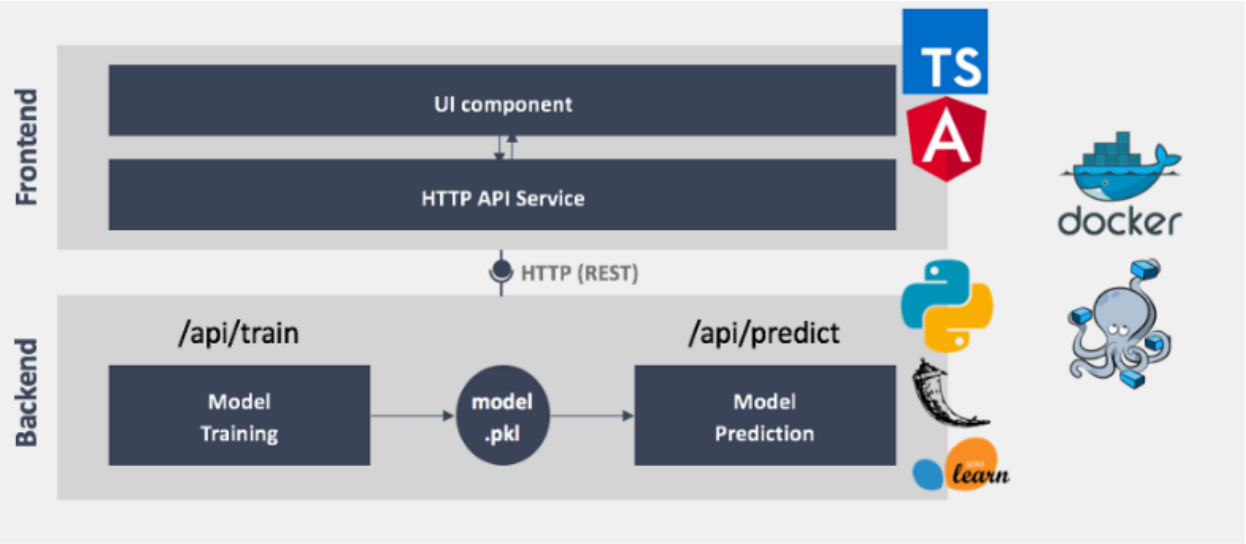
## PERSONNAS:

1. **End Users:**

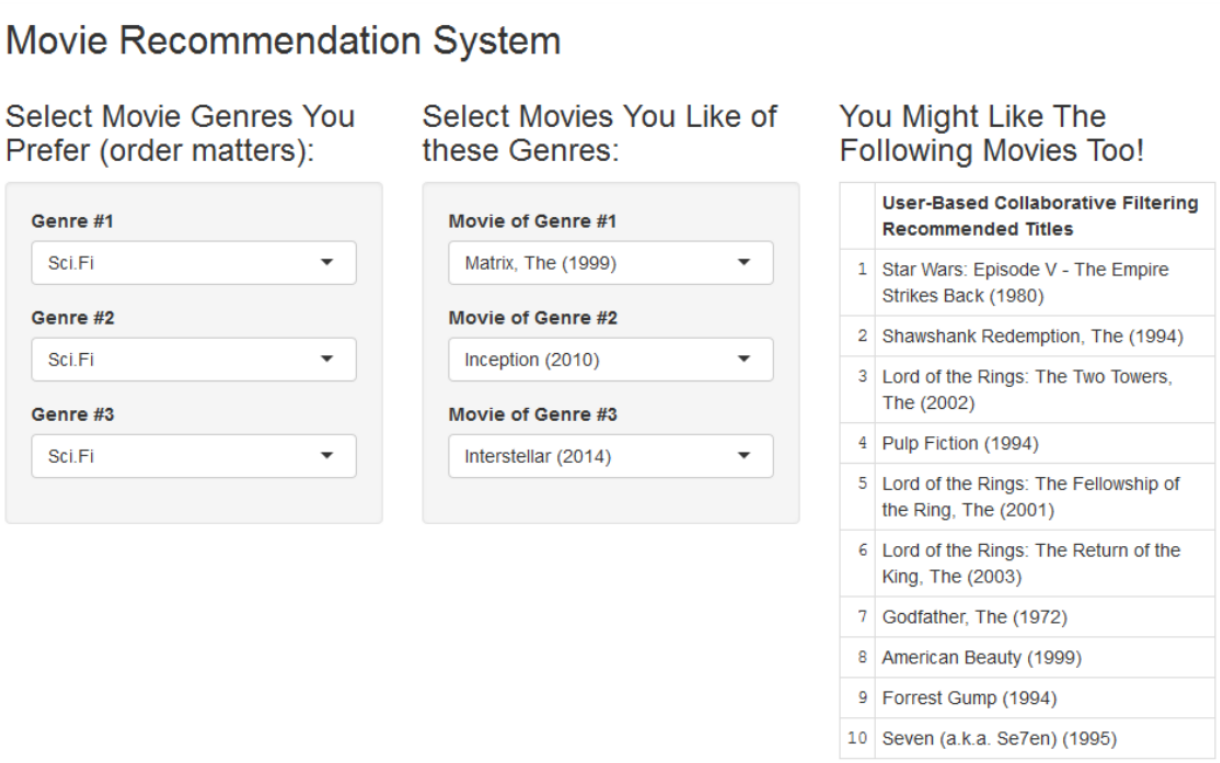
- To check for Recommendations of best movies

# DEPLOYEMENT DETAILS:

1. Language: Python
2. Pipeline: sklearn
3. Container: Docker
4. Cloud Tools/Platforms: AWS (Amazon Web Services) EC2



## USER INTERFACE DESIGN PLAN




## REFERENCES AND SOURCES:


https://grouplens.org/datasets/movielens/latest/
https://pdfs.semanticscholar.org/e13e/b41de769f124b3c91771167fb7b01bc85559.pdf
http://ieeexplore.ieee.org/document/5575081/