**1. Overview**

The Movie Matchmaker app is a hybrid movie recommendation system that suggests movies to users based on:

1. **Collaborative Filtering (User Behavior):** What movies other users with similar tastes liked.
2. **Content-Based Filtering (Movie Features):** Similarity in movie attributes such as genres.
3. **Popularity & Ratings Boost:** Gives extra weight to highly-rated and frequently-watched movies.

The system is designed for **non-technical users**, with intuitive sliders and checkboxes to control recommendation preferences.

**2. Inputs & Outputs**

**Inputs:**

* Selected movie (title) by the user.
* User preference for weighting between CF and content-based recommendations (alpha).
* Checkbox for giving weight to popular/well-rated movies.
* Number of recommendations (n\_recs).

**Outputs:**

* List of recommended movies (title), sorted by a hybrid score.

Optional (future): Movie posters or thumbnails for each recommendation.

**3. Data Sources**

1. **Movies Dataset (movies.csv)**
   * Columns: movieId, title, genres
   * Example:

* Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
* Jumanji (1995),Adventure|Children|Fantasy

2.  **Ratings Dataset (ratings.csv)**

* Columns: userId, movieId, rating, timestamp
* Example:
* 1,1,4.0,964982703
* 1,3,4.0,964981247

**4. Technical Workflow**

**Step 1: Load Data**

* Load movies and ratings datasets using pandas.
* Cache datasets with Streamlit’s @st.cache\_data to avoid repeated reading.

**Step 2: Collaborative Filtering (CF)**

* Use **SVD (Singular Value Decomposition)** from surprise library.
* **Training Process:**
  1. Convert ratings dataframe into surprise.Dataset format.
  2. Split data into train/test sets (20% test).
  3. Train SVD on the train set → latent factors for movies (qi) and users (pu).
* **Similarity Computation:**
  1. Cosine similarity between movie latent factor vectors (qi).
  2. This captures “movies liked by similar users.”

**Step 3: Content-Based Filtering (CB)**

* Use **movie genres** as features.
* Apply CountVectorizer to convert genres into a sparse feature matrix.
* Compute **cosine similarity** between genre vectors.
* Captures “movies that are thematically similar.”

**Step 4: Popularity & Rating Adjustment**

* Compute:
  + Total number of ratings per movie (count)
  + Average rating per movie (avg\_rating)
* Normalize both values to [0,1].
* Optional boost: Add weight to popular and highly-rated movies.

**Step 5: Hybrid Score Computation**

For each candidate movie:

Hybrid Score=α⋅CF Score+(1−α)⋅CB Score\text{Hybrid Score} = \alpha \cdot \text{CF Score} + (1-\alpha) \cdot \text{CB Score}Hybrid Score=α⋅CF Score+(1−α)⋅CB Score

* If popularity boost is enabled:

Final Score=0.8⋅Hybrid Score+0.2⋅(Popularity + Avg Rating)\text{Final Score} = 0.8 \cdot \text{Hybrid Score} + 0.2 \cdot (\text{Popularity + Avg Rating})Final Score=0.8⋅Hybrid Score+0.2⋅(Popularity + Avg Rating)

* Normalize all scores between 0–1 to ensure fair weighting.
* Exclude the selected movie from recommendations.

**Step 6: Recommendation Selection**

* Sort movies by **final hybrid score** in descending order.
* Return top n\_recs titles as recommendations.

**5. User Interface Design (Streamlit)**

| **UI Component** | **Description** |
| --- | --- |
| **Title & Intro Text** | Explains purpose in simple language. |
| **Movie Dropdown** | Select movie you like. |
| **Slider: Balance** | Controls CF vs CB weighting (alpha). |
| **Checkbox: Popularity Boost** | Apply extra weight to popular/well-rated movies. |
| **Number Input** | Number of recommendations to display. |
| **Button: Show Recommendations** | Trigger recommendation generation. |
| **Results List** | Display recommended movies with plain titles (optional future: posters). |

**6. Functional Flow Diagram (Simplified)**

**[User selects a movie]**

**|**

**v**

**+--------------------------+**

**| Compute CF Score |**

**+--------------------------+**

**|**

**v**

**+--------------------------+**

**| Compute CB Score |**

**+--------------------------+**

**|**

**v**

**+-----------------------------------+**

**| Combine Scores (Hybrid) |**

**+------------------------------------+**

**|**

**v**

**+---------------------------------+**

**| Apply Popularity Boost |**

**+---------------------------------+**

**|**

**v**

**+----------------------+**

**| Sort & Select N |**

**+----------------------+**

**|**

**v**

**[Display Recommendations]**

**7. Performance & Caching**

* Datasets, SVD model, and similarity matrices are cached with Streamlit decorators (@st.cache\_data / @st.cache\_resource) to speed up response.
* Precomputing content similarity and popularity stats avoids redundant computation.