Movie Recommendation System

Link : https://movie-rec-system-2025-gtezhdhhbna4b8bu.switzerlandnorth-01.azurewebsites.net/

Overview

This repository contains a complete **Movie Recommendation System** built with Python.
It includes:

- An **ETL pipeline** for preparing movie datasets
- A **content-based recommender engine**
- A **Flask web application**
- Helper scripts for working with TMDb data
- Documentation and developer guides

How to run?

1. Create and activate a virtual environment (PowerShell)

```
python -m venv venv
.\venv\Scripts\Activate.ps1
```

2. Install dependencies

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

3. Run the web application

```
python MovieRecommendationSystem/app.py
```

Then open:

👉 http://127.0.0.1:5000/

Project Structure

```
MovieRecommendationSystem/
│
├── app.py                 # Flask application entry point
├── routes/                # Route handlers (movies, user, auth)
├── static/                # CSS, JS, images
├── templates/             # HTML (Jinja) templates
│
├── database/
│   ├── models.py          # SQLAlchemy models
│   └── db.py              # DB initialization helpers
│
├── recommender/
│   ├── engine.py          # Main recommendation logic
│   └── preprocess.py      # Feature building, similarity matrices
│
├── etl_pipeline/          # Scripts for dataset preparation
│   ├── extract.py
│   ├── transform.py
│   ├── load.py
│   └── run_pipeline.py
│
├── tmdb_5000_movies.csv
├── tmdb_5000_credits.csv
└── large_movies.csv       # Example processed dataset
```

Installation Guide (Windows / PowerShell)

Requirements

- Python **3.8+**
- Git

Steps

1. Clone the repository

```
git clone <repo-url>
cd Movie-Recommendation-System-main
```

2. Create and activate a virtual environment

```
python -m venv venv
.\venv\Scripts\Activate.ps1
```

3. Install Python dependencies

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

4. Prepare the data (optional)

The repo already includes:

- `tmdb_5000_movies.csv`

- `tmdb_5000_credits.csv`
- `large_movies.csv`

To rebuild datasets:

```
python etl_pipeline/run_pipeline.py
```

5. Run the application

```
python MovieRecommendationSystem/app.py
```

Usage

Run the web server

```
.\venv\Scripts\Activate.ps1
python MovieRecommendationSystem/app.py
```

Access UI at:

👉 http://127.0.0.1:5000/

Common Developer Tasks

Rebuild ETL dataset

```
python etl_pipeline/run_pipeline.py
```

Generate large dataset

```
python generate_large_dataset.py
```

Check missing poster images

```
python verify_posters.py
```

Use recommendation engine programmatically

```
from recommender.engine import Recommender
r = Recommender()
print(r.recommend_for_movie(movie_id=1234, top_n=10))
```

API

Route	Method	Description
`/`	GET	Home page (popular / recommended movies)
`/movie/<int:movie_id>`	GET	Movie details
`/recommendations`	GET	Recommendations page
`/auth/login`	POST	User login
`/auth/signup`	POST	User signup
`/user/watchlist`	GET	Show user watchlist

Backend details:

- Templates under `templates/`
- Static assets under `static/`
- Recommendation engine in `recommender/`

Database Schema

User Table

Field	Type	Description
`id`	INTEGER (PK)	User ID (primary key)
`username`	STRING (UNIQUE)	Unique username
`email`	STRING	User email address
`password_hash`	STRING	Hashed password
`created_at`	DATETIME	Account creation timestamp

Movie Table

Field	Type	Description
`id`	INTEGER (PK)	TMDb movie ID
`title`	VARCHAR	Movie title

`overview`	TEXT	Summary / plot
`genres`	VARCHAR / JSON	Genre list
`keywords`	VARCHAR / JSON	Tags/keywords
`cast`	VARCHAR / JSON	List of actors
`crew`	VARCHAR / JSON	Directors, writers…
`poster_path`	VARCHAR	URL/path to poster
`popularity`	FLOAT	TMDb popularity score
`vote_average`	FLOAT	Average rating
`metadata`	JSON (optional)	Extra info

Watchlist Table

Field	Type	Description
`id`	INTEGER (PK)	Entry ID
`user_id`	INTEGER (FK)	User who added the movie
`movie_id`	INTEGER (FK)	Movie added
`added_at`	DATETIME	Timestamp

Architecture Overview

Components

1. Flask Web Application

- Registers routes
- Renders templates
- Serves UI

2. ETL Pipeline

- **extract.py:** Load raw CSV/API data
- **transform.py:** Clean and enrich data
- **load.py:** Save processed dataset

3. Recommender Engine

- Feature extraction
- Similarity computation
- Recommendation generation

4. Database Layer

- SQLAlchemy models
- User + Watchlist persistence

Data Flow

1. Raw TMDb CSV files or API responses are collected.
2. ETL pipeline processes them into `large_movies.csv`.
3. Recommender preprocesses the dataset.
4. Flask app queries the recommender for recommendations.