PHASE-2

Project Title: Delivering personalized movie recommendations with an AI driven matchmaking system

Student Name: [SIVARANJINI N]

Registered Number :[421623244052]

Institution: [Mailam Engineering College]

Department: [Computer science and Business system]

Date Of Submission: [10th may 2025]

Github Link: https://github.com/Thulasimathi26/Data-Science.git

1. Problem Statement

"With too many movies available online, users often struggle to find films that truly match their taste and mood. Existing recommendation systems are usually based only on viewing history or ratings, lacking real personalization. This project aims to build an AI-driven matchmaking system that gives personalized movie recommendations by understanding the user's preferences, mood, and behavior, making suggestions more relevant and enjoyable".

2. Project Objectives

1. To analyze user preferences based on watch history, ratings, and selected genres.

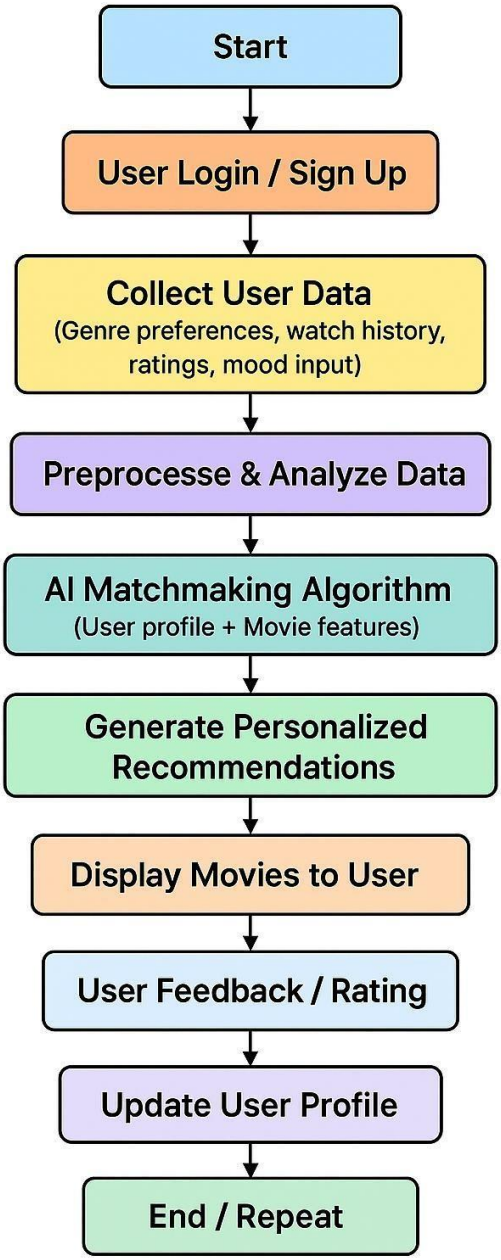
2. To integrate mood-based inputs (like happy, sad, bored, etc.) for emotional-level recommendations.

3. To design an AI matchmaking system that matches users with movies they are most likely to enjoy.

4. To improve recommendation accuracy using machine learning algorithms.

5. To enhance user satisfaction by providing relevant and personalized movie suggestions.

3.Flowchart of the Project Workflow



Data Description

Dataset Name: MovieLens Dataset (or custom user-movie interaction dataset)

- Source: MovieLens – provided by GroupLens Research
- Type of Data: Structured data (CSV format)
- Records and Features: Around 100,000+ ratings from 1,000 users on 1,700 movies

  Features: User ID, Movie ID, Rating, Timestamp, Genre
- Target Variable: Recommended Movie(s) for a specific user
- Static or Dynamic: Static (can be extended to dynamic with live user feedback)
- Attributes Covered:nUser preferences (ratings)

  Movie details (title, genres, release year)

  User behavior (watch history via timestamps)
- Dataset Link: https://grouplens.org/datasets/movielens/100k/

4.Data Preprocessing

1. Data Cleaning:

Remove duplicates and missing values (e.g., missing ratings or genres). Normalize text fields (like movie titles and genres).

2. Data Transformation:

Convert timestamps to readable date-time formats (optional).
Encode categorical variables like genres and languages using One-Hot Encoding or Multi-Label Encoding.
Normalize user ratings to a common scale (e.g., 0 to 1).

3. Feature Engineering:

Create user profiles by aggregating their genre preferences and average ratings.
Extract mood or sentiment features (if mood input is available).

Combine movie metadata (genre, language, rating) into feature vectors.

4. Data Splitting:Split data into training and testing sets (e.g., 80% training, 20% testing).
Use user-based or item-based splitting to preserve user behavior patterns.

5. Vectorization: Represent users and movies in vector form (e.g., using TF-IDF, embeddings, or matrix factorization).

Prepare similarity matrices if using collaborative filtering or content-based filtering.

5.Exploratory Data Analysis (EDA)

1. Basic Statistics:

Total number of users, movies, and ratings
Average rating given by users
Most frequently rated movies
Distribution of ratings (how many 5-star, 4-star, etc.)

2. User Behavior Analysis:

Number of ratings per user (active vs inactive users)
Preferred genres among different users
Most watched genre overall

3. Movie Insights:

Movies with the highest average ratings
Movies with the most number of ratings
Trend of movie releases over the years

4. Genre Analysis:

Popular genres based on user preferences
Genre combinations that are most common (e.g., Action + Adventure)

5. Rating Patterns Over Time:

Visualize rating activity over different time periods
Find peak user engagement times

6. Correlation Analysis:

Check correlation between genres and high ratings
Check if mood input aligns with certain genres or movies

7. Visualization Tools (used):

Bar plots, histograms for rating distribution
Pie charts for genre preferences
Heatmaps for correlation between features

6.Feature Engineering

User Profile Features:

Average Rating Given: Calculates how critical or generous a user is.
Preferred Genres: Count or percentage of genres the user frequently watches.
Watch Time Pattern: Time of day or days of the week the user is most active.
Mood Tags (Optional): Mood preferences tagged by user (e.g., happy → comedy).

2. Movie Features:

Genre Encoding: Use One-Hot or Multi-Label Encoding for genres.
Movie Popularity: Based on number of ratings and average rating.
Release Year: Helps filter outdated content if needed.
IMDB Ratings: External score as a quality indicator.

Length of Movie: Duration in minutes to match user patience level.

3. Interaction Features (User x Movie):

User-Movie Rating Matrix: Used in collaborative filtering.
Genre Match Score: How well a movie's genre matches a user's favorite. Mood Match Score: Matching mood tags with emotional tone of the movie.

7.Model Building

1. Choose Algorithm: Content-Based Filtering
Collaborative Filtering
Hybrid Model (combination of both)

2. Data Preparation: Use user-movie rating matrix
Encode features like genres and mood
Normalize ratings if needed

3. Train the Model: Use similarity measures (cosine, Pearson) for recommendations
For collaborative filtering, use algorithms like Matrix Factorization (SVD)
For content-based, match movie features with user profiles

4. Evaluate the Model:Use metrics like Precision, Recall, RMSE
Validate using train-test split or cross-validation.

5. Generate Recommendations: Predict top-N movies for each user.
Filter based on mood or genre if needed.

8:Visualization of Results & Model Insights

1. Rating Distribution: Bar chart showing how ratings are spread (1 to 5 stars).

2. Top Recommended Movies: Display top 5 or 10 movies for sample users.

3. Genre Popularity: Pie chart or bar chart showing most preferred genres.

4. User Activity: Heatmap of user engagement (time vsnumber of ratings).

5. Model Accuracy:Line graph for RMSE/MAE comparison Precision–Recall curve to show model performance.

6. Before vs After Personalization:Show difference in user satisfaction or match rate.

9.Tools and Technologies Used

1. Programming Languages:

Python – For AI/ML algorithms and backend logic.

JavaScript – For frontend (if building a web app).

2. Machine Learning & AI:

Scikit-learn – For basic ML models.
TensorFlow / PyTorch – For deep learning models (if you use neural networks).Surprise – A Python library for building recommendation systems.LightFM – For hybrid recommender system(collaborative + content-based).

3. Data Handling:

Pandas & NumPy – For data manipulation and analysis.

SQL / MongoDB – For storing user/movie data.

4. Recommendation Techniques:

Collaborative Filtering – Recommending based on user-user or item-item similarities.

Content-Based Filtering – Recommending based on movie genres, actors, etc.

Hybrid Systems – Combining both above.

5. Backend & API:
Flask / FastAPI – To create REST APIs for your AI system.Node.js (optional) – If JavaScript is used in backend.

6. Frontend:

React.js / HTML-CSS-JS – For creating the useinterface.

7. Deployment:
Heroku / Render / AWS / GCP – To host your application.Docker – For containerization

(optional but useful).

8. Additional Tools:
Jupyter Notebook – For experimenting with ML models.
Postman – For API testing.
GitHub – For version control and collaboration.

10.Team Members and Contributions

SIVARANJINI – Machine learning Engineer
PRATHIKSHA – Backend developer
SANTHINI – frontend developer
DEVASRI – Data analyst/ tester
ANBARASI – Testing and evaluation