Movie Ratings Dataset Exploration with Pivot Tables

The **Movie Rating App** is a Flask-based web application that allows users to explore movies, provide ratings, and share reviews. It integrates both backend and frontend components to deliver an interactive and user-friendly experience.

The system uses **CSV files** (`movies.csv`, `ratings.csv`, `users.csv`) as its data storage instead of a traditional database, making it lightweight and easy to deploy. Movie information such as title, genre, description, and poster image is displayed in a card-based layout. Users can select a movie and submit a rating (1–5 stars) along with a written review.

BY:Jangam Kavya

4GW23CI019

INDEX:

Introduction(Overview):

The **Movie Rating App** is a web-based platform developed using Python’s Flask framework. It provides an interactive way for users to explore movies, submit ratings, and write reviews, while also allowing them to view ratings and feedback from other users.

The main objective of this project is to create a **lightweight and user-friendly application** that demonstrates the integration of backend logic, frontend design, and data handling in a single system. Unlike complex database-driven applications, this project uses **CSV files** for storing movie, user, and rating data, making it simple, portable, and easy to maintain.

Key features include:

- A **home page** with a welcoming design and navigation options.
- A **movie listing page** displaying movies with their details, descriptions, genres, and posters.
- A **rating and review system**, where users can select a movie, give it a score (1–5), and optionally provide textual feedback.
- **Aggregated statistics**, such as the average rating per movie and the average rating given by each user.

The project highlights how Flask can be used to build a **full-stack web application** that connects data storage, server-side logic, and dynamic user interfaces.

System Requirements

1. Software Requirements

- **Operating System:** Windows 10 or higher / Linux / macOS
- **Python:** Version 3.8 or above
- **Libraries/Frameworks:**
  - Flask (for web application)
  - Pandas (for data analysis)
  - Jinja2 (template rendering, comes with Flask)
- **Web Browser:** Chrome, Firefox, or Edge (for accessing the portal)
- **Text Editor / IDE:** VS Code, PyCharm, or any Python IDE

2. Hardware Requirements

- **Processor:** Intel i3 or higher
- **RAM:** Minimum 4 GB
- **Storage:** Minimum 2 GB free space
- **Internet Connection:** Optional, only for downloading libraries or images

Technologies

Backend Framework: Flask – Python web framework for building the server-side of the application.

Programming Language: Python – For backend logic, data handling, and CSV operations.

Data Analysis: Pandas – For merging datasets, pivot table calculations, and exporting CSV reports.

Frontend Development: HTML, CSS, JavaScript – For creating interactive web pages, forms, and slideshow animations.

Template Engine: Jinja2 – Used with Flask to dynamically render HTML pages.

Data Storage: CSV files – Used to store movies, ratings, and user information.

Development Tools: VS Code or PyCharm – IDEs for writing and testing Python and web code.

Web Browser: Chrome, Firefox, or Edge – For accessing and testing the application interface.

Implementation

The **Movie Rating App** was implemented using Flask as the backend framework, Pandas for data processing, and HTML/CSS/JavaScript for the frontend interface. The implementation is divided into multiple modules:

1. Data Management

- The system uses **CSV files** instead of a traditional database.
- Three files are maintained:
  - `movies.csv`: Stores movie details (MovieID, Title, Genre, Description, PhotoURL).
  - `users.csv`: Stores user details (UserID, Name).
  - `ratings.csv`: Stores user ratings and reviews (UserID, MovieID, Rating, Review).
- Pandas is used to **read, update, and aggregate data** from these files dynamically.

2. Backend (Flask Application)

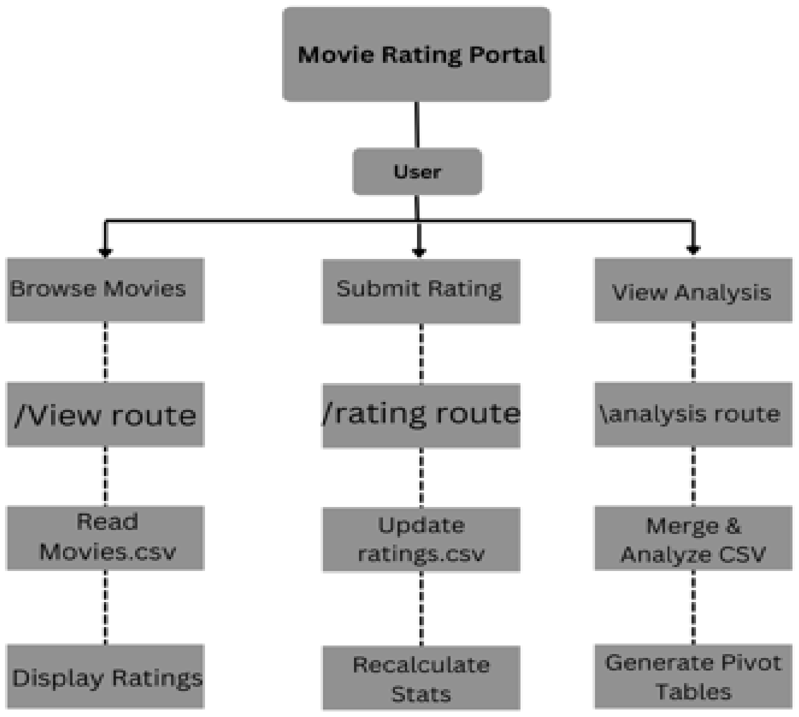
- Flask handles routing and data processing.
- Key routes include:
  - `/` → Displays the **home page** with navigation options.
  - `/movies` → Lists all movies with details, ratings, and modals for extended information.
  - `/ratings` → Allows users to **submit ratings and reviews** via a form and displays submitted feedback.
  - `/submit_rating` → Handles form submissions and updates the `ratings.csv` file.

   - `/stats` → Shows aggregated statistics such as **average rating per user** and **average rating per movie**.

3. Frontend (User Interface)

- Built using **HTML5, CSS3, and JavaScript**.
- Designed for a **modern and responsive look**, including:
   - **Home Page**: Background image with overlay and navigation buttons.
   - **Movie List Page**: Grid view with movie posters, titles, and descriptions. A modal popup displays extended details and ratings.
   - **Ratings Page**: A structured form to select user, movie, rating, and provide a review. Also displays a list of submitted reviews.
- Font Awesome icons and hover effects enhance the UI.

UML Diagram:


Movie Rating Portal
User
Browse Movies
Submit Rating
View Analysis
/View route
/rating route
\analysis route
Read Movies.csv
Update ratings.csv
Merge & Analyze CSV
Display Ratings
Recalculate Stats
Generate Pivot Tables

Front-end Desgin:

The **Movie Rating App** front-end is built with **HTML5, CSS3, and JavaScript**, supported by **Font Awesome icons** for visuals. It is designed to be clean, interactive, and user-friendly.

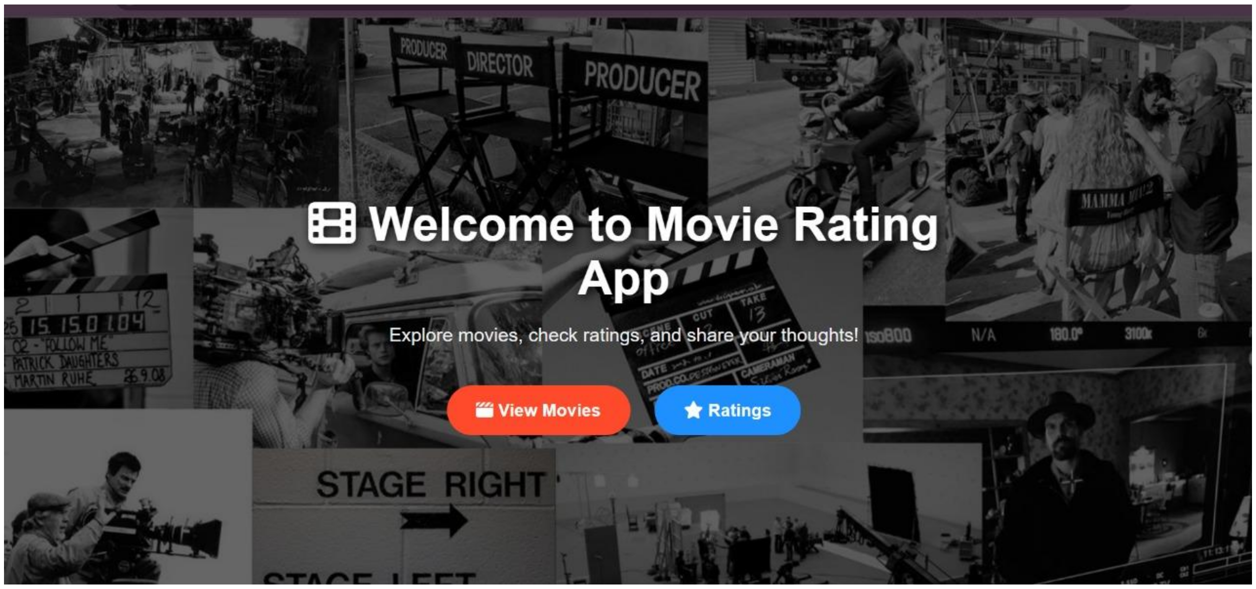
The **Home Page** serves as the entry point. It features a full-screen background image with a dark overlay, creating a cinematic effect. A heading with a movie icon and a welcome message is displayed. Two large buttons allow navigation to the **Movies** and **Ratings** pages. Hover effects make the buttons lively and interactive.

The **Movies Page** presents movies in a **grid-based card layout**. Each card displays the movie's poster, title, short description, and genre. On clicking a card, a **modal popup** opens, showing full details such as the complete description, poster, and the average rating with the number of reviews. This ensures both quick browsing and detailed viewing.

The **Ratings Page** contains a **submission form** where users can select their name, pick a movie, provide a rating (1–5), and write a review. A submit button styled in bright color posts the feedback. Below the form, reviews are displayed as cards showing the user's name, movie rated, rating given, and optional review text.

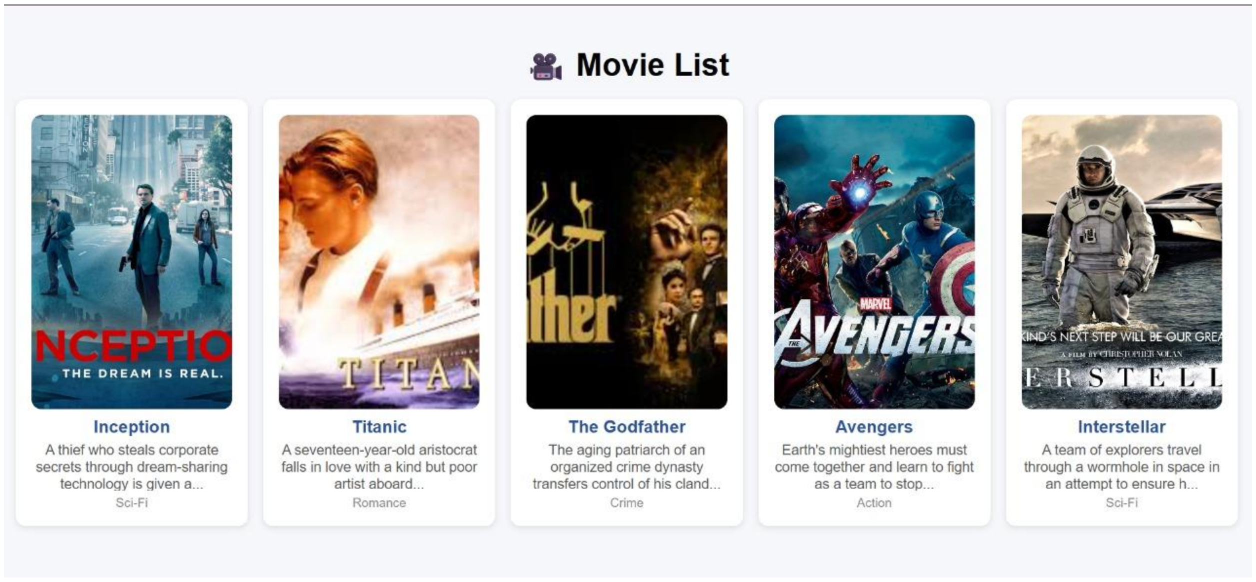
Styling choices include **modern fonts (Poppins)**, bright theme colors like orange and blue, smooth hover animations, and card shadows for depth. The design is also **responsive**, adjusting seamlessly to desktop, tablet, and mobile screens.

Overall, the front-end ensures **simplicity, responsiveness, and engagement**, offering users an intuitive experience.



Welcome to Movie Rating App

Explore movies, check ratings, and share your thoughts!



Movie List

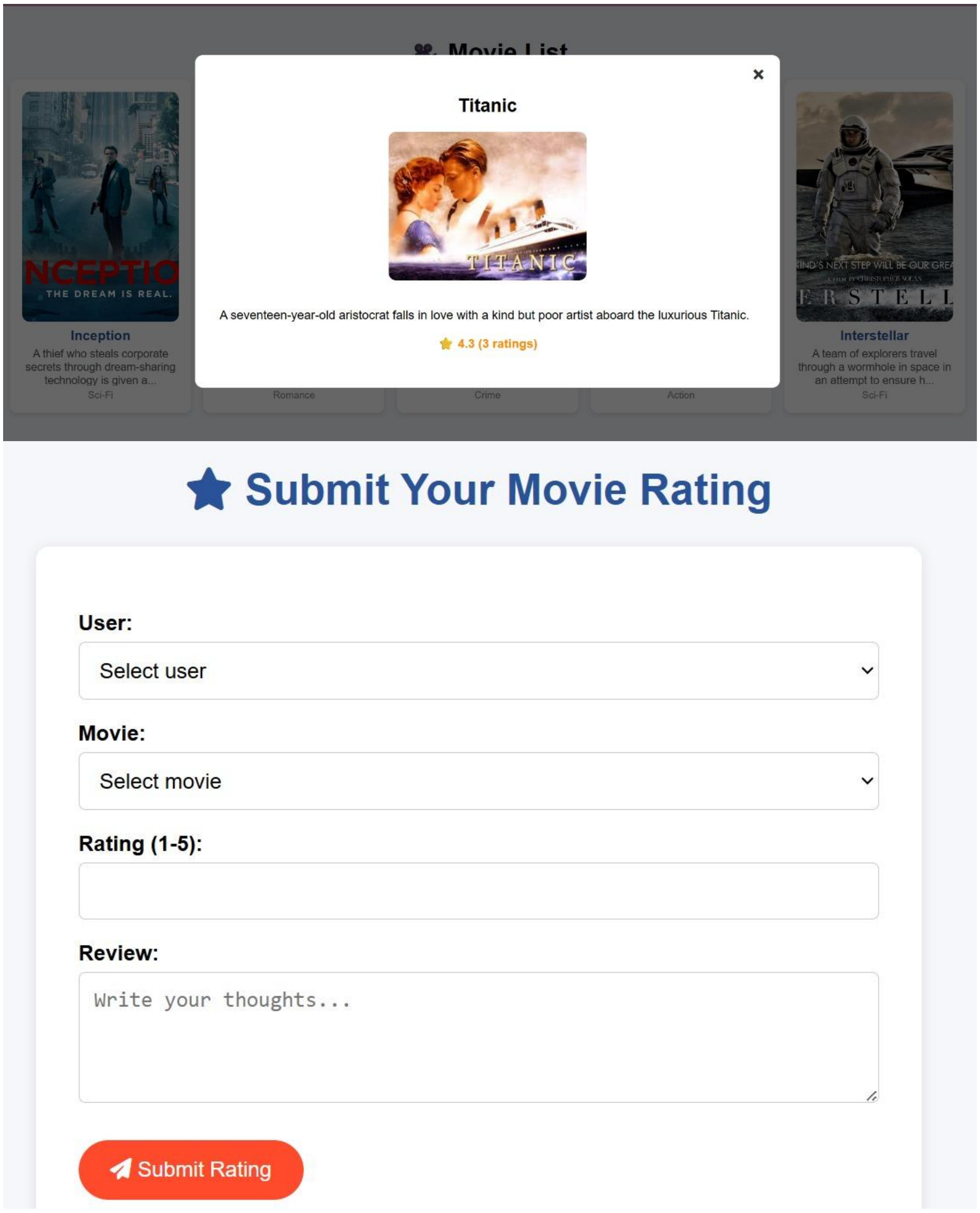
Inception
A thief who steals corporate secrets through dream-sharing technology is given a...
Sci-Fi

Titanic
A seventeen-year-old aristocrat falls in love with a kind but poor artist aboard...
Romance

The Godfather
The aging patriarch of an organized crime dynasty transfers control of his cland...
Crime

Avengers
Earth's mightiest heroes must come together and learn to fight as a team to stop...
Action

Interstellar
A team of explorers travel through a wormhole in space in an attempt to ensure h...
Sci-Fi


Movie List
×
Titanic
Inception
A thief who steals corporate secrets through dream-sharing technology is given a...
Sci-Fi
Romance
Crime
Action
Interstellar
A team of explorers travel through a wormhole in space in an attempt to ensure h...
Sci-Fi
Submit Rating

A seventeen-year-old aristocrat falls in love with a kind but poor artist aboard the luxurious Titanic.

4.3 (3 ratings)

Submit Your Movie Rating

User: Select user

Movie: Select movie

Rating (1-5):

Review: Write your thoughts...

-

Code:

app.py

```python
from flask import Flask, render_template, request, redirect, url_for
import pandas as pd
import os

app = Flask(__name__)

# ---------- Paths ----------
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
DATA_DIR = os.path.join(BASE_DIR, "data")
MOVIES_FILE = os.path.join(DATA_DIR, "movies.csv")
RATINGS_FILE = os.path.join(DATA_DIR, "ratings.csv")
USERS_FILE = os.path.join(DATA_DIR, "users.csv")

def load_dfs():
    """Read fresh copies each request so pages reflect new submissions
immediately."""
    movies_df = pd.read_csv(MOVIES_FILE)
    ratings_df = pd.read_csv(RATINGS_FILE)
    users_df = pd.read_csv(USERS_FILE)

    if "Review" not in ratings_df.columns:
        ratings_df["Review"] = ""
    ratings_df["Rating"] = pd.to_numeric(ratings_df["Rating"], errors="coerce")

    # Ensure Description and PhotoURL exist
    if "Description" not in movies_df.columns:
        movies_df["Description"] = ""
    if "PhotoURL" not in movies_df.columns:
        movies_df["PhotoURL"] = "/static/images/default.jpg"

    return movies_df, ratings_df, users_df

@app.route("/")
def home():
    return render_template("home.html")

@app.route("/movies")
def movies():
    movies_df, ratings_df, users_df = load_dfs()
    if not ratings_df.empty:
        agg = ratings_df.groupby("MovieID")["Rating"].agg(["mean",
"count"]).reset_index()
        merged = movies_df.merge(agg, on="MovieID", how="left").rename(
            columns={"mean": "AvgRating", "count": "NumRatings"}
        )
    else:
        merged = movies_df.copy()
        merged["AvgRating"] = None
        merged["NumRatings"] = 0

    return render_template("movies.html",
movies=merged.to_dict(orient="records"))

@app.route("/ratings")
def ratings():
    movies_df, ratings_df, users_df = load_dfs()
    merged = ratings_df.merge(movies_df, on="MovieID",
how="left").merge(users_df, on="UserID", how="left")
    return render_template(
        "ratings.html",
        movies=movies_df.to_dict(orient="records"),
        users=users_df.to_dict(orient="records"),
        reviews=merged.to_dict(orient="records"),
    )

@app.route("/submit_rating", methods=["POST"])
def submit_rating():
    _, ratings_df, _ = load_dfs()
    user_id = request.form["user"]
    movie_id = request.form["movie"]
    rating = request.form["rating"]
    review = request.form["review"]

    new_entry = {"UserID": user_id, "MovieID": movie_id, "Rating": rating,
"Review": review}
    ratings_df = pd.concat([ratings_df, pd.DataFrame([new_entry])],
ignore_index=True)
    ratings_df.to_csv(RATINGS_FILE, index=False)

    return redirect(url_for("ratings"))

@app.route("/stats")
def stats():
    movies_df, ratings_df, users_df = load_dfs()
    avg_by_user = ratings_df.groupby("UserID")["Rating"].mean().reset_index()
    avg_by_user = avg_by_user.merge(users_df, on="UserID", how="left")

    avg_by_movie = ratings_df.groupby("MovieID")["Rating"].mean().reset_index()
    avg_by_movie = avg_by_movie.merge(movies_df, on="MovieID", how="left")

    return render_template(
        "stats.html",
        avg_by_user=avg_by_user.to_dict(orient="records"),
        avg_by_movie=avg_by_movie.to_dict(orient="records"),
    )

if __name__ == "__main__":
    app.run(debug=True)
```

home.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Movie Rating App</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">
    <style>
        body {
            margin: 0;
            padding: 0;
            font-family: 'Poppins', sans-serif;
            background: url("{{ url_for('static',
filename='images/b49b6e21bac5ddac209d3428f825dd23.jpg') }}") no-repeat center
center/cover;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            color: white;
            position: relative;
        }
        .overlay {
            background: rgba(0,0,0,0.6);
            width: 100%;
            height: 100%;
            position: absolute;
            top: 0;
            left: 0;
            z-index: 1;
        }
        .content {
            z-index: 2;
            text-align: center;
            max-width: 700px;
            padding: 20px;
        }
        h1 {
            font-size: 3rem;
            margin-bottom: 20px;
            text-shadow: 2px 2px 10px black;
        }
        p {
            font-size: 1.2rem;
            margin-bottom: 30px;
            color: #f0f0f0;
        }
        .btn {
            display: inline-block;
            margin: 10px;
            padding: 15px 30px;
            border-radius: 30px;
            font-size: 1.1rem;
            font-weight: bold;
            cursor: pointer;
            text-decoration: none;
            color: white;
            transition: 0.3s;
        }
        .btn-primary { background: #ff4b2b; }
        .btn-secondary { background: #1e90ff; }
        .btn:hover {
            transform: scale(1.1);
            box-shadow: 0px 4px 15px rgba(0,0,0,0.4);
        }
    </style>
</head>
<body>
    <div class="overlay"></div>
    <div class="content">
        <h1><i class="fa-solid fa-film"></i> Welcome to Movie Rating App</h1>
        <p>Explore movies, check ratings, and share your thoughts!</p>

        <!-- Buttons -->
        <a href="{{ url_for('movies') }}" class="btn btn-primary"><i class="fa-
solid fa-clapperboard"></i> View Movies</a>
        <a href="{{ url_for('ratings') }}" class="btn btn-secondary"><i
class="fa-solid fa-star"></i> Ratings</a>
    </div>
</body>
</html>
```

movies.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Movies</title>
  <style>
    body { font-family: Arial, sans-serif; margin:0; background:#f5f7fa;
padding:1rem; }
    h1 { text-align: center; margin-bottom:1rem; }
    .grid { display: grid; grid-template-columns: repeat(auto-fit, minmax(220px,
1fr)); gap:1rem; }
    .card { background:white; border-radius:10px; padding:1rem; box-shadow:0 2px
6px rgba(0,0,0,0.1); transition:0.2s; cursor:pointer; display:flex; flex-
direction:column; align-items:center; text-align:center; }
    .card:hover { transform: scale(1.05); }
    .card img { width:100%; height:300px; object-fit:cover; border-radius:10px;
margin-bottom:0.5rem; }
    .title { font-weight:bold; margin-bottom:0.3rem; color:#2a5298; font-
size:1.1rem; }
    .description-snippet { font-size:0.9rem; color:#555; height:50px;
overflow:hidden; }
    .genre { font-size:0.8rem; color:#888; margin-top:0.3rem; }

    /* Modal */
    .modal { display:none; position:fixed; z-index:1000; left:0; top:0;
width:100%; height:100%; overflow:auto; background-color: rgba(0,0,0,0.6); }
```

```
    .modal-content { background-color:#fff; margin:5% auto; padding:2rem; border-
radius:10px; max-width:700px; text-align:center; position:relative; }
    .modal-content img { max-width:100%; border-radius:10px; margin-bottom:1rem;
}
    .close { position:absolute; top:10px; right:20px; font-size:28px; font-
weight:bold; cursor:pointer; color:#333; }
    .description { text-align:justify; max-height:300px; overflow-y:auto; }
    .rating { font-weight:bold; margin-top:10px; color:#ff8c00; }
  </style>
</head>
<body>
  <h1>🎥 Movie List</h1>
  <div class="grid">
    {% for movie in movies %}
    <div class="card" onclick="openModal(
        '{{ movie.Title|escape }}',
        '{{ movie.Description|escape }}',
        '{{ movie.PhotoURL|escape }}',
        {{ movie.AvgRating or 'null' }},
        {{ movie.NumRatings }}
    )">
      <img src="{{ movie.PhotoURL }}" alt="{{ movie.Title }}">
      <div class="title">{{ movie.Title }}</div>
      <div class="description-snippet">
        {{ movie.Description[:80] }}{% if movie.Description|length > 80 %}...{%
endif %}
      </div>
      <div class="genre">{{ movie.Genre }}</div>
    </div>
    {% endfor %}
  </div>

  <!-- Modal -->
  <div id="movieModal" class="modal">
    <div class="modal-content">
      <span class="close" onclick="closeModal()">&times;</span>
      <h2 id="modalTitle"></h2>
      <img id="modalImage" src="" alt="Movie Poster">
      <p id="modalDescription" class="description"></p>
      <p id="modalRating" class="rating"></p>
    </div>
  </div>

  <script>
    function openModal(title, description, imageUrl, avgRating, numRatings) {
      document.getElementById('modalTitle').innerText = title;
      document.getElementById('modalDescription').innerText = description;
      document.getElementById('modalImage').src = imageUrl;
      document.getElementById('modalRating').innerText =
          avgRating ? `⭐ ${avgRating.toFixed(1)} (${numRatings} ratings)` : "No
ratings yet";
      document.getElementById('movieModal').style.display = 'block';
    }
    function closeModal() { document.getElementById('movieModal').style.display =
'none'; }
    window.onclick = function(event) { if(event.target ===
document.getElementById('movieModal')) closeModal(); }
  </script>
</body>
</html>
```

Rating.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Submit Ratings</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.4.0/css/all.min.css">
    <style>
        body {
            font-family: 'Poppins', sans-serif;
            background: #f5f7fa;
            margin: 0;
            padding: 2rem;
        }
        h1 { text-align:center; margin-bottom: 1.5rem; color: #2a5298; }
        form {
            background: #fff;
            padding: 2rem;
            border-radius: 10px;
            max-width: 600px;
            margin: 0 auto 2rem auto;
            box-shadow: 0 2px 10px rgba(0,0,0,0.1);
        }
        label { display:block; margin-top: 1rem; font-weight: bold; }
        select, input[type="number"], textarea {
            width: 100%;
            padding: 0.7rem;
            margin-top: 0.3rem;
            border-radius: 5px;
            border: 1px solid #ccc;
            font-size: 1rem;
            box-sizing: border-box;
        }
        button {
            margin-top: 1.5rem;
            padding: 0.8rem 1.5rem;
            border: none;
            border-radius: 30px;
            background: #ff4b2b;
            color: white;
            font-size: 1rem;
            cursor: pointer;
            transition: 0.3s;
        }
        button:hover { transform: scale(1.05); }
        .reviews {
            max-width: 800px;
            margin: 0 auto;
        }
        .review-card {
            background: #fff;
            padding: 1rem;
            margin-bottom: 1rem;
            border-radius: 10px;
            box-shadow: 0 2px 6px rgba(0,0,0,0.1);
        }
        .review-card h3 { margin:0; color: #2a5298; }
        .review-card p { margin:0.3rem 0; color:#555; }
        .rating { color: #ff8c00; font-weight:bold; }
    </style>
</head>
<body>
    <h1><i class="fa-solid fa-star"></i> Submit Your Movie Rating</h1>

    <!-- Rating Form -->
    <form method="POST" action="{{ url_for('submit_rating') }}">
        <label for="user">User:</label>
        <select id="user" name="user" required>
            <option value="" disabled selected>Select user</option>
            {% for user in users %}
```

```html
                <option value="{{ user.UserID }}">{{ user.Name }}</option>
            {% endfor %}
        </select>

        <label for="movie">Movie:</label>
        <select id="movie" name="movie" required>
            <option value="" disabled selected>Select movie</option>
            {% for movie in movies %}
                <option value="{{ movie.MovieID }}">{{ movie.Title }}</option>
            {% endfor %}
        </select>

        <label for="rating">Rating (1-5):</label>
        <input type="number" id="rating" name="rating" min="1" max="5" required>

        <label for="review">Review:</label>
        <textarea id="review" name="review" rows="4" placeholder="Write your thoughts..."></textarea>

        <button type="submit"><i class="fa-solid fa-paper-plane"></i> Submit Rating</button>
    </form>

    <!-- Display Submitted Reviews -->
    <div class="reviews">
        <h2 style="text-align:center; margin-bottom:1rem; color:#2a5298;">User Reviews</h2>
        {% for r in reviews %}
        <div class="review-card">
            <h3>{{ r.Name }} rated "{{ r.Title }}"</h3>
            <p class="rating">⭐ {{ r.Rating }}</p>
            {% if r.Review %}<p>{{ r.Review }}</p>{% endif %}
        </div>
        {% endfor %}
    </div>
</body>
</html>
```

Code Explanation:

The **Movie Rating Portal** is implemented using **Python and Flask** for the backend, with **Pandas** for data handling and **CSV operations**. The code is structured into several key components:

1. Backend (Flask App)

- The main Flask application handles **routing, data processing, and rendering templates**.
- Four primary routes are defined:
    - **/** (Home): Displays the homepage with a fullscreen slideshow and navigation buttons.
    - **/movies**: Reads `ratings.csv`, calculates the average rating per movie using Pandas, merges with movie details, and displays the results in a grid layout.
    - **/ratings**: Displays a form to submit ratings. New ratings are appended to `ratings.csv` and the data is reloaded.
    - **/stats**: Merges movies and ratings, generates pivot tables for average ratings per movie, genre, and user, and exports the results to CSV files.

---

2. Frontend (HTML, CSS, JavaScript)

- The **homepage** has a fullscreen **cinema-style background** with overlay buttons for navigating to “Movies” and “Ratings” pages.
- The **Movies page** displays movies in **card format** with posters, descriptions, and calculated average ratings.

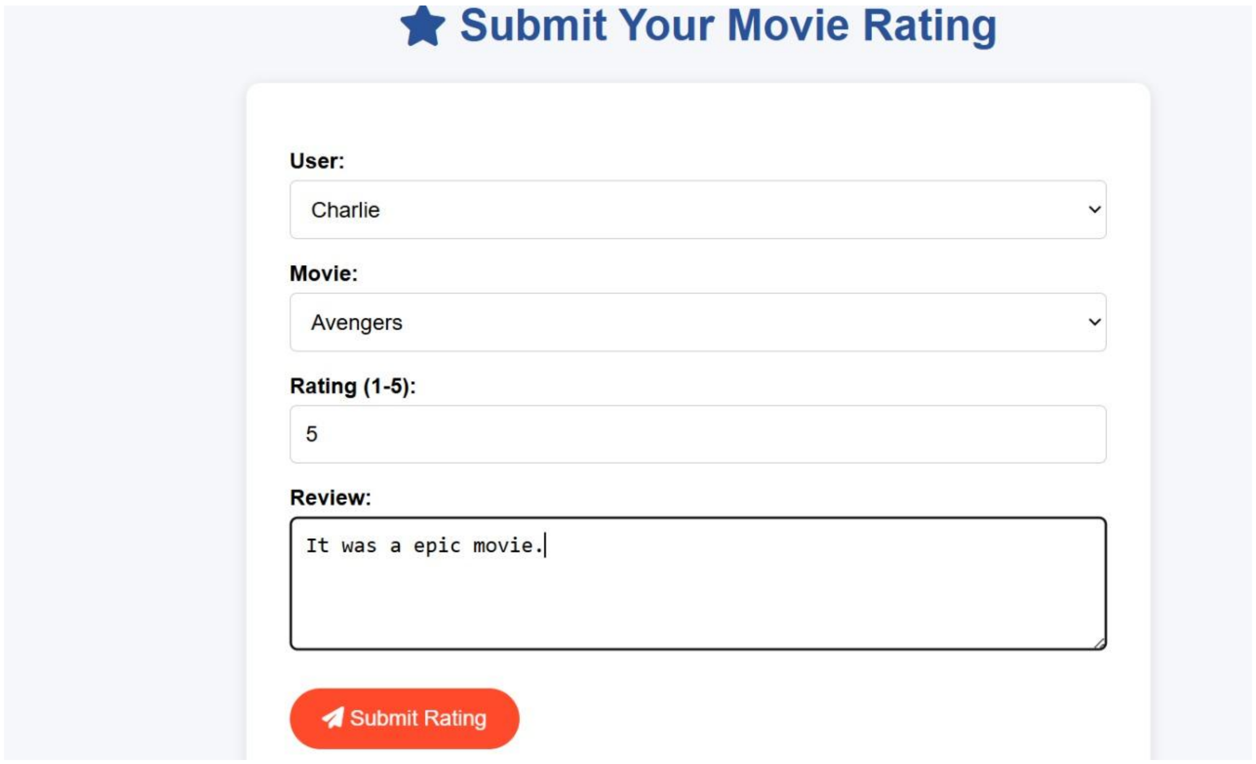
- The **Ratings page** provides a **form** where users can select a movie, choose a rating (1–5), and write a review. A confirmation message is shown after submission.
- **CSS** is used to style forms, tables, buttons, cards, and backgrounds, while **JavaScript** manages dynamic effects like slideshow transitions and modal popups.

---

3. Data Handling

- **CSV files** (`movies.csv`, `ratings.csv`, `users.csv`) are used for persistent storage of movies, ratings, and user data.
- **Pandas** is responsible for reading, merging, and analyzing this data.
- **Pivot tables** are generated to efficiently calculate:
  - Average ratings per movie.
  - Average ratings per genre.
  - User-specific rating trends.
- The processed results can also be exported into new CSV files for further reporting or analysis.

Output & Screen shots:



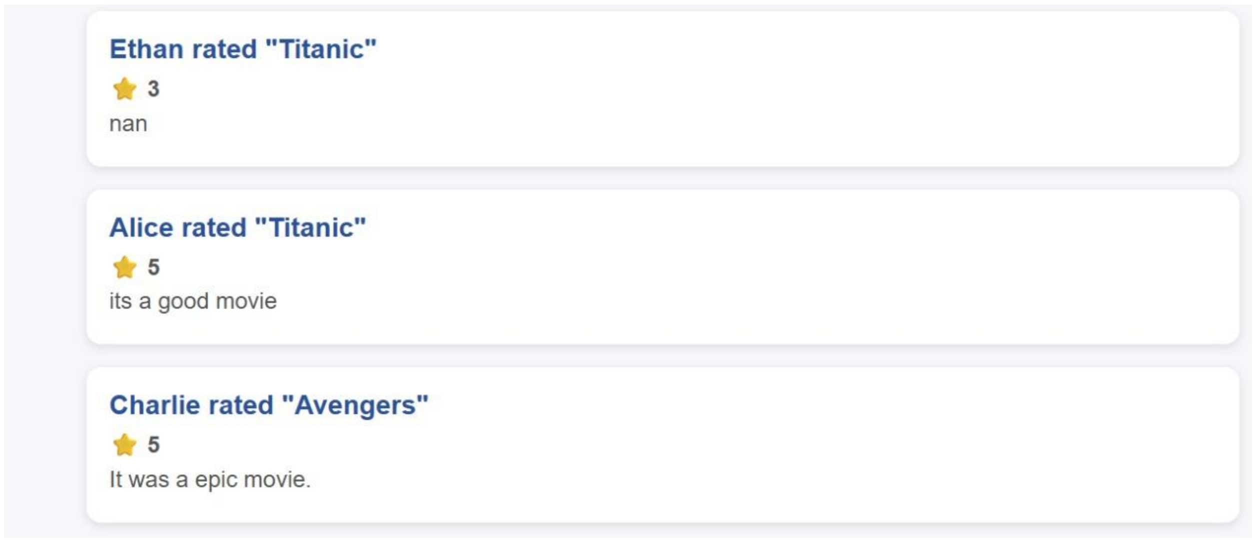
Submit Your Movie Rating

User:
Charlie

Movie:
Avengers

Rating (1-5):
5

Review:
It was a epic movie.



Ethan rated "Titanic"
3
nan

Alice rated "Titanic"
5
its a good movie

Charlie rated "Avengers"
5
It was a epic movie.

Advantages & Limitations:

Advantages:

- User-friendly interface with interactive slideshow and navigation.
- Easy to submit, view, and analyze movie ratings.
- Real-time data updates with automatic recalculation of averages.
- Data-driven insights using Pandas pivot tables for movies, genres, and users.
- Lightweight and easy to deploy using Flask and CSV storage.
- Exported CSV reports enable further analysis or offline usage.

Limitations:

- Data is stored in CSV files, which is not suitable for large-scale datasets.
- No authentication or user validation; anyone can submit ratings.
- Limited functionality compared to full-fledged movie rating platforms (e.g., no comments, reviews, or recommendations).
- Performance may degrade if the number of movies, users, or ratings grows significantly.
- The design and functionality are basic and may require enhancements for commercial use.

Future Enhancements:

- Implement a database (e.g., MySQL, PostgreSQL) instead of CSV files for better scalability and performance.
- Add user authentication and authorization to secure rating submissions.
- Enable movie search, filtering, and sorting options for better user experience.
- Incorporate user reviews and comments along with ratings.
- Add recommendation system using collaborative filtering or content-based filtering.
- Enhance frontend with responsive design for mobile and tablet devices.
- Include data visualization (charts and graphs) for ratings analysis.
- Allow admin dashboard for managing movies, users, and ratings efficiently.

Conclusion:

The Movie Rating Portal is a simple yet effective web application that demonstrates the integration of backend data processing with an interactive frontend. It allows users to view movies, submit ratings, and analyze data through pivot tables, providing insights into movie popularity, user preferences, and genre performance. Using Flask and Pandas, the project highlights essential concepts of web development, data handling, and dynamic content rendering. Although designed as a mini-project, it lays the foundation for building more advanced movie rating platforms with features like user authentication, recommendations, and data visualization.