Movie Ratings Dataset Exploration with Pivot Tables

The **Movie Rating Portal** is a simple and interactive web application where users can explore movies and give their ratings.

It is built using Flask, Python, Pandas, HTML, and CSS. The project has a clean and modern homepage with a slideshow background, along with easy navigation buttons that make the user experience smooth and engaging

BY:

Veluru Nandini

4GW23CI062

INDEX:

Introduction(Overview):

The Movie Rating Portal is a web-based application designed to make rating and exploring movies simple and engaging. Built using Flask (Python web framework), Pandas for data analysis, HTML, CSS, and JavaScript, the portal combines both frontend interactivity and backend data processing.

The homepage features a slideshow background with navigation buttons, giving it a modern and attractive look. Users can browse the list of movies, provide their ratings, and view insightful analysis reports. The system processes user ratings to calculate the average rating per movie, genre, and user, along with the total count of ratings per movie. All results are exported into CSV files, making the application useful not only for user interaction but also for basic data analysis.

This project serves as a mini-model of how online rating platforms like IMDb or Rotten Tomatoes function on a smaller scale, focusing on user experience, interactivity, and data-driven insights.

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 Portal is implemented using **Flask** as the backend framework and **Python** for server-side logic and data handling. **Pandas** is used to process CSV files for movies, users, and ratings, allowing for calculations like average ratings and rating counts.

The system consists of the following modules:

1. Homepage:
Displays a slideshow of images with navigation buttons to access the **View Movies** and **Rating** pages.

2. View Movies Module (`/view`):
Shows the list of all movies along with their average ratings.

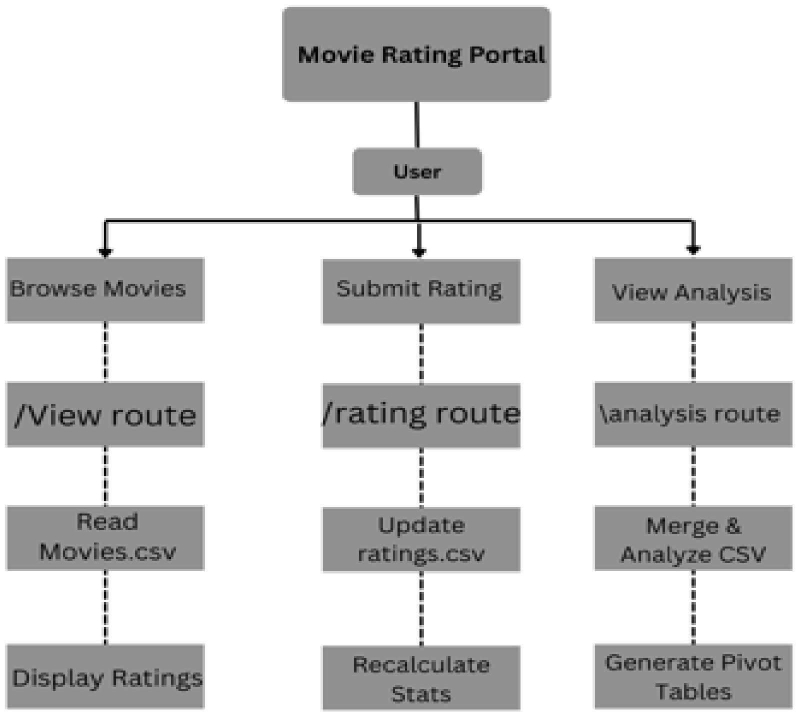
3. Rating Module (`/rating`):
Allows users to select a movie and submit their rating, which is stored in the ratings CSV file.

4. Analysis Module (`/analysis`):
Merges movie and rating data to generate pivot tables for:

- Average rating per movie
- Average rating per genre
- Average rating per user
- Count of ratings per movie
  The processed data is exported as CSV files for further analysis.

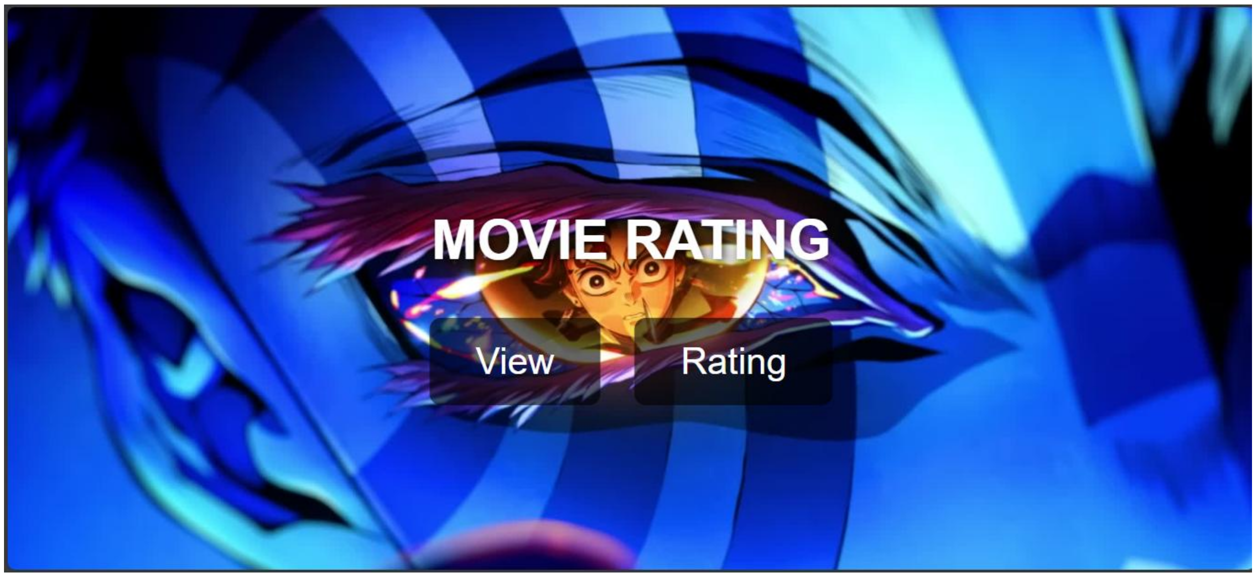
5. Data Storage:
All user, movie, and rating information is stored in CSV files, which are read and updated dynamically through the application.**6. Frontend:**
The frontend is built using **HTML, CSS, and JavaScript**, providing forms, tables, and interactive slideshow navigation.

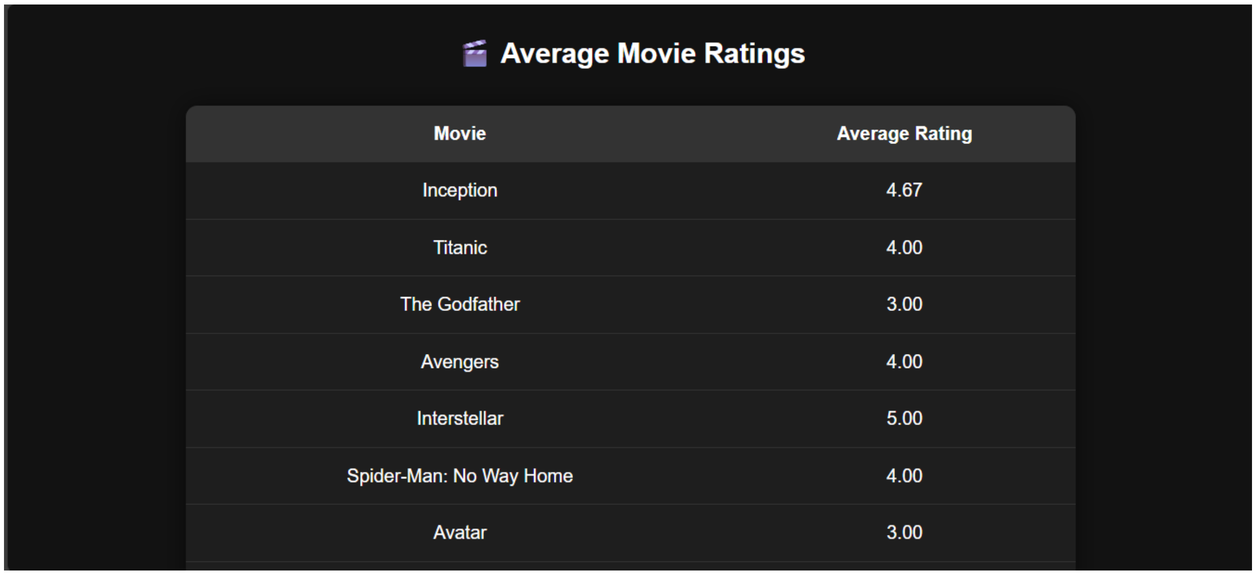
UML Diagram:


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

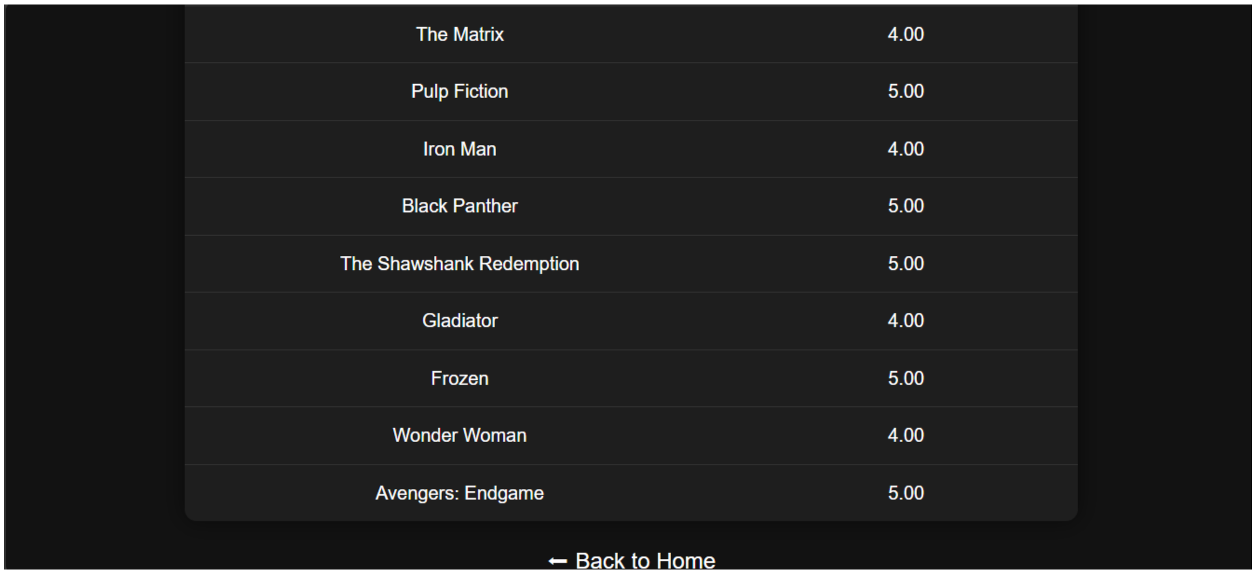
Front-end Desgin:

The Movie Rating Portal features a modern, dark-themed interface built with HTML, CSS, and JavaScript. The homepage includes a full-screen slideshow with navigation buttons for viewing movies or submitting ratings. The **View Movies** page shows a table of movies with average ratings, while the **Rating** page provides a form for submitting user ratings. The **Analysis** page displays pivot table results for movies, genres, and users. The design emphasizes clarity, responsiveness, and user-friendly navigation.



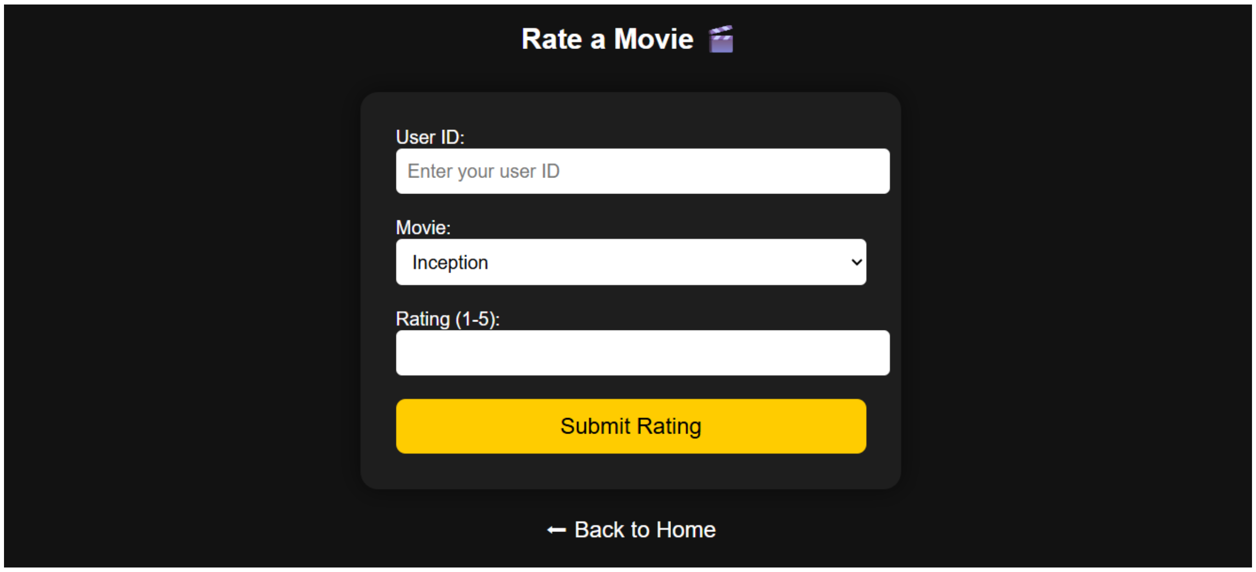


🎬 Average Movie Ratings



Movie	Average Rating
Inception	4.67
Titanic	4.00
The Godfather	3.00
Avengers	4.00
Interstellar	5.00
Spider-Man: No Way Home	4.00
Avatar	3.00
The Matrix	4.00
Pulp Fiction	5.00
Iron Man	4.00
Black Panther	5.00
The Shawshank Redemption	5.00
Gladiator	4.00
Frozen	5.00
Wonder Woman	4.00
Avengers: Endgame	5.00

⟵ Back to Home


Rate a Movie 🎬
User ID:
Enter your user ID
Movie:
Inception
Rating (1-5):
Submit Rating
⬅ Back to Home

Code:

main.py

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

app = Flask(__name__)


movies_file = "movies.csv"
ratings_file = "ratings.csv"
users_file = "users.csv"


movies = pd.read_csv(movies_file)
if not os.path.exists(ratings_file):
    pd.DataFrame(columns=["UserID", "MovieID", "Rating",
"Timestamp"]).to_csv(ratings_file, index=False)
ratings = pd.read_csv(ratings_file)
users = pd.read_csv(users_file)


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


@app.route("/view")
def view():
    global ratings

    avg_ratings = ratings.groupby("MovieID")["Rating"].mean().reset_index()
    avg_ratings = avg_ratings.merge(movies, on="MovieID")
    return render_template("view.html",
movies=avg_ratings.to_dict(orient="records"))


@app.route("/rating", methods=["GET", "POST"])
def rating():
    global ratings
    message = ""
    if request.method == "POST":
        user = request.form["user"]
        movie = request.form["movie"]
        rating_val = float(request.form["rating"])
        timestamp = pd.Timestamp.now().strftime("%Y-%m-%d %H:%M:%S")


        new_row = pd.DataFrame([[user, movie, rating_val, timestamp]],
                               columns=["UserID", "MovieID", "Rating",
"Timestamp"])
        new_row.to_csv(ratings_file, mode="a", header=False, index=False)


        ratings = pd.read_csv(ratings_file)

        message = "✅ Rating submitted successfully!"

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


@app.route("/analysis")
def analysis():
    global ratings, movies


    merged = ratings.merge(movies, on="MovieID", how="left")


    movie_avg = merged.pivot_table(values="Rating", index="Title",
aggfunc="mean").reset_index()


    genre_avg = merged.pivot_table(values="Rating", index="Genre",
aggfunc="mean").reset_index()


    user_avg = merged.pivot_table(values="Rating", index="UserID",
aggfunc="mean").reset_index()


    count_ratings = merged.pivot_table(values="Rating", index="Title",
aggfunc="count").reset_index()
    count_ratings.rename(columns={"Rating": "RatingCount"}, inplace=True)
```

```python
    movie_avg.to_csv("movie_avg_ratings.csv", index=False)
    genre_avg.to_csv("genre_avg_ratings.csv", index=False)
    user_avg.to_csv("user_avg_ratings.csv", index=False)
    merged.to_csv("cleaned_movie_ratings.csv", index=False)

    return render_template(
        "analysis.html",
        movie_avg=movie_avg.to_dict(orient="records"),
        genre_avg=genre_avg.to_dict(orient="records"),
        user_avg=user_avg.to_dict(orient="records"),
        count_ratings=count_ratings.to_dict(orient="records")
    )


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

home.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Movie Ratings Home</title>
<style>
  html, body {
    margin: 0;
    padding: 0;
    height: 100%;
    width: 100%;
    overflow: hidden;
    font-family: Arial, sans-serif;
  }

  .slideshow-container {
```

```
  position: relative;
  height: 100vh;
  width: 100%;
}

.slide {
  position: absolute;
  top: 0; left: 0;
  width: 100%; height: 100%;
  background-size: cover;
  background-position: center;
  opacity: 0;
  transition: opacity 1s ease-in-out;
}

.slide.active { opacity: 1; }

.overlay {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  text-align: center;
  color: white;
  z-index: 10;
}

.overlay h1 {
  font-size: 3rem;
  margin-bottom: 40px;
  text-shadow: 2px 2px 8px rgba(0,0,0,0.7);
}

.overlay-buttons {
  display: flex;
  gap: 30px;
  justify-content: center;
}

.btn {
  padding: 20px 40px;
  font-size: 2rem;
  color: white;
  text-decoration: none;
  background-color: rgba(0,0,0,0.6);
```

```
    border-radius: 10px;
    transition: 0.3s;
  }

  .btn:hover {
    background-color: rgba(0,0,0,0.9);
    transform: scale(1.1);
  }
</style>
</head>
<body>

<div class="slideshow-container">
  <div class="slide" style="background-image:
url('/static/images/slide1.jpg');"></div>
  <div class="slide" style="background-image:
url('/static/images/slide2.jpg');"></div>
  <div class="slide" style="background-image:
url('/static/images/slide3.jpg');"></div>
  <div class="slide" style="background-image:
url('/static/images/slide4.webp');"></div>
  <div class="slide" style="background-image:
url('/static/images/slide5.jpg');"></div>
  <div class="slide" style="background-image:
url('/static/images/slide6.webp');"></div>
  <div class="slide" style="background-image:
url('/static/images/slide7.jpg');"></div>
  <div class="slide" style="background-image:
url('/static/images/slide8.jpeg');"></div>
  <div class="slide" style="background-image:
url('/static/images/slide9.webp');"></div>
  <div class="slide" style="background-image:
url('/static/images/slide1.jpg');"></div>


  <div class="overlay">
    <h1>MOVIE RATING </h1>
    <div class="overlay-buttons">
      <a href="/view" class="btn">View</a>
      <a href="/rating" class="btn">Rating</a>
    </div>
  </div>
</div>

<script>
```

```
  const slides = document.querySelectorAll('.slide');
  let current = 0;

  function showSlide(i) {
    slides.forEach(s => s.classList.remove('active'));
    slides[i].classList.add('active');
  }

  showSlide(current);
  setInterval(() => {
    current = (current + 1) % slides.length;
    showSlide(current);
  }, 3000);
</script>

</body>
</html>
```

View.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Movie Ratings</title>
<style>
  body {
    font-family: Arial;
    background-color: #121212;
    color: #ffffff;
    padding: 50px;
  }
  table {
    border-collapse: collapse;
    width: 80%;
    margin: auto;
    background-color: #1e1e1e;
    border-radius: 10px;
    overflow: hidden;
    box-shadow: 0 0 20px rgba(0,0,0,0.5);
  }
  th, td {
    padding: 15px;
    text-align: center;
    border-bottom: 1px solid #333;
  }
  th {
    background-color: #333;
    color: #fff;
  }
  tr:last-child td { border-bottom: none; }
  h2 { text-align:center; margin-bottom:30px; }
  a.back {
    display: block;
    text-align: center;
    margin-top: 20px;
    color: #ffffff;
    text-decoration: none;
    font-size: 1.2rem;
  }
  a.back:hover { color: #ffcc00; }
</style>
</head>
<body>

<h2>🎬 Average Movie Ratings</h2>

<table>
<tr><th>Movie</th><th>Average Rating</th></tr>
{% for movie in movies %}
<tr>
  <td>{{ movie.Title }}</td>
  <td>{{ "%.2f"|format(movie.Rating) }}</td>
</tr>
{% endfor %}
</table>

<a href="/" class="back">⬅ Back to Home</a>

</body>
</html>
```

Rating.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Rate a Movie</title>
<style>
  body {
    font-family: Arial;
    background-color: #121212;
    color: #fff;
    padding: 50px;
  }
  form {
    background-color: #1e1e1e;
    padding: 30px;
    border-radius: 15px;
    max-width: 400px;
    margin: auto;
    box-shadow: 0 0 20px rgba(0,0,0,0.5);
  }
  input, select {
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    font-size: 1rem;
    border: none;
    border-radius: 5px;
  }
  input:focus, select:focus { outline: none; }
  button {
    width: 100%;
    padding: 12px;
    font-size: 1.2rem;
    background-color: #ffcc00;
    color: #000;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    transition: 0.3s;
  }
  button:hover { background-color: #ffaa00; }
  .message {
    text-align: center;
    color: #00ff00;
    margin-bottom: 20px;
    font-size: 1.2rem;
  }
  h2 { text-align:center; margin-bottom: 30px; }
  a.back {
    display:block;
    text-align:center;
    margin-top: 20px;
    color: #fff;
    text-decoration: none;
    font-size: 1.2rem;
  }
  a.back:hover { color: #ffcc00; }
</style>
</head>
<body>

<h2>Rate a Movie 🎬</h2>

{% if message %}
  <div class="message">{{ message }}</div>
{% endif %}

<form method="POST">
  <label>User ID:</label>
  <input type="text" name="user" placeholder="Enter your user ID" required>

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

  <label>Rating (1-5):</label>
  <input type="number" name="rating" min="1" max="5" step="0.5" required>

  <button type="submit">Submit Rating</button>
</form>

<a href="/" class="back">⬅ Back to Home</a>

</body>
```

```
</html>
```

Style.css

```css
/* Fullscreen container */
.slideshow-container {
  height: 100vh;
  width: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
  position: relative;
  transition: background-image 1s ease-in-out;
}

/* Center buttons */
.overlay-buttons {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  display: flex;
  gap: 20px;
  z-index: 10;
}

/* Buttons */
.btn {
  padding: 20px 40px;
  font-size: 2rem;
  color: white;
  text-decoration: none;
  background-color: rgba(0,0,0,0.6);
  border-radius: 10px;
  transition: 0.3s;
}

.btn:hover {
  background-color: rgba(0,0,0,0.9);
  transform: scale(1.1);
}
```

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:

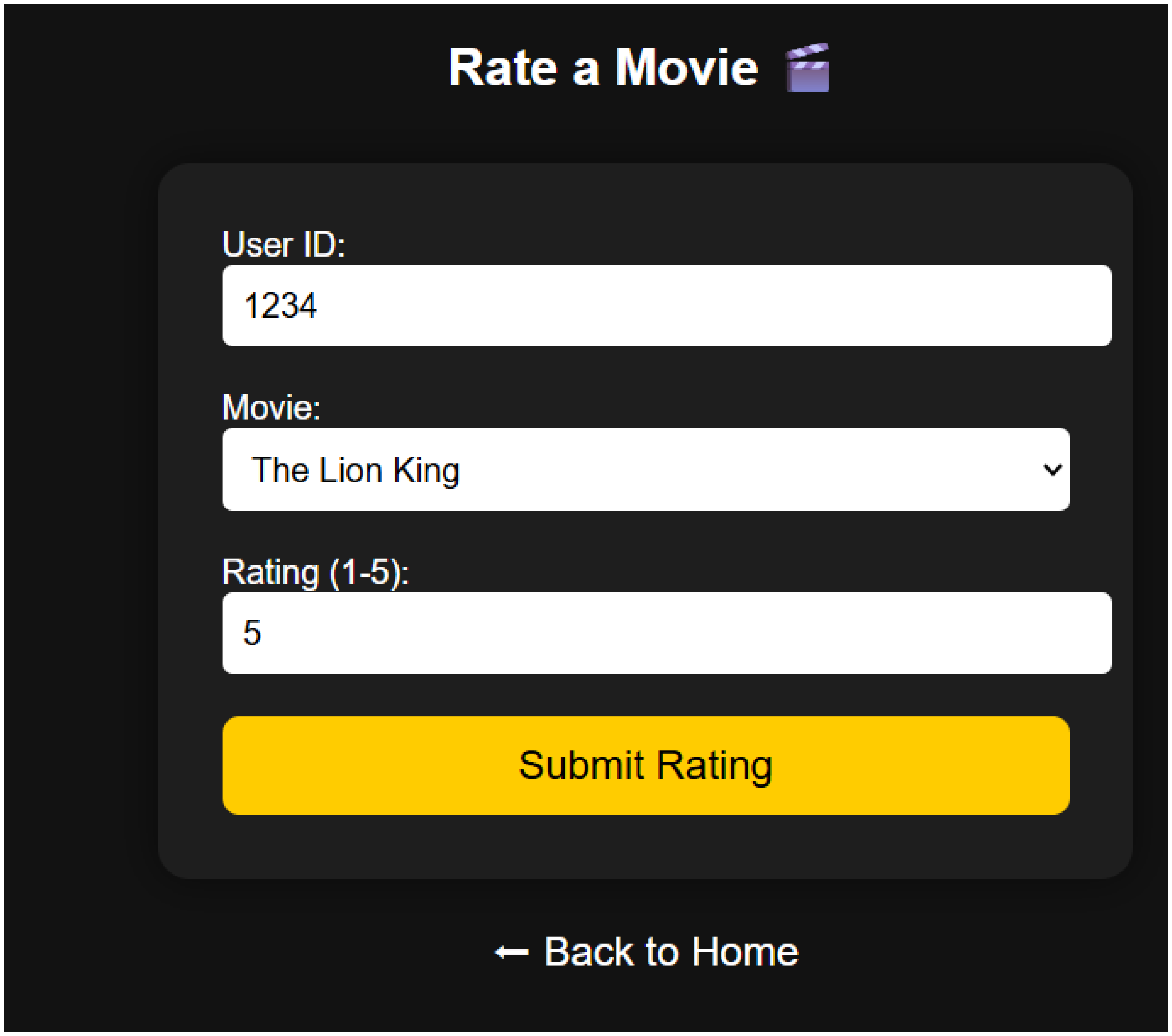
1. **Backend (Flask App)**
   - The main Flask application handles routing, data processing, and rendering templates.
   - Four primary routes are defined:
     - `/` (Home): Displays the homepage with a fullscreen slideshow and navigation buttons.
     - `/view`: Reads `ratings.csv`, calculates the average rating per movie using Pandas, merges with movie details, and displays the results in a table.
     - `/rating`: Displays a form to submit ratings. New ratings are appended to `ratings.csv` and the data is reloaded.
     - `/analysis`: 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 slideshow with overlay buttons for navigating to “View” and “Rating” pages.
   - The View Movies page displays a table of movies with their average ratings.
   - The Rating page provides a form for users to submit their ratings. A success message is shown after submission.

    - CSS styles forms, tables, buttons, and slideshows, while JavaScript handles the slideshow transitions.

3. **Data Handling**
    - CSV files (`movies.csv`, `ratings.csv`, `users.csv`) store the data. Pandas reads, merges, analyzes, and exports this data.
    - Pivot tables are used to calculate averages and counts efficiently.

The code combines backend logic, data analysis, and frontend interactivity to provide a user-friendly movie rating experience.

Output & Screen shots:

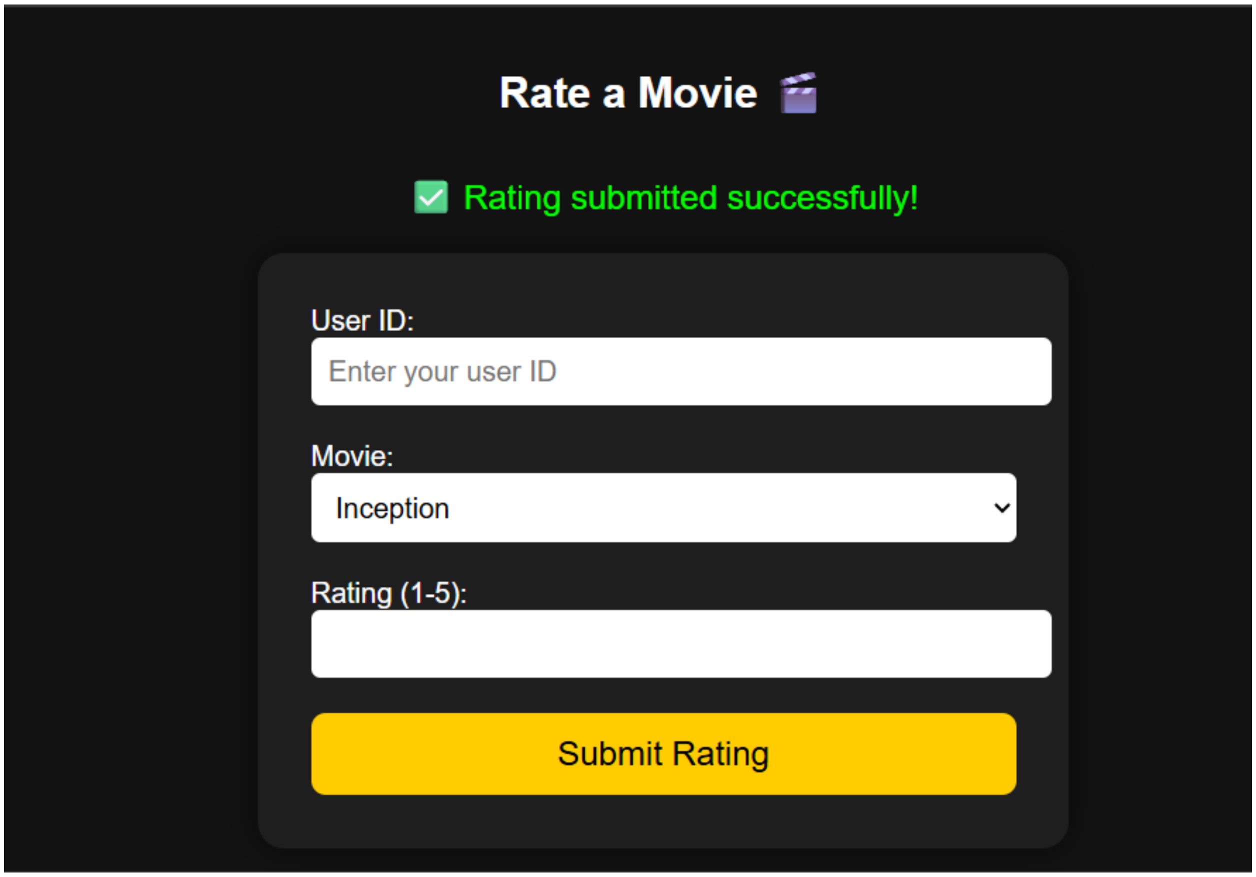


Rate a Movie 🎬

User ID: 1234

Movie: The Lion King

Rating (1-5): 5



Rate a Movie 🎬

✅ Rating submitted successfully!

User ID:

Enter your user ID

Movie:

Inception

Rating (1-5):

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.