**Table of content**

|  |  |  |
| --- | --- | --- |
| **S. No** | **HEADING** | **Pg. No** |
|  | Abstract | 2 |
| 1 | Introduction | 3 |
| 2 | Materials and Methods | 3 |
| 3 | Results and Discussion | 6 |
| 4 | Conclusion | 8 |
|  | References | 8 |

**Movie Recommendation System Using Spark**

**Abstract:**

This project presents a Genre and Year-based Movie Recommendation System developed using PySpark and Flask. The system utilizes collaborative filtering techniques to suggest movies to users based on their preferences. The dataset includes movie ratings, genres, and tags, sourced from Kaggle(Movie 2M dataset). The Flask web application provides an interactive interface for users to input their preferred genre and year, generating personalized movie recommendations.

**Introduction:**

With the increasing volume of available movies, personalized recommendation systems have become essential to help users discover content aligned with their interests. This project leverages the power of PySpark for data processing and collaborative filtering, coupled with Flask to create a user-friendly web interface. The recommendation algorithm considers genre, release year, and user ratings to suggest relevant movies.

**Materials and Methods:**

The project utilizes Flask, PySpark, and related libraries to build the Movie Recommendation System. The key steps involve loading data from CSV files, preprocessing data, and implementing a recommendation algorithm. The Flask app sets up routes for the homepage and movie recommendations. The algorithm recommends movies by filtering based on genre, year, and high ratings.

**Code:**

# Initialize Flask app

app = Flask(\_\_name\_\_)

# Create SparkSession

spark = SparkSession.builder \

.appName("GenreYearBasedMovieRecommendation") \

.getOrCreate()

# Load ratings, movies, and tags data from Google Drive

ratings\_df = spark.read.csv("./datasets/rating.csv", header=True, inferSchema=True)

movies\_df = spark.read.csv("./datasets/movie.csv", header=True, inferSchema=True)

tags\_df = spark.read.csv("./datasets/tag.csv", header=True, inferSchema=True)

# Preprocess data

ratings\_df = ratings\_df.select("userId", "movieId", "rating")

movies\_df = movies\_df.select("movieId", "title", "genres")

tags\_df = tags\_df.select("movieId", "tag", "timestamp")

# Split genres into individual genres

movies\_df = movies\_df.withColumn("genres", explode(split("genres", "\\|")))

# Extract year from timestamp

tags\_df = tags\_df.withColumn("year", year("timestamp"))

# Function to recommend movies based on genre, year, and high rating

def recommend\_movies\_by\_genre\_and\_year(genre, year):

recommended\_movies = ratings\_df.join(movies\_df, on="movieId").filter(movies\_df.genres == genre).join(tags\_df, on="movieId").filter(tags\_df.year == year).groupBy("title").agg({"rating": "avg"}).orderBy(desc("avg(rating)"))

return recommended\_movies

# Define route for homepage

@app.route('/')

def index():

return render\_template('index.html')

# Define route for movie recommendation

@app.route('/recommend', methods=['POST'])

def recommend():

if request.method == 'POST':

selected\_genre = request.form['genre']

selected\_year = int(request.form['year'])

recommended\_movies = recommend\_movies\_by\_genre\_and\_year(selected\_genre,

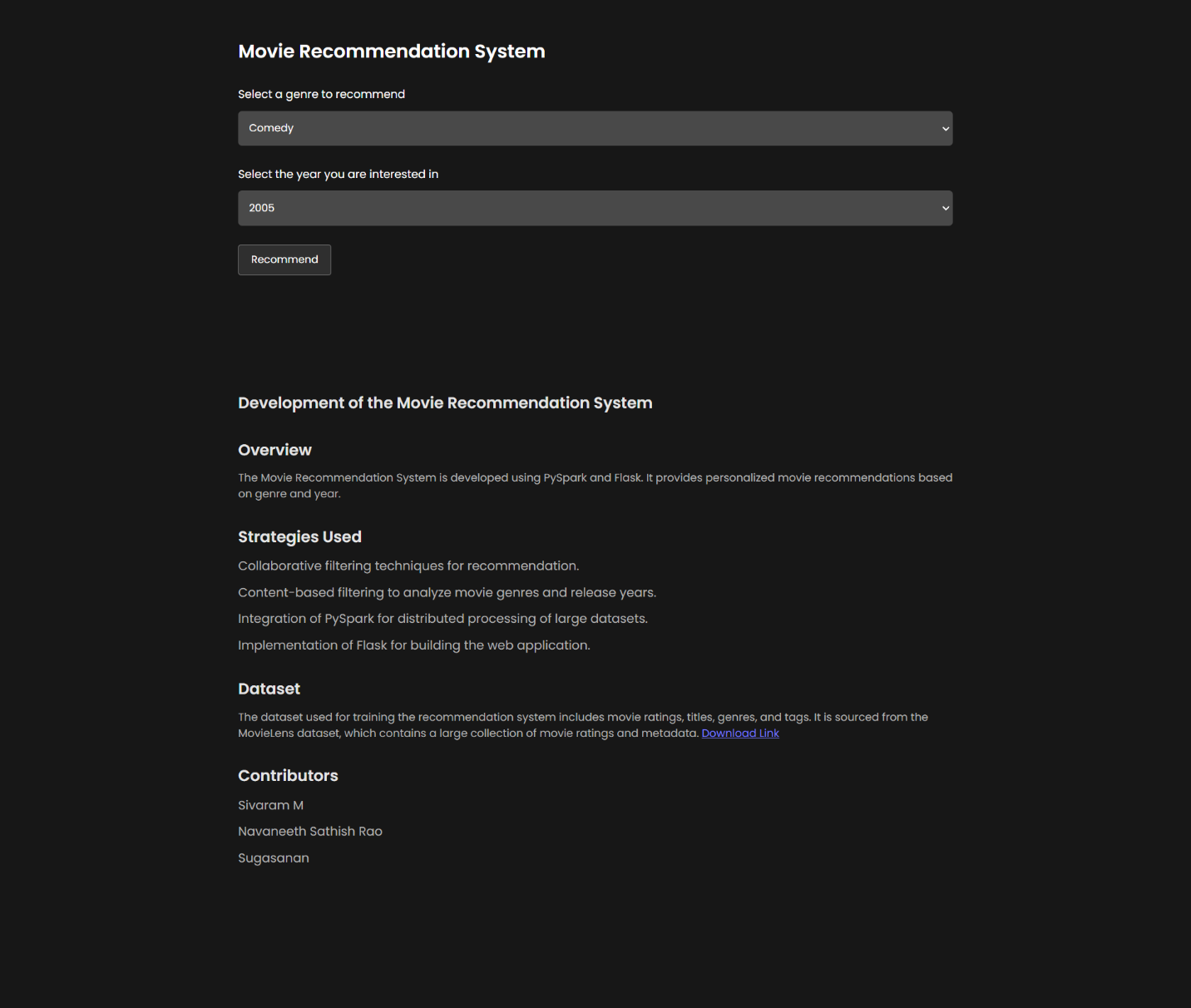
selected\_year)

movie\_list = recommended\_movies.collect()

return render\_template('recommendation.html', movies=movie\_list)

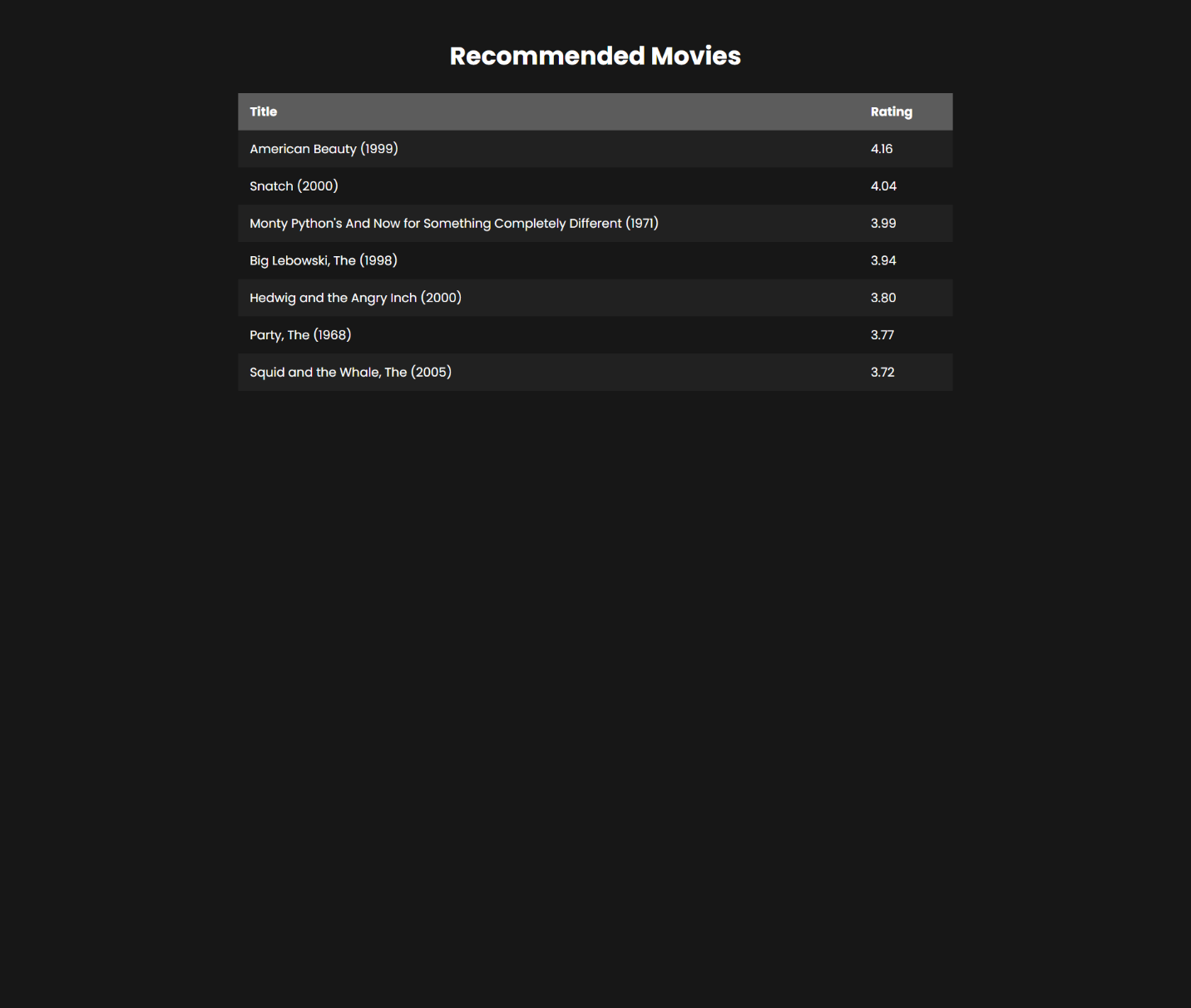
**Results and Discussion:**

The Flask web application allows users to input their preferred genre and year, generating a list of recommended movies.



**Screenshot of Main screen**

The **result page** displays these movies along with their average ratings. The system provides a personalized and intuitive movie recommendation experience. Screenshots of the web interface showcase its user-friendly design and responsiveness.



**Screenshot of Result Page**

The collaborative filtering algorithm effectively identifies movies matching the user's preferences, ensuring a tailored movie-watching experience.

**Conclusion:**

This project successfully implements a Genre and Year-based Movie Recommendation System using Flask and PySpark.

The collaborative filtering approach provides accurate and personalized movie suggestions. The user interface is intuitive and user-friendly, enhancing the overall movie discovery experience.

**References**:

1. MovieLens 20M Dataset: https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset
2. Apache Spark Documentation: https://spark.apache.org/docs/latest/api/python/index.html
3. ChatGPT for Content Generation: https://chat.openai.com/