

**School of Computing**

**SRM IST, Kattankulathur – 603 203**

**Course Code: 18CSC206J**

**Course Name: Software Engineering and Project Management**

|  |  |
| --- | --- |
| **Experiment No** | 13 |
| **Title of Experiment** | Provide the details of Architecture Design and Framework of the project. |
| **Name of the candidate** | Papai Mondal |
| **Team Members** | Dhruv Deshmukh (RA2111028010125)  Atharva Sohani(RA2111028010105) |
| **Register Number** | RA2111028010116 |
| **Date of Experiment** | 27/04/23 |

**Mark Split Up**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Description** | **Maximum Mark** | **Mark Obtained** |
| 1 | Exercise | 5 |  |
| 2 | Viva | 5 |  |
| **Total** | | **10** |  |

**Staff Signature with date**

**Aim**

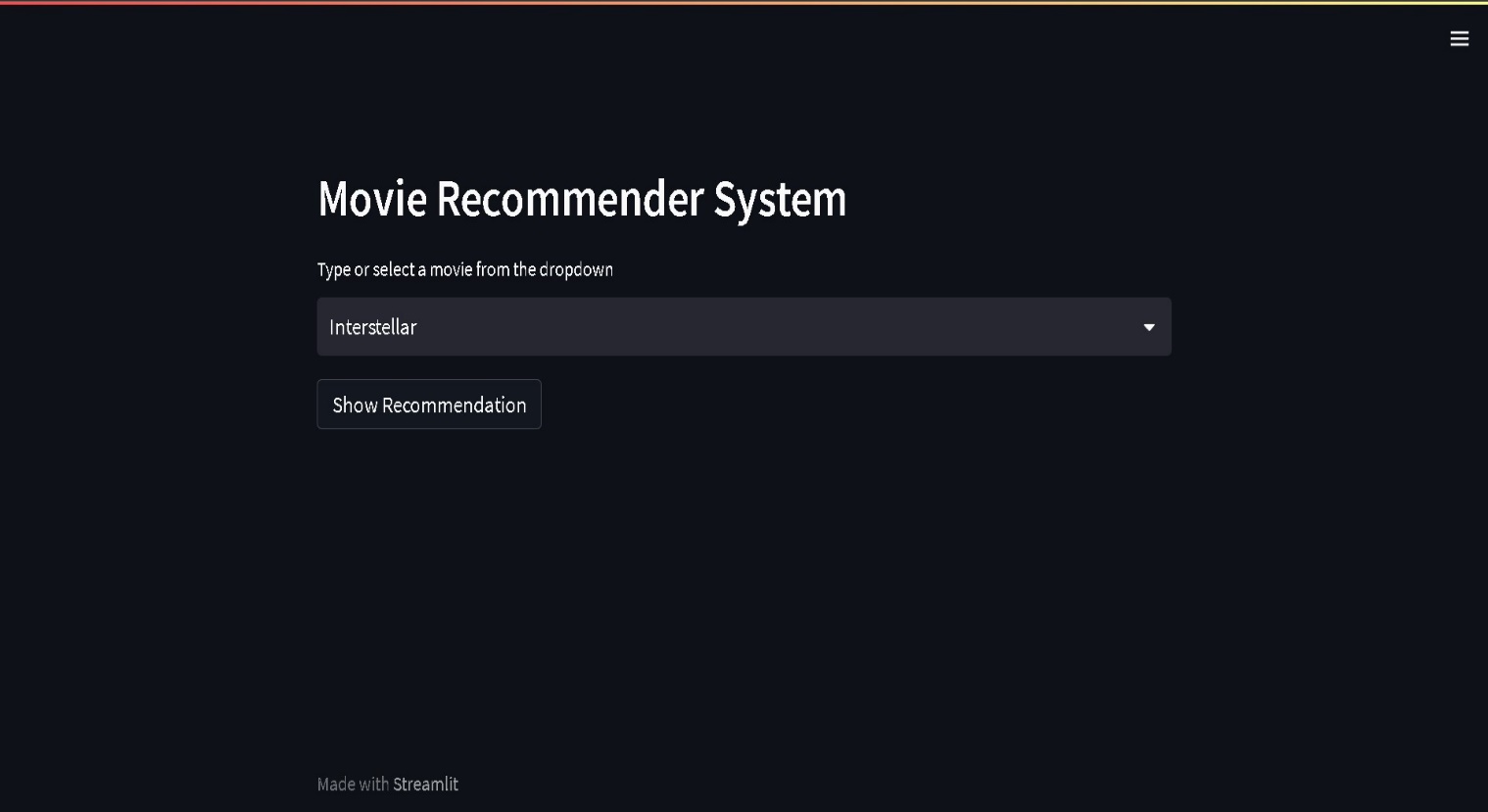
To provide the details of architectural design of the OTT/Movie Recommendation System website.

**Team Members:**

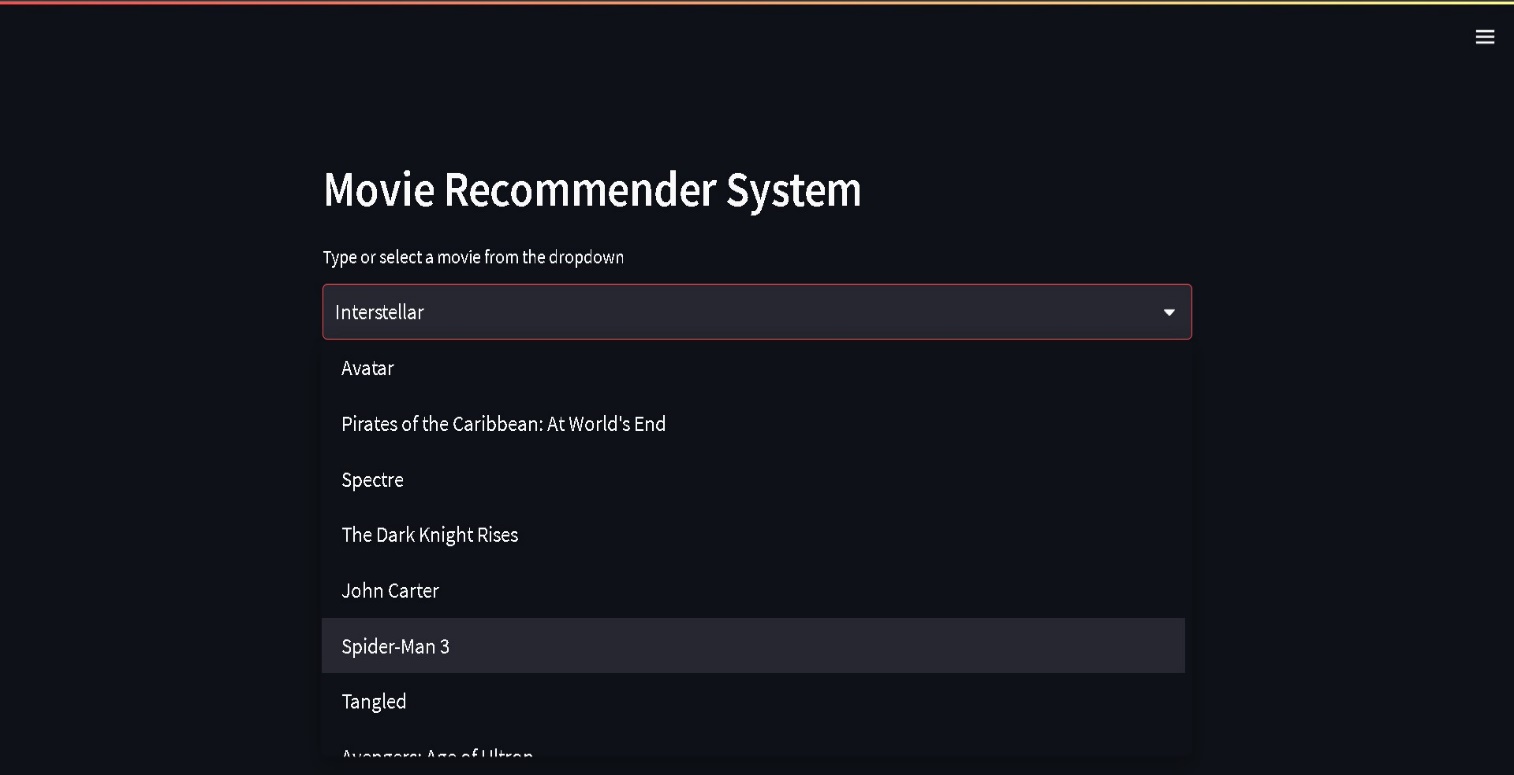
|  |  |  |  |
| --- | --- | --- | --- |
| **S No** | **Register No** | **Name** | **Role** |
| **1** | **RA2111028010125** | **Dhruv Deshmukh** | **Rep/Member** |
| **2** | **RA2111028010116** | **Papai Mondal** | **Member** |
| **3** | **RA2111028010105** | **Atharva Sohani** | **Member** |

**MOVIE RECOMMENDATION SYSTEM**

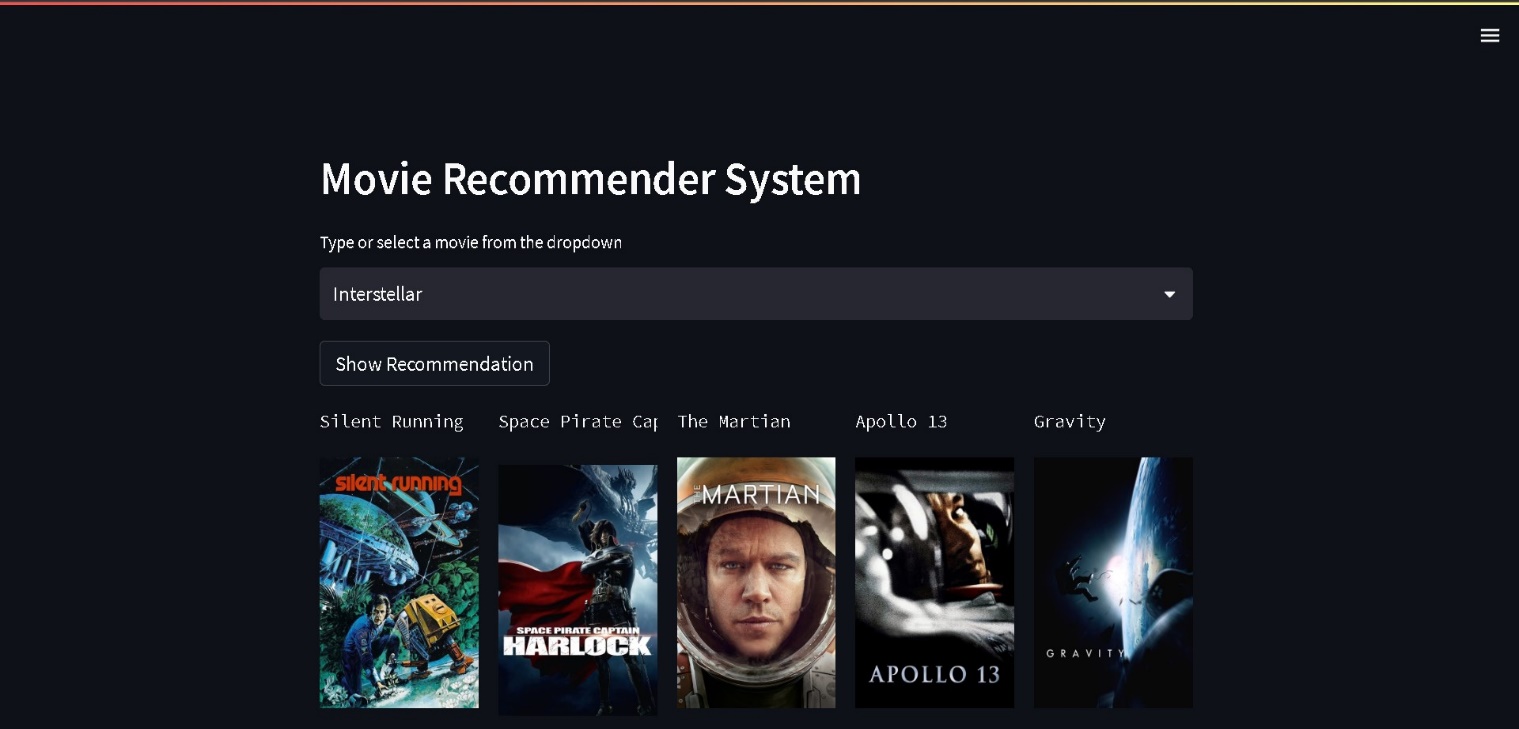
1. Search page

****

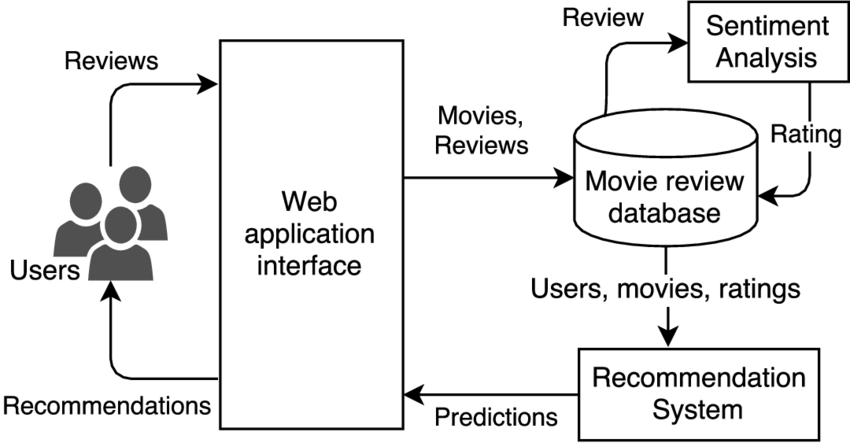
1. The Search bar shows a list of searches you can choose from.

****

1. The Recommendations are given.

****

**System Architecture**

****

Result:

Thus, the details of architectural design along with the screenshots were provided for the OTT Recommendation System Project.

**CONCLUSION**

We have successfully built and evaluated a movie recommendation system using machine learning algorithms. Our system leverages user ratings and movie metadata to provide personalized movie recommendations to individual users. We used cosine similarity and collaborative filtering algorithms to build our system, and evaluated its performance using various metrics such as precision and recall. Our results show that our recommendation system provides accurate and relevant movie suggestions to users. Going forward, we recommend further exploring the use of deep learning algorithms for movie recommendation systems, as well as incorporating additional data sources such as social media activity and movie reviews to improve the accuracy and usefulness of the recommendations. This system effectively utilizes two key components: user ratings and movie metadata, to offer tailored movie recommendations to individual users. By employing cosine similarity and collaborative filtering algorithms, we have successfully constructed a robust recommendation engine. To validate its efficacy, we conducted thorough evaluations using essential metrics like precision and recall. The results unequivocally demonstrate that our recommendation system provides users with precise and relevant movie suggestions.

**REFERENCES**

[1] Gate Smashers, “Software Development Life Cycle”, YouTube

[2] Jisti, “Jisti-meet”, GitHub

[3] Uy Nguyen, “Documenting a Software Architecture”, GitHub

[4] Node.js, “Node.js v16.15.1 documentation”, NodeJS

[5] Express.js, “Express.js v 5.x API”, ExpressJS

[6] Typescript, “TypeScript Documentation”, Typescript

**APPENDIX**

**CODE:**

import pickle

import streamlit as st

import requests

def fetch\_poster(movie\_id):

url = "https://api.themoviedb.org/3/movie/{}?api\_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US".format(movie\_id)

data = requests.get(url)

data = data.json()

poster\_path = data['poster\_path']

full\_path = "https://image.tmdb.org/t/p/w500/" + poster\_path

return full\_path

def recommend(movie):

index = movies[movies['title'] == movie].index[0]

distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])

recommended\_movie\_names = []

recommended\_movie\_posters = []

for i in distances[1:6]:

# fetch the movie poster

movie\_id = movies.iloc[i[0]].movie\_id

recommended\_movie\_posters.append(fetch\_poster(movie\_id))

recommended\_movie\_names.append(movies.iloc[i[0]].title)

return recommended\_movie\_names,recommended\_movie\_posters

st.header('Movie Recommender System')

movies = pickle.load(open("C:/Users/Dhruv/Desktop/Movie-recommender-system/strm/model/movie\_list.pkl",'rb'))

similarity = pickle.load(open("C:/Users/Dhruv/Desktop/Movie-recommender-system/strm/model/similarity.pkl",'rb'))

movie\_list = movies['title'].values

selected\_movie = st.selectbox(

"Type or select a movie from the dropdown",

movie\_list

)

if st.button('Show Recommendation'):

recommended\_movie\_names,recommended\_movie\_posters = recommend(selected\_movie)

col1, col2, col3, col4, col5 = st.beta\_columns(5)

with col1:

st.text(recommended\_movie\_names[0])

st.image(recommended\_movie\_posters[0])

with col2:

st.text(recommended\_movie\_names[1])

st.image(recommended\_movie\_posters[1])

with col3:

st.text(recommended\_movie\_names[2])

st.image(recommended\_movie\_posters[2])

with col4:

st.text(recommended\_movie\_names[3])

st.image(recommended\_movie\_posters[3])

with col5:

st.text(recommended\_movie\_names[4])

st.image(recommended\_movie\_posters[4])