

**Summer Project On**  
**Movie Recommendations System**  
**By**  
**Rohit Mahadik (2021510030)**

Under the guidance of  
**Internal Supervisor**

**Prof. Sakina Salmani**



Department of Master Of Computer Application  
Sardar Patel Institute of Technology  
Autonomous Institute Affiliated to Mumbai University  
2022-23

## **CERTIFICATE OF APPROVAL**

This is to certify that the following student

**Rohit Mahadik (2021510030)**

Have satisfactorily carried out work on the project  
entitled

**“Movie Recommendations System”**

Towards the fulfilment of project, as laid down  
by

Sardar Patel Institute of Technology  
during year  
2022-23.

Project Guide:  
Prof. Sakina Salmani

## **PROJECT APPROVAL CERTIFICATE**

This is to certify that the following student

**Rohit Mahadik (UCID)**

Have successfully completed the Project report on

**“Movie Recommendations System”,**

which is found to be satisfactory and is approved

at

SARDAR PATEL INSTITUTE OF TECHNOLOGY,  
ANDHERI (W), MUMBAI

INTERNAL EXAMINER

EXTERNAL EXAMINER

HEAD OF DEPARTMENT

PRINCIPAL

# Contents

<b>Abstract</b>	i
<b>Objectives</b>	i
<b>List Of Figures</b>	ii
<b>List Of Tables</b>	ii
<b>1 Introduction</b>	1
1.1 Problem Definition . . . . .	1
1.2 Objectives and Scope . . . . .	1
1.2.1 Objectives . . . . .	1
1.2.2 Scope . . . . .	1
1.3 Existing System . . . . .	2
1.4 Proposed System . . . . .	2
1.5 System Requirements . . . . .	3
<b>2 Software Requirement Specification (SRS) and Design</b>	4
2.1 Purpose . . . . .	4
2.2 Definition . . . . .	4
2.3 Overall Description . . . . .	4
2.3.1 Product Functions . . . . .	4
2.3.2 User Characteristics . . . . .	5
<b>3 Project Analysis and Design</b>	6
3.1 Methodologies Adapted . . . . .	6
3.2 Modules . . . . .	7
3.2.1 Activity diagram . . . . .	7
3.2.2 Work Breakdown Structure . . . . .	8
3.2.3 PERT Chart . . . . .	9
3.2.4 Gantt Chart . . . . .	9
3.2.5 Use-Case . . . . .	11
<b>4 Project Implementation and Testing</b>	14
4.1 Home - Landing View . . . . .	14
4.2 Selecting Movies . . . . .	15
4.3 Recommend . . . . .	16
4.4 Actual Recommendations . . . . .	17
4.5 Settings . . . . .	18
4.6 Code 1 . . . . .	19
4.7 Code 2 . . . . .	19
4.8 Code 3 . . . . .	20
4.9 Code 4 . . . . .	20
<b>5 Test Cases</b>	21

<b>6</b>	<b>Limitations</b>	<b>22</b>
<b>7</b>	<b>Future Enhancements</b>	<b>22</b>
<b>8</b>	<b>User Manual</b>	<b>23</b>
<b>9</b>	<b>Bibliography</b>	<b>24</b>
9.1	Web References . . . . .	24

## **Abstract**

A movie recommendation system, or a movie recommender system, is an ML-based approach to filtering or predicting the users' film preferences based on their past choices and behavior. It's an advanced filtration mechanism that predicts the possible movie choices of the concerned user and their preferences towards a domain-specific item, aka movie.

We at Label Your Data have gathered the most up-to-date information about modern movie recommendation systems and how to build them using different ML solutions. We've also touched upon some of the most popular examples of these systems that help many movie fans today stay up to date with all the new releases as well as classics of the cinematography.

## **Objectives**

The Web based Application "Movie Recommendations System" is used

- To provide user a movie recommendations system.
- To provide a similar movie for a user to watch.
- To recommend five similar movies for a user to watch.

## List of Figures

3.1.1Diagrammatic Representation of Waterfall Model . . . . .	6
3.2.1Activity Diagram . . . . .	7
3.2.2Work Breakdown Structure . . . . .	8
3.2.3PERT Chart . . . . .	9
3.2.4Gantt Chart . . . . .	10
3.2.5Use-Case Diagram . . . . .	11
4.1.1 Home View . . . . .	14
4.2.1Selection of title . . . . .	15
4.3.1Hit Recommend . . . . .	16
4.4.15 Movie Recommendations . . . . .	17
4.5.1Edit Settings . . . . .	18

## List of Tables

1.5.1 Hardware Requirements on Server Side . . . . .	3
1.5.2 Hardware Requirements on Client Side . . . . .	3
1.5.3 Software Requirements on Server Side . . . . .	3
1.5.3 Software Requirements on Client Side . . . . .	3
4.2.1 Use Case Table - Selection . . . . .	12
4.2.2 Use Case Table - Suggestions . . . . .	12
4.2.3 Use Case Table - Recommend . . . . .	12
4.2.4 Use Case Table - Recommendations . . . . .	13
4.2.5 Use Case Table - View Marks . . . . .	13
6.1 Test Case . . . . .	21

## 1 Introduction

### 1.1 Problem Definition

To provide the user a movie to watch without having to install any external application.

### 1.2 Objectives and Scope

#### 1.2.1 Objectives

The Android based application "Movie Recommendations System" is

- To provide user a movie recommendations system.
- To provide a similar movie for a user to watch.
- To recommend five similar movies for a user to watch.

#### 1.2.2 Scope

The user can select a movie from a library of over 5000+ movies and have themself similar movies to watch

In this web based application, the user must know the title of the movie.

Our System is being made for so that user will not have to depend on any particular OTT or external source.

### 1.3 Existing System

Currently, only if you are subscribed to paid OTT subscriptions, only then you can have a movie recommender for yourself.

Some of the disadvantages of existing system are as follows :

- Time consuming, Manual work  
Every time a user wants to watch some similar movie, he/she will have to manually subscribe to expensive OTT platforms and search for that movie.
- Inaccuracy  
Many systems are reported to generate non-similar movies based on what an user has selected.
- Complexity  
Many users find it difficult to find a similar movie an any particular platform.

### 1.4 Proposed System

The User is anyone visiting my website will firstly have to enter the title of the movie or he/she can select a movie from a drop-down button.

The user will have to hit recommend and based on the data that I have stored on my project, my system will show 5 similar movies the user can watch, if he/she wants to.

The user will also be shown poster of the movies along with the title.

Some of the advantages of our system are as follows :

- User Friendly  
It provides attractive interface to the user for selecting the title of the movie  
The items of the system are very self-descriptive.
- Accuracy  
On selecting a movie, the user will be getting the most similar movie there exists

## 1.5 System Requirements

- Hardware Requirements on Server Side

Table 1.5.1: Hardware Requirements on Server Side

Processor	Dual Core Processor or Above
RAM	Minimum 4 GB RAM
Storage	Minimum 10 GB Hard Disk Space for smooth run

- Hardware Requirements on Client Side

Table 1.5.2: Hardware Requirements on Client Side

Device	Android Device with Touch Screen minimum 5" inch Display
Processor	Dual Core Processor or Above
RAM	Minimum 2 GB RAM
Storage	Minimum 250 MB Storage Space

- Software Requirements on Server Side

Table 1.5.3: Software Requirements on Server Side

Operating System	OS Independent
Database	Firestore

- Software Requirements on Client Side

Table 1.5.3: Software Requirements on Client Side

Operating System	Windows/MacOS/ChromeOS
Internet	Required

## 2 Software Requirement Specification (SRS) and Design

### 2.1 Purpose

The purpose of my system is to develop a web based application that can help user to get similar movies to watch incase they like any particular movie. This can save lots of efforts of users having to get a subscription of any OTT platforms or having the need to install any external application.

### 2.2 Definition

To build a Web based Application so the user can get similar movies to watch in-case they like any particular movie. The user will have to hit recommend and based on the data that I have stored on my project, my system will show 5 similar movies the user can watch, if he/she wants to.

### 2.3 Overall Description

#### 2.3.1 Product Functions

The system function includes:

1. Selection: Users are required to select a movie title from over a library of 5000+ movie titles.
2. Suggestions: As the user starts typing the title of the movie, my system will start recommending the remaining letters of the movie title.
3. Recommend: This will be a button that the user will have to click on in order to get similar movies.
4. Recommendations: Finally, my system will display 5 similar movie titles along with their posters.

### 2.3.2 User Characteristics

There is one type of user:

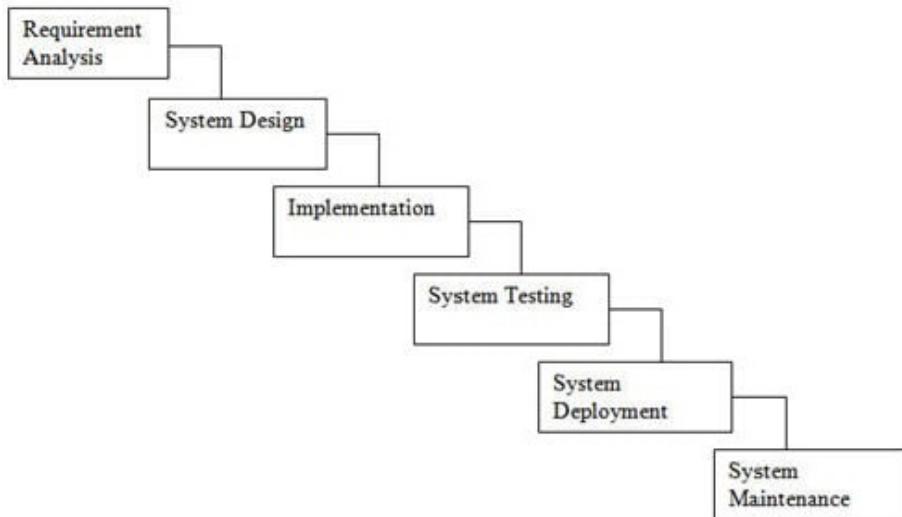
- Normal user: The user will select a movie he/she likes and will get 5 more similar movies to watch.

### 3 Project Analysis and Design

#### 3.1 Methodologies Adapted

In Waterfall model, very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.

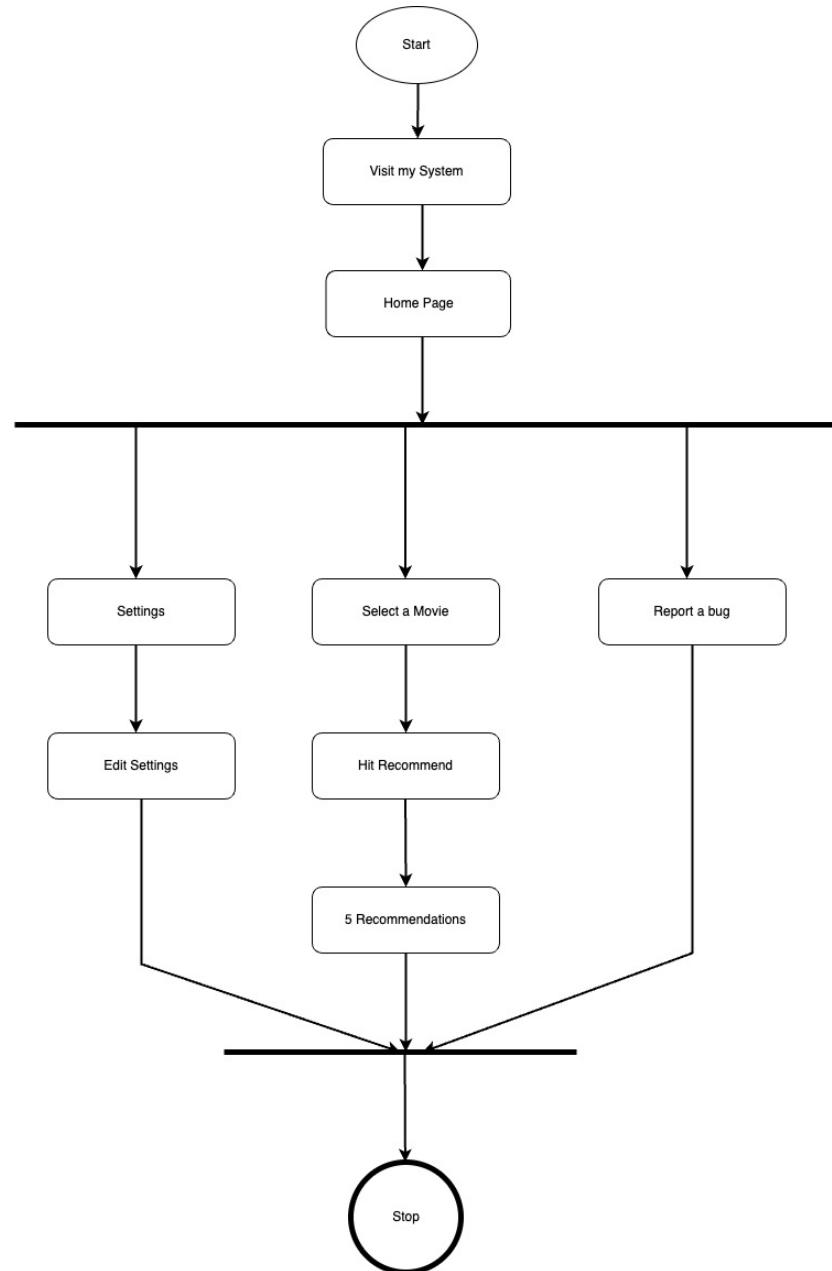
Once the product is developed and if any failure occurs then the cost of such issues is very high, because we need to update everything from document till the logic.



3.1.1: Diagrammatic Representation of Waterfall Model

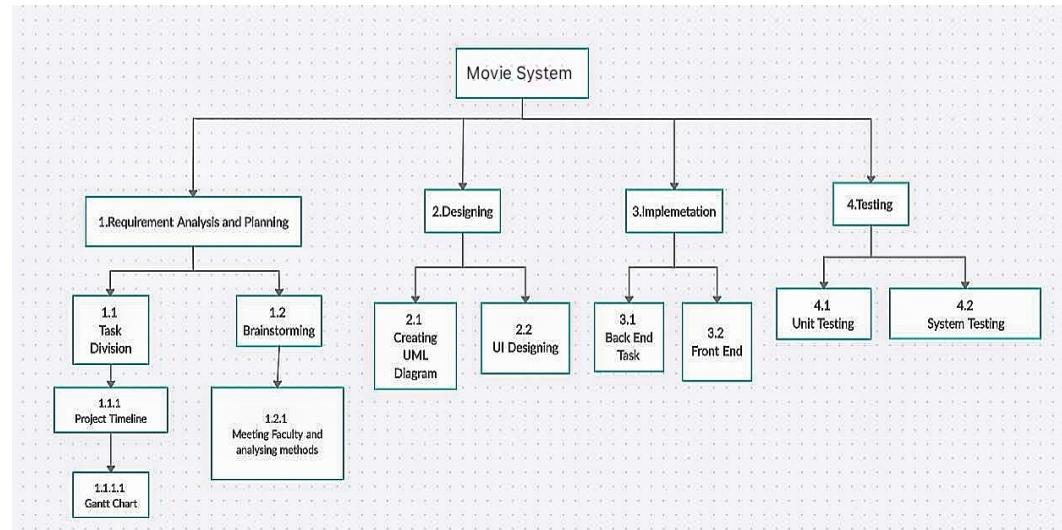
### 3.2 Modules

#### 3.2.1 Activity diagram



3.2.1: Activity Diagram

### 3.2.2 Work Breakdown Structure



3.2.2: Work Breakdown Structure

### 3.2.3 PERT Chart



3.2.3: PERT Chart

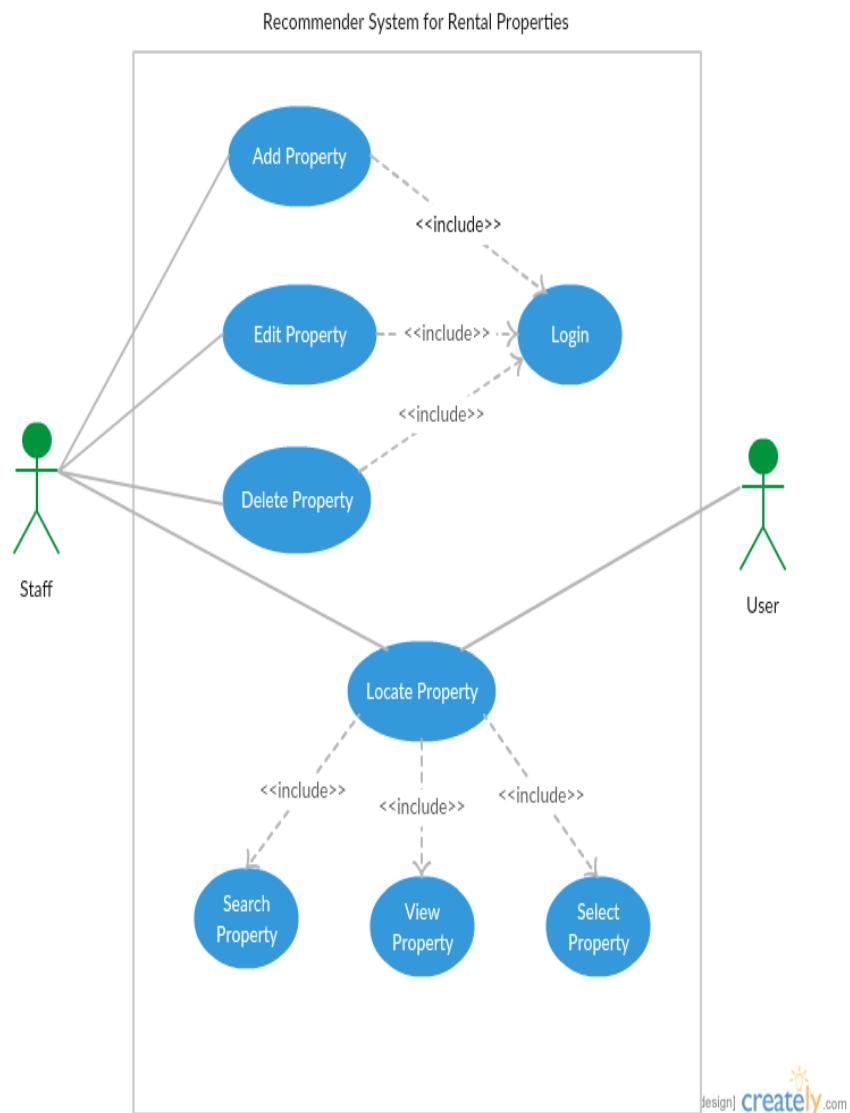
### 3.2.4 Gantt Chart

### MOVIE RECOMMENDATIONS SYSTEM GANTT CHART

Task ID	Task Name	Start Date	End Date	Duration (In Days)	April - May	May - June	June - July	July - August	August - September
T01	Problem Statement	26/04/22	07/05/22	12					
T02	Brainstorming	08/05/22	09/05/22	2					
T03	Requirement Gathering	10/05/22	10/05/22	1					
T04	Project Planning	11/05/22	19/05/22	9					
T05	Analysis	20/05/22	30/05/22	11					
T06	Design	31/05/22	19/06/22	20					
T07	Modelling	20/06/22	19/07/22	30					
T08	Implementation/Coding	20/07/22	14/08/22	26					
T09	Testing	15/08/22	28/08/22	14					
T10	Hosting	29/08/22	21/09/22	24					
T11	Documentation(Final)	15/09/22	18/09/22	4					

3.2.4: Gantt Chart

### 3.2.5 Use-Case



3.2.5: Use-Case Diagram

## Movie Recommendations System

---

Rohit Mahadik (UCID)

Use Cases:

1. Selection
2. Suggestions
3. Recommend
4. Recommendations

Table 4.2.1: Use Case Table - Selection

Use Case ID	1
Use Case Name	Selection
Actor	User
Pre-Condition	Visits my Website
Post-Condition	User at Home page
Flow of events	Visit my Website and Start typing the title of the movie.

Table 4.2.2: Use Case Table - Suggestions

Use Case ID	2
Use Case Name	Suggestions
Actor	User
Pre-Condition	They start typing title of the movie.
Post-Condition	User is ready to hit recommend after successfully typing the entire title of the movie.
Flow of events	Visits my Website and Start typing the title of the movie and my system recommends the remaining letters of that movie title.

Table 4.2.3: Use Case Table - Recommend

Use Case ID	3
Use Case Name	Recommend
Actor	User
Pre-Condition	Typed the entire title of the movie.
Post-Condition	User clicks on the button "Recommend".

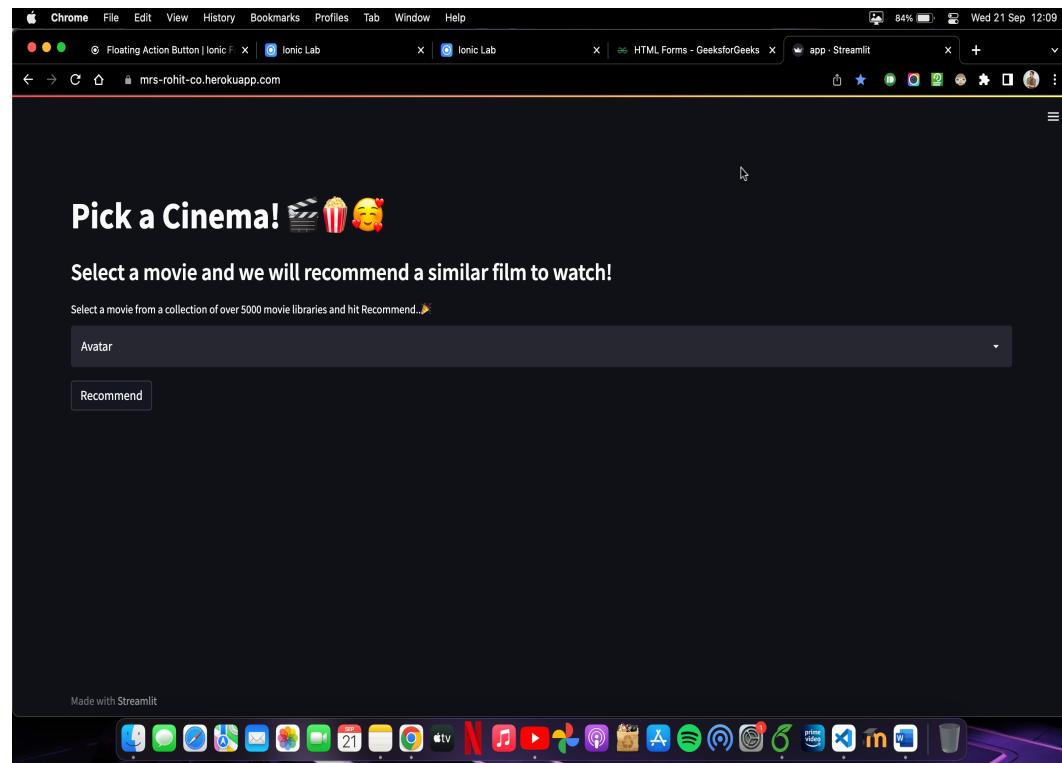
Table 4.2.4: Use Case Table - Recommendations

Use Case ID	4
Use Case Name	Recommendations
Actor	User
Pre-Condition	User clicks on the button "Recommend".
Post-Condition	There will be 5 similar movies suggested based on user selected movie.

Table 4.2.5: Use Case Table - View Marks

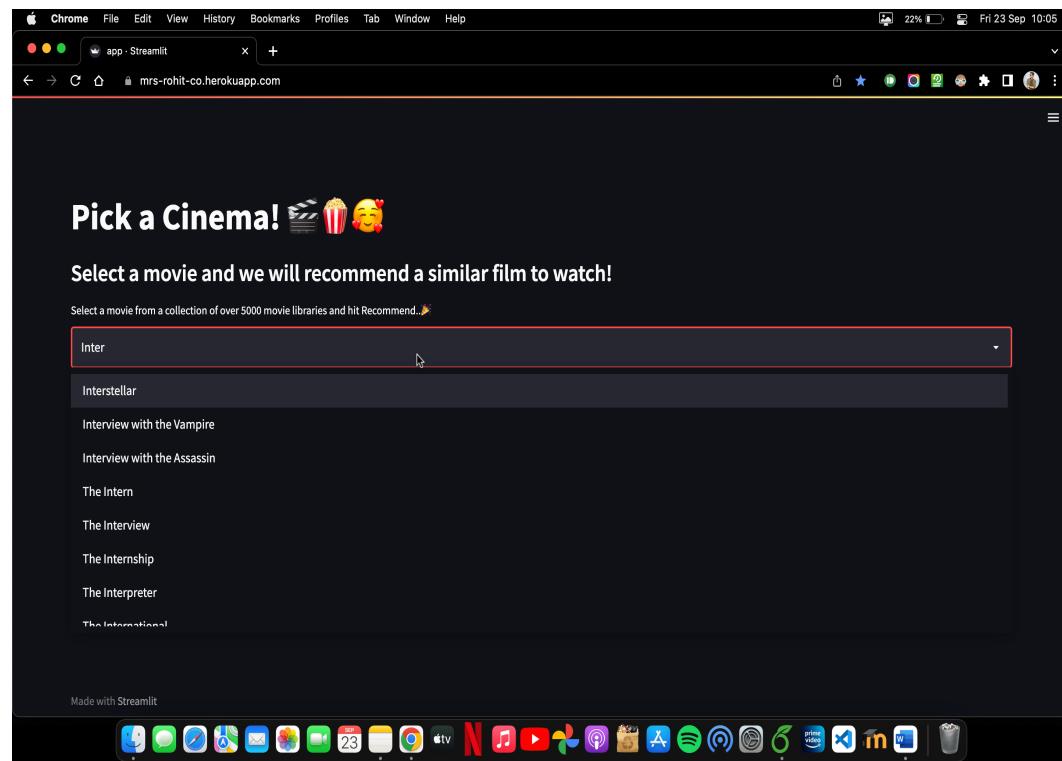
## 4 Project Implementation and Testing

### 4.1 Home - Landing View



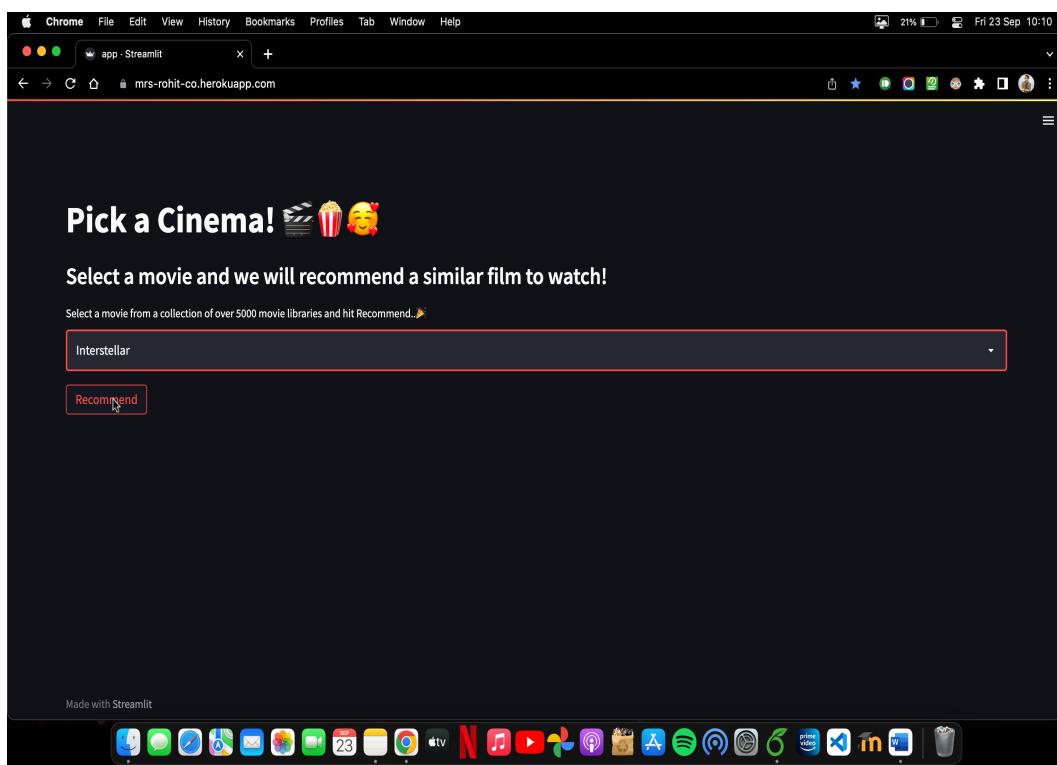
4.1.1: Home View

## 4.2 Selecting Movies



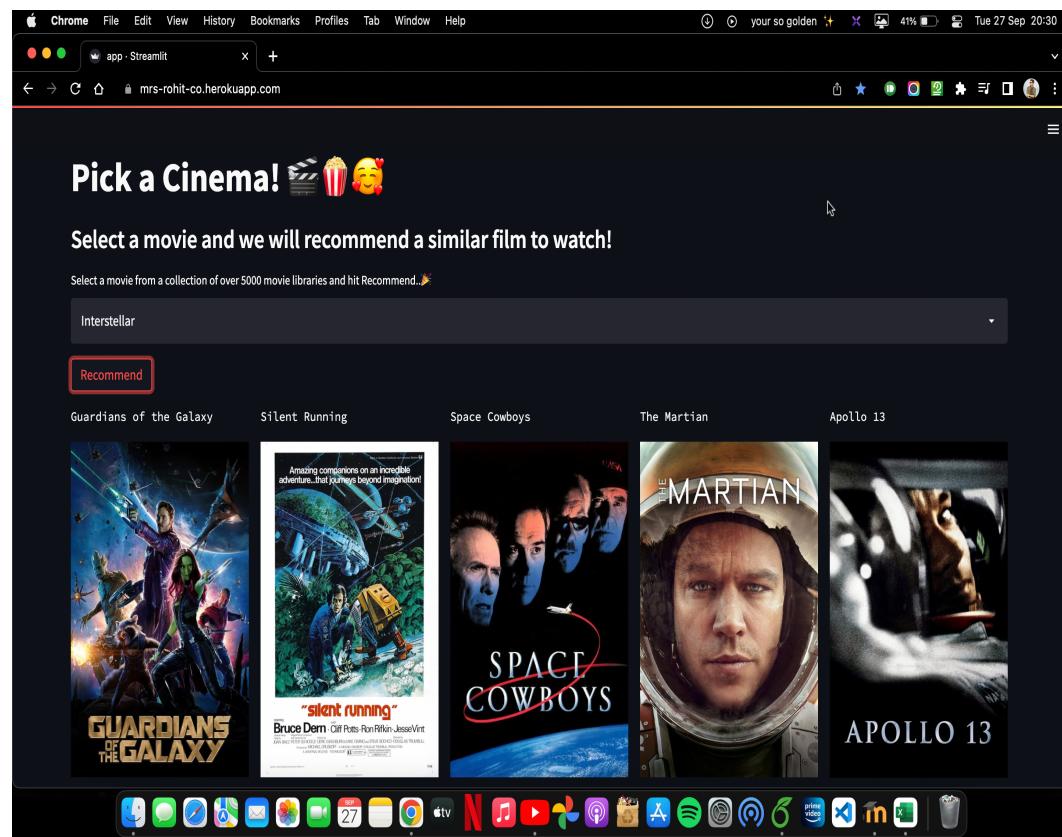
4.2.1: Selection of title

## 4.3 Recommend



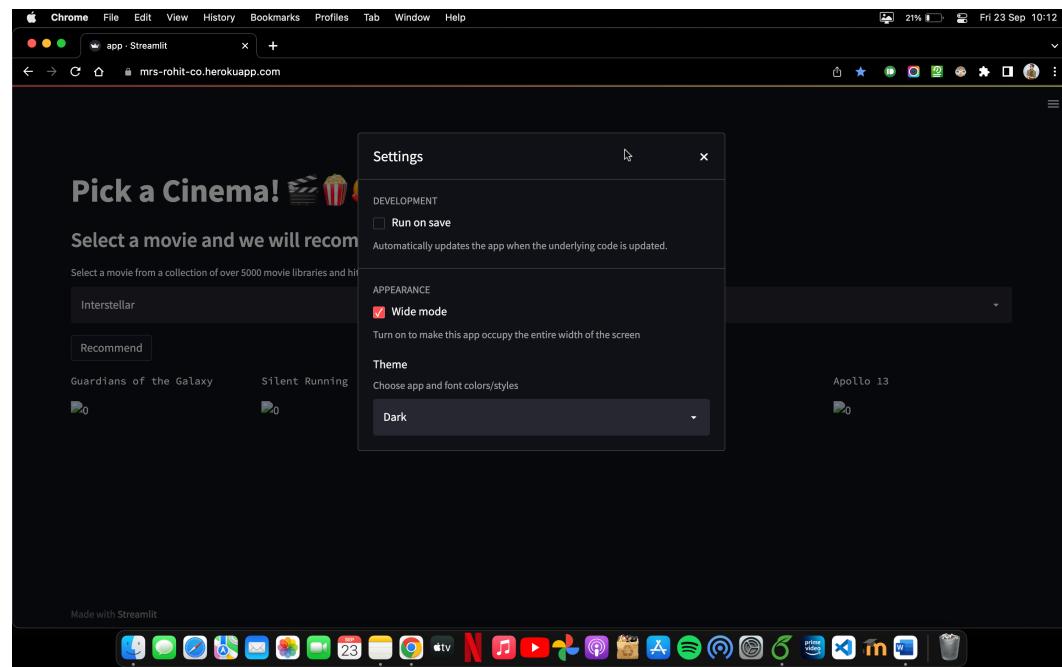
4.3.1: Hit Recommend

## 4.4 Actual Recommendations



4.4.1: 5 Movie Recommendations

## 4.5 Settings



4.5.1: Edit Settings

## Movie Recommendations System

Rohit Mahadik (UCID)

### 4.6 Code 1

The screenshot shows the PyCharm IDE interface with the project 'movies-recommendation-system' open. The code editor displays a Python script named `app.py`. The code implements a movie recommendation system using a similarity matrix and an API for movie posters. The terminal below shows a command error related to `comdef`.

```
movies-recommendation-system app.py
Project movies-recommendation-system
  venv
    -gignore
    app.py
    main.py
    movie_dict.pkl
    movie_pkls
    Profile
    requirements.txt
    setup.sh
    similarity.pkl
  External Libraries
    Python 3.10 (movies-recomm)
      Binary Skeletons
      Extended Definitions
      lib-dynload
      python3.10 library root
      site-packages
    Typeshed Stubs
  Scratches and Consoles

1 import pandas as pd
2 import requests
3 import streamlit as st
4 import pickle
5 import pandas as pd
6
7 def fetch_poster(movie_id):
8     response = requests.get("https://api.themoviedb.org/3/movie/{}?api_key=8265bd1679663a7a12c168da8d2e8&language=en-US".format(movie_id))
9     data = response.json()
10    print(data)
11    return "https://image.tmdb.org/t/p/w300/" + data['poster_path']
12
13 def recommend(movie):
14     movie_index = movies[movies['title'] == movie].index[0]
15     distances = similarity[movie_index]
16     movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]
17
18     recommended_movies = []
19     recommended_movies_posters = []
20
21     for i in movies_list:
22         movie_id = movies.iloc[i[0]].movie_id
23         recommended_movies.append(movies.iloc[i[0]].title)
24         # get poster from API
25         recommended_movies_posters.append(fetch_poster(movie_id))
26
27     recommended_movies
28     recommended_movies_posters
29
30 fetch_poster()

Terminal: Local: + ~
/dev/fd/12:18: command not found: comdef
(venv) (base) rohit74536@Rohits-MacBook-Pro movies-recommendation-system %
```

### 4.7 Code 2

The screenshot shows the PyCharm IDE interface with the project 'movies-recommendation-system' open. The code editor displays a Python script named `app.py`. This version includes a Streamlit interface for user interaction. The terminal below shows a command error related to `comdef`.

```
movies-recommendation-system app.py
Project movies-recommendation-system
  venv
    -gignore
    app.py
    main.py
    movie_dict.pkl
    movie_pkls
    Profile
    requirements.txt
    setup.sh
    similarity.pkl
  External Libraries
    Python 3.10 (Movies-recom)
      Binary Skeletons
      Extended Definitions
      lib-dynload
      python3.10 library root
      site-packages
    Typeshed Stubs
  Scratches and Consoles

13 def recommend(movie):
14     movie_index = movies[movies['title'] == movie].index[0]
15     distances = similarity[movie_index]
16     movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]
17
18     recommended_movies = []
19     recommended_movies_posters = []
20
21     for i in movies_list:
22         movie_id = movies.iloc[i[0]].movie_id
23         recommended_movies.append(movies.iloc[i[0]].title)
24         # get poster from API
25         recommended_movies_posters.append(fetch_poster(movie_id))
26
27     recommended_movies
28     recommended_movies_posters
29
30     movies_dict = pickle.load(open('movie_dict.pkl', 'rb'))
31     movies = pd.DataFrame(movies_dict)
32
33     similarity = pickle.load(open('similarity.pkl', 'rb'))
34
35     st.title("Pick a Cinema 🎬")
36     st.subheader("Select a movie and we will recommend a similar film to watch!")
37
38     selected_movie_name = st.selectbox(
39         "Select a movie from a collection of over 5000 movie libraries and hit Recommend 🚀",
40         fetch_poster()

Terminal: Local: + ~
/dev/fd/12:18: command not found: comdef
(venv) (base) rohit74536@Rohits-MacBook-Pro movies-recommendation-system %
```

## Movie Recommendations System

Rohit Mahadik (UCID)

### 4.8 Code 3

The screenshot shows the PyCharm IDE interface with the project 'movies-recommendation-system' open. The code editor displays a Python script named `app.py`. The code implements a movie recommendation interface using Streamlit. It loads movie data from `movie_dict.pkl` and similarity data from `similarity.pkl`. The Streamlit interface includes a title, a sidebar for selecting a movie, and a main area displaying recommended movies with their names and posters.

```
movies = pd.DataFrame(movies_dict)
similarity = pickle.load(open('similarity.pkl','rb'))
st.title("Pick a Cinema! 🎬")
st.subheader('Select a movie and we will recommend a similar film to watch!')
selected_movie_name = st.selectbox(
    'Select a movie from a collection of over 5000 movie libraries and hit Recommend ➤',
    movies['title'].values)
if st.button('Recommend'):
    names, posters = recommend(selected_movie_name)
    col1, col2, col3, col4, col5 = st.columns(5)
    with col1:
        st.text(names[0])
        st.image(posters[0])
    with col2:
        st.text(names[1])
        st.image(posters[1])
    with col3:
        st.text(names[2])
        st.image(posters[2])
    with col4:
        st.text(names[3])
        st.image(posters[3])
    with col5:
        st.text(names[4])
        st.image(posters[4])
```

### 4.9 Code 4

The screenshot shows the PyCharm IDE interface with the project 'movies-recommendation-system' open. The code editor displays a Python script named `app.py`, which is identical to the one in Code 3. The Streamlit interface is identical, showing a title, a sidebar for selecting a movie, and a main area displaying recommended movies with their names and posters.

```
movies = pd.DataFrame(movies_dict)
similarity = pickle.load(open('similarity.pkl','rb'))
st.title("Pick a Cinema! 🎬")
st.subheader('Select a movie and we will recommend a similar film to watch!')
selected_movie_name = st.selectbox(
    'Select a movie from a collection of over 5000 movie libraries and hit Recommend ➤',
    movies['title'].values)
if st.button('Recommend'):
    names, posters = recommend(selected_movie_name)
    col1, col2, col3, col4, col5 = st.columns(5)
    with col1:
        st.text(names[0])
        st.image(posters[0])
    with col2:
        st.text(names[1])
        st.image(posters[1])
    with col3:
        st.text(names[2])
        st.image(posters[2])
    with col4:
        st.text(names[3])
        st.image(posters[3])
    with col5:
        st.text(names[4])
        st.image(posters[4])
```

## 5 Test Cases

Table 6.1: Test Case

Test Case ID	Test Case Name	Test Data	Expected Output	Actual Output	Result
1	User enters the URL for my system	Enters the correct URL	Home Page	Home Page	Pass
2	User starts typing the title of the movie	Enters the correct title of the movie	Similar text movie titles start showing up	Similar text movie titles start showing up	Pass
3	User gets Recommendation for remaining title of the movie	My system starts suggesting remaining title of the movie	Suggestion Successful	Suggestion Successful	Pass
4	User hits the recommend button	System starts searching for similar movies	5 different movie titles has been suggested	5 different movie titles has been suggested	Pass

## 6 Limitations

- It needs internet to be accessed.
- It might take time if the movie doesn't have that much similarity between any other movies.
- No support for other than Hollywood movies.
- It does not have a feature to view the trailer of the movie.
- Movies are not categorized.

## 7 Future Enhancements

- Support for Bollywood movies as well.
- Feature of viewing trailer of any movie.
- Category wise movie recommendations.
- Support for Cast and Crew information.

## 8 User Manual

### **Part 1 – Selection**

Users are required to select a movie title from over a library of 5000+ movie titles.

### **Part 2 – Suggestions**

As the user starts typing the title of the movie, my system will start recommending the remaining letters of the movie title.

### **Part 3 – Recommend**

This will be a button that the user will have to click on in order to get similar movies.

### **Part 4 – Actual Recommendations**

Finally, my system will display 5 similar movie titles along with their posters.

## 9 Bibliography

### 9.1 Web References

- [1.] <https://jupyter.org/>
- [2.] <https://www.imdb.com/>
- [3.] <https://www.youtube.com/>
- [4.] <https://stackoverflow.com/>
- [5.] <https://www.draw.io/>
- [6.] <https://www.geeksforgeeks.org/>