**A**

**Major project report**

**On**

**“Movie Recommender System”**

Partial Fulfillment of the Requirements for the

Degree of

**Bachelor of Computer Applications (AI&ML IBM) (BCA-V sem.)**

****

**Under the guidance of Shefali Ma'am**

**Submitted by: Kunal Mudgal**

**Roll No: 21BCAN249**

**DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER APPLICATION**

**JECRC UNIVERSITY, JAIPUR December 2023**

**Acknowledgment**

With Candour and Pleasure, I take the opportunity to express my sincere thanks and obligation to my esteemed guide. It is because of his able and mature guidance and cooperation that it would not have been possible for me to complete my project.

It is my pleasant duty to thank all the staff members of the computer center who never hesitated during the project.

Finally, I gratefully acknowledge the support, encouragement, and patience of my

family, and as always, nothing in my life would be possible without God.

Thank You

**Candidate's Declaration**

I hereby declare that the project work, which is being presented in the Project Report, entitled, **Movie Recommender System** partial fulfillment for the award of Degree of " Bachelor of Computer Application" in Deptt. of Information Technology, JECRC University is a record of my own investigations carried under the guidance of Shefali Sharma. I have not submitted the matter presented in this Project Report anywhere for the award of any other Degree.

Name of Candidate: Kunal Mudgal

Signature:

Registration No.: 21BCAN249

Name(s) of Supervisors) Guide: Shefali Sharma, Countersigned by Hod I

28112122

IT & CA

|  |  |  |
| --- | --- | --- |
| **1.** | **Introduction** | **1-2** |
| **2.** | **Requirement& constraints** | **3-9** |
| **3.** | **Feasibility study** | **10-13** |
| **4.** | **Existing System** | **14-15** |
| **5.** | **Technology Used** | **17-19** |
| **6.** | **System Design** | **20-24** |
| **7.** | **DFD** | **25-31** |
| **8.** | **E-R Diagram** | **32** |
| **9.** | **Database Design** | **33-37** |
| **10.** | **Testing** | **38-40** |
| **11.** | **Coding** | **40-52** |
| **12.** | **conclusion** | **52-55** |
| **13.** | **Bibliography** | **56** |

**Introduction:**

The Movie Recommender System is an innovative project designed to revolutionize how individuals discover and enjoy movies. This system operates on an extensive dataset comprising over 5000 movies, offering users a comprehensive and diverse range of options for personalized recommendations. With the exponential growth of the entertainment industry, navigating through the vast array of available movies has become increasingly challenging for users. The Movie Recommender System addresses this issue by employing advanced algorithms and leveraging a rich dataset to suggest movies tailored to individual preferences.The Movie Recommender System is an innovative project designed to revolutionize the way individuals discover and enjoy movies. This system operates on an extensive dataset comprising over 5000 movies, offering users a comprehensive and diverse range of options for personalized recommendations. With the exponential growth of the entertainment industry, navigating through the vast array of available movies has become increasingly challenging for users. The Movie Recommender System addresses this issue by employing advanced algorithms and leveraging a rich dataset to suggest movies tailored to individual preferences.This project aims to enhance the user experience by providing a seamless platform for discovering similar movies based on specific inputs. Users can input a movie of their choice, and the system utilizes sophisticated algorithms to analyze the dataset, offering five recommendations that share thematic, genre, or stylistic similarities. This personalized approach ensures that users receive recommendations that align closely with their cinematic tastes. The extensive dataset not only enriches the recommendation process but also allows for a diverse selection, catering to a wide range of preferences. As technology continues to shape the entertainment landscape, the Movie Recommender System emerges as a valuable tool, empowering users to explore and enjoy movies in a more personalized and efficient manner.

**Requirements & Constraints :**

The Movie Recommender System is built on a robust foundation of tools and libraries, ensuring a seamless and efficient development process. To facilitate the project's data science aspects, Anaconda, a comprehensive platform for data science and machine learning, serves as the project environment. This choice allows for easy management of dependencies, version control, and reproducibility, streamlining the development workflow.

In the realm of data manipulation and analysis, essential machine learning libraries such as NumPy and Pandas play pivotal roles. NumPy provides support for efficient numerical operations, while Pandas offers a versatile data manipulation toolkit. Together, they empower the system to handle the extensive dataset of over 5000 movies with agility and precision. These libraries enable the implementation of sophisticated algorithms, allowing the system to generate accurate and personalized movie recommendations.

On the frontend, the system utilizes Streamlit, a powerful and user-friendly Python library for creating interactive web applications. Streamlit simplifies the process of transforming data insights into visually appealing and interactive interfaces, providing an intuitive user experience. Through Streamlit, users can easily input their movie preferences and receive instant recommendations in a user-friendly and accessible format.

However, despite the project's advanced technological stack, certain constraints shape its development. Time serves as a critical constraint, requiring efficient project management and timely execution. Additionally, budget constraints influence decision-making, emphasizing the need for cost-effective solutions. While the chosen technologies are powerful, their utilization is constrained by the hardware specifications of the development environment and the target deployment platform.

Furthermore, ensuring compatibility across various operating systems may pose challenges, demanding thorough testing and optimization. Balancing the desire for feature-rich functionality with the need for a streamlined and responsive system introduces a constraint in terms of system complexity.

In navigating these constraints, the Movie Recommender System project aims to strike a harmonious balance between innovation, efficiency, and practicality, ensuring a robust and user-friendly movie recommendation experience.

**Problem Analysis :**

The Movie Recommender System addresses several critical challenges prevalent in the current landscape of movie recommendation systems. Existing systems often grapple with a lack of granularity in their recommendations, providing users with generic suggestions that may not align closely with individual preferences. This leads to user dissatisfaction, reduced engagement, and a missed opportunity to cater to the diverse tastes of the audience.

Moreover, conventional recommendation systems frequently struggle to adapt to the dynamic nature of user preferences. Users' tastes and preferences evolve over time, influenced by various factors such as changing trends, moods, or life experiences. The Movie Recommender System aims to overcome this challenge by employing advanced algorithms that dynamically adapt to user behavior, ensuring that recommendations remain relevant and in tune with evolving tastes.

Another common issue faced by existing systems is the inability to capture the subtleties of user preferences beyond simple genre or popularity metrics. The Movie Recommender System leverages a comprehensive dataset of over 5000 movies to consider a multitude of factors, including thematic elements, stylistic nuances, and user-specific viewing history. This approach enhances the precision of recommendations, offering users a more refined and satisfying experience.

Furthermore, the project acknowledges the limitations of some recommendation systems in handling niche or less mainstream movies. The Movie Recommender System seeks to bridge this gap by providing a diverse range of recommendations, ensuring that users discover hidden gems and lesser-known titles that align with their unique tastes.

In summary, the Movie Recommender System addresses critical issues in existing recommendation systems, offering a solution that is granular, adaptable, and capable of catering to a diverse array of user preferences. Through advanced algorithms and a comprehensive dataset, the system aims to provide an enriched and personalized movie recommendation experience.

**Feasibility Study:**

The feasibility study for the Movie Recommender System encompasses various dimensions to assess the practicality and viability of the project.

**Technical Feasibility:**

The technical feasibility of the Movie Recommender System is well-founded, leveraging widely adopted technologies such as Anaconda, NumPy, Pandas, and Streamlit. These tools ensure a robust development environment for data science, machine learning, and user interface design. Compatibility with popular operating systems enhances the system's accessibility.

**Operational Feasibility:**

Operationally, the Movie Recommender System exhibits a high degree of feasibility. Its user-friendly interface, powered by Streamlit, simplifies user interactions, making it accessible to users with varying levels of technical proficiency. The straightforward input process and instant recommendation output enhance the overall user experience.

**Economic Feasibility:**

The economic feasibility of the project is supported by the cost-effective nature of open-source tools and libraries, such as Anaconda, NumPy, and Pandas. The use of these technologies reduces licensing expenses and promotes sustainability. Additionally, the potential benefits, including improved user engagement and satisfaction, justify the investment.

**Risk Analysis:**

Despite its feasibility, the project does face certain risks. Technological risks include potential compatibility issues across various platforms, which necessitates thorough testing. Time constraints pose a risk to the timely completion of the project, demanding effective project management strategies. The dynamic nature of user preferences may introduce uncertainties, requiring adaptive algorithms to maintain relevancy.

**Ethical Considerations:**

The Movie Recommender System adheres to ethical standards by prioritizing user privacy and data security. It ensures that user data is handled responsibly, and recommendations are generated without compromising individual privacy. The system also avoids promoting content that may be considered inappropriate or biased.

In conclusion, the Movie Recommender System demonstrates strong feasibility on technical, operational, and economic fronts. While recognizing and mitigating potential risks, the project aligns with ethical considerations, ensuring a responsible and sustainable development process.

**Existing System:**

The existing movie recommendation systems encounter several limitations that the Movie Recommender System seeks to overcome. Current systems often rely on simplistic algorithms, offering generic recommendations based on popular genres or overall user preferences. These systems struggle to provide nuanced suggestions tailored to individual tastes, leading to suboptimal user experiences.

Furthermore, many existing systems face challenges in adapting to the dynamic nature of user preferences. They may lack the sophistication to analyze evolving viewing patterns, resulting in recommendations that may not accurately reflect users' changing tastes over time. The Movie Recommender System aims to fill this gap by implementing advanced algorithms that dynamically adjust to user behavior, ensuring recommendations remain relevant and aligned with evolving preferences.

Moreover, conventional systems may neglect niche or less mainstream movies, focusing predominantly on popular titles. This limitation restricts the diversity of recommendations, limiting users to a relatively narrow set of options. The Movie Recommender System addresses this by leveraging a comprehensive dataset of over 5000 movies, enabling the inclusion of a broader range of titles, including hidden gems and less mainstream content. Through these enhancements, the Movie Recommender System seeks to provide a more sophisticated and personalized movie recommendation experience compared to existing systems.

**Technology Used :**

The Movie Recommender System leverages a sophisticated technological stack to ensure efficient data processing, analysis, and a user-friendly interface. The chosen technologies are carefully selected to accommodate the complexities of handling a vast dataset of over 5000 movies and delivering personalized recommendations seamlessly.

**Jupyter Notebook:**

The project's data exploration and analysis are conducted in Jupyter Notebook. This interactive computing environment allows for the development and execution of code in a collaborative and exploratory manner. Jupyter Notebook facilitates data cleaning, preprocessing, and initial algorithm testing, providing a flexible and iterative approach to developing the recommendation system.

**Pandas and NumPy:**

Pandas and NumPy, two essential libraries in the Python ecosystem, play a crucial role in managing and manipulating the extensive movie dataset. Pandas excels in data manipulation and analysis, providing data structures for efficient handling of tabular data. NumPy, on the other hand, is employed for numerical operations, enhancing the system's capability to process and analyze the dataset effectively. Together, they form the backbone of the data processing pipeline, ensuring the system's ability to derive meaningful insights from the dataset.

**Streamlit:**

For the frontend, the Movie Recommender System utilizes Streamlit, a Python library designed for creating interactive web applications. Streamlit simplifies the process of converting data insights into visually appealing and user-friendly interfaces. Its intuitive design allows users to input their movie preferences effortlessly, and it facilitates the display of recommendations in a clear and interactive format. Streamlit's role in the project is crucial for delivering a seamless and engaging user experience.

**Anaconda:**

Anaconda is chosen as the project's development environment, providing a comprehensive platform for data science and machine learning. It streamlines the management of dependencies, ensuring that the project's libraries and tools are consistently maintained and compatible. Anaconda contributes to the project's reproducibility and scalability, enhancing the overall efficiency of the development process.

**Conclusion:**

The Movie Recommender System's technological foundation, consisting of Jupyter Notebook, Pandas, NumPy, Streamlit, and Anaconda, synergistically integrates data processing, analysis, and frontend development. This well-rounded technological stack ensures that the system not only efficiently processes and analyzes the extensive movie dataset but also delivers a user-friendly and visually appealing interface for a seamless recommendation experience.

**System Design :**

The Movie Recommender System is designed with modularity, scalability, and user-friendliness in mind. The system architecture is divided into key components to facilitate effective data processing, analysis, and seamless user interactions.

**1. Data Processing Module:**

1.1 Jupyter Notebook:

Functionality: Jupyter Notebook serves as the primary environment for data exploration, cleaning, and preprocessing.

Integration: Integrated with Pandas and NumPy for efficient data manipulation and analysis.

1.2 Pandas and NumPy:

Functionality: Pandas for data manipulation, cleaning, and preprocessing; NumPy for numerical operations.

Integration: Collaboratively used to handle the extensive dataset of over 5000 movies, ensuring data integrity.

**2. Recommendation Algorithm Module:**

2.1 Machine Learning Algorithms:

Functionality: Advanced recommendation algorithms dynamically adapt to user behavior.

Integration: Trained on the preprocessed dataset to generate personalized movie recommendations.

**3. Frontend Development Module**:

3.1 Streamlit:

Functionality: Streamlit simplifies the creation of an interactive web interface for users to input movie preferences and view recommendations.

Integration: Seamlessly integrated with the recommendation algorithm, ensuring real-time updates.

**4. Database Module:**

4.1 SQLite Database:

Functionality: A lightweight relational database to store and retrieve movie-related data efficiently.

Integration: Interacts with Pandas for data storage and retrieval during the recommendation process.

**5. User Interface Module:**

5.1 Streamlit Interface:

Functionality: Intuitive interface allowing users to input movie preferences and view recommendations.

Integration: Connected to the recommendation algorithm to fetch and display personalized movie suggestions.

**6. Scalability Considerations:**

Modularity: Each module operates independently, facilitating scalability and future enhancements.

Parallel Processing: Utilizes the inherent parallel processing capabilities of Pandas and NumPy for efficient data handling.

**7. System Flow:**

User Input: Users input their preferred movie into the Streamlit interface.

Data Processing: Jupyter Notebook, Pandas, and NumPy preprocess and analyze the dataset.

Recommendation Generation: Machine learning algorithms generate personalized movie recommendations.

Database Interaction: SQLite database stores and retrieves relevant movie data.

User Interface Update: Streamlit updates the interface in real-time, presenting personalized recommendations to the user.

**8. Conclusion:**

The Movie Recommender System's system design ensures a cohesive and efficient flow from user input to recommendation generation, providing a seamless and personalized experience for movie enthusiasts.

**Data Flow Diagram :**

Input: Users initiate the process by submitting their preferred movie choices through the intuitive Streamlit interface. This input serves as the foundation for generating personalized movie recommendations.

Output: The result of the interaction is a personalized list of movie recommendations tailored to the user's preferences, displayed in real-time through the Streamlit interface.

Data Processing Module:

Input: Streamlit efficiently passes the user's movie preferences to the Data Processing Module, initiating the recommendation generation process.

Processes:

Jupyter Notebook: As the initial processing step, Jupyter Notebook is employed for thorough data exploration, cleaning, and preprocessing. This ensures the dataset is well-prepared for subsequent analysis.

Pandas and NumPy: These powerful libraries are utilized for in-depth data analysis and manipulation. Pandas handles data cleaning and preprocessing tasks, while NumPy supports efficient numerical operations on the extensive dataset of over 5000 movies.

Output: The culmination of these processes results in preprocessed data that is then seamlessly transmitted to the Recommendation Algorithm Module.

Recommendation Algorithm Module:

Input: The Recommendation Algorithm Module receives the preprocessed data, which serves as the foundation for generating personalized movie recommendations.

Process: Employing advanced machine learning algorithms, this module dynamically adapts to user behavior and preferences. The algorithms analyze the preprocessed data to generate accurate and personalized movie recommendations, taking into account various factors such as thematic elements, stylistic nuances, and individual viewing history.

Output: The output of this module is a set of personalized movie recommendations that perfectly align with the user's cinematic tastes. These recommendations are subsequently forwarded to the User Interface Module.

Database Module:

Input: The Database Module, utilizing SQLite, interacts seamlessly with Pandas during the recommendation generation process. This interaction is crucial for efficient data storage and retrieval, enhancing the overall system performance.

Output: The SQLite database ensures that relevant movie data is stored and retrieved promptly, contributing to the effectiveness of the recommendation algorithms and the overall system.

User Interface Module (Streamlit):

Input: The User Interface Module, specifically the Streamlit component, receives the personalized movie recommendations generated by the Recommendation Algorithm Module.

Output: Streamlit, being an interactive web application library, dynamically updates its interface in real-time. The result is a visually appealing display of personalized movie suggestions presented to the user. The user can easily explore and engage with the recommendations through this user-friendly interface.

Flow Description:

Users initiate the process by submitting their preferred movies through the Streamlit interface.

Streamlit efficiently forwards the user's movie preferences to the Data Processing Module.

Jupyter Notebook, Pandas, and NumPy collaboratively process and analyze the extensive dataset, ensuring its cleanliness and suitability for analysis.

Preprocessed data is seamlessly transferred to the Recommendation Algorithm Module, where machine learning algorithms dynamically generate personalized movie recommendations.

The Database Module interacts with Pandas to ensure efficient data storage and retrieval, contributing to the overall effectiveness of the recommendation generation process.

The personalized movie recommendations are sent back to the User Interface Module (Streamlit).

Streamlit dynamically updates its interface, presenting real-time, personalized movie suggestions to users.

In conclusion, this carefully designed and streamlined data flow ensures a harmonious integration of user interaction, data processing, recommendation algorithms, and interface presentation. The Movie Recommender System thus provides a seamless and user-centric experience, enhancing the discovery of movies tailored to individual preferences.

+------------------+ +---------------------------+ +---------------------+

| | | | | |

| User +---->| User Interface (Streamlit)|<----+ Recommendation |

| | | | | Algorithm Module |

+------------------+ +---------------------------+ +---------------------+

|

| User Input

|

+-----v-----+

| |

| Data |

| Processing|

| Module |

| |

+-----|-----+

|

| Preprocessed Data

|

+-----v-----+

| |

| Database |

| Module |

| |

+-----|-----+

|

| Stored and Retrieved Data

|

+-----v-----+

| |

| Jupyter |

| Notebook |

| |

|  |
| --- |
|  |

**Database Design:**

+---------------------+

| Movies |

+---------------------+

| MovieID (PK) |

| Title |

| Genre |

| ReleaseYear |

| ... |

+---------------------+

+---------------------+

| Genres |

+---------------------+

| GenreID (PK) |

| GenreName |

+---------------------+

**Movies Table:**

MovieID (Primary Key): A unique identifier for each movie, ensuring data integrity.

Title: Represents the title of the movie, serving as a key attribute for user recognition.

Genre: Captures the genre(s) of each movie, allowing for categorization and recommendation alignment.

ReleaseYear: Indicates the year the movie was released, providing temporal context.

... (Additional Attributes): Accommodates any other relevant attributes such as director, runtime, or language, contributing to a comprehensive movie profile.

**Genres Table:**

GenreID (Primary Key): A unique identifier for each genre, facilitating efficient association.

GenreName: Stores the name of each genre, forming the basis for categorization.

Relationships:

Movies to Genres (Many-to-Many): A movie can be associated with multiple genres, and a genre can be linked to various movies. Despite the usual intermediary table in a many-to-many relationship, this design simplifies by incorporating genre information directly into the "Movies" table.

**Explanation:**

In this database design, the "Movies" table serves as the primary repository for movie-related information. The inclusion of MovieID as a primary key ensures each movie's uniqueness, while attributes like Title, Genre, and ReleaseYear capture fundamental details about each film. Additional attributes allow for a richer movie profile, accommodating diverse information that contributes to the recommendation process.

The "Genres" table, on the other hand, focuses on the unique identification of genres through GenreID and the storage of genre names. This separation facilitates the efficient categorization and organization of movies based on their genres.

The simplified design choice to incorporate genre information directly into the "Movies" table acknowledges the absence of user-specific data tracking, aligning with the project's scope. This pragmatic approach streamlines the database structure, ensuring it remains focused on the core aspects of movies and genres without unnecessary complexity. The resulting database design serves the purpose of the Movie Recommender System effectively, providing a foundation for recommending movies based on their inherent characteristics.

**Testing :**

Testing is a crucial phase in the development of the Movie Recommender System to ensure its functionality, accuracy, and reliability. The testing process encompasses various aspects, including unit testing, integration testing, and user acceptance testing.

Unit Testing:

Individual components, such as functions within the recommendation algorithm or data processing module, undergo unit testing. This verifies that each unit of code operates as intended, identifying and rectifying any potential errors or bugs.

Integration Testing:

Integration testing evaluates the collaboration between different modules. In the Movie Recommender System, this involves validating that data seamlessly flows between the user interface, data processing, recommendation algorithm, and database modules. Any inconsistencies or communication issues are addressed to guarantee smooth interactions.

User Acceptance Testing (UAT):

UAT ensures the system meets user expectations. In the context of the Movie Recommender System, real users would input movie preferences, and the system's recommendations are evaluated for accuracy and relevance. This phase validates that the system aligns with user needs and provides a satisfying movie recommendation experience.

Comprehensive testing is integral to delivering a robust and user-friendly Movie Recommender System, assuring users of its reliability and enhancing overall user satisfaction.

**Coding:**

**App.py**

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('model/movie\_list.pkl','rb'))

similarity = pickle.load(open('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])

Jupyter Notebook Code:

**import** numpy **as** np *# linear algebra*

**import** pandas **as** pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

**import** os

**for** dirname, \_, filenames **in** os**.**walk('/kaggle/input'):

**for** filename **in** filenames:

print(os**.**path**.**join(dirname, filename))

movies = pd.read\_csv('/kaggle/input/tmdb-movie-metadata/tmdb\_5000\_movies.csv')

credits = pd.read\_csv('/kaggle/input/tmdb-movie-metadata/tmdb\_5000\_credits.csv')

movies**.**head(2)

movies**.**shape

credits**.**head()

movies **=** movies**.**merge(credits,on**=**'title')

In [ ]:

movies**.**head()

*# budget*

*# homepage*

*# id*

*# original\_language*

*# original\_title*

*# popularity*

*# production\_comapny*

*# production\_countries*

*# release-date(not sure)*

In [33]:

movies **=** movies[['movie\_id','title','overview','genres','keywords','cast','crew']]

In [8]:

movies**.**head()

**import** ast

In [34]:

**def** convert(text):

L **=** []

**for** i **in** ast**.**literal\_eval(text):

L**.**append(i['name'])

**return** L

In [35]:

movies**.**dropna(inplace**=True**)

In [36]:

movies['genres'] **=** movies['genres']**.**apply(convert)

movies**.**head()

movies['keywords'] **=** movies['keywords']**.**apply(convert)

movies**.**head()

**import** ast

ast**.**literal\_eval('[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]')

In [38]:

**def** convert3(text):

L **=** []

counter **=** 0

**for** i **in** ast**.**literal\_eval(text):

**if** counter **<** 3:

L**.**append(i['name'])

counter**+=**1

**return** L

In [39]:

movies['cast'] **=** movies['cast']**.**apply(convert)

movies**.**head()

movies['cast'] **=** movies['cast']**.**apply(**lambda** x:x[0:3])

In [41]:

**def** fetch\_director(text):

L **=** []

**for** i **in** ast**.**literal\_eval(text):

**if** i['job'] **==** 'Director':

L**.**append(i['name'])

**return** L

In [42]:

movies['crew'] **=** movies['crew']**.**apply(fetch\_director)

In [22]:

*#movies['overview'] = movies['overview'].apply(lambda x:x.split())*

movies**.**sample(5)

**def** collapse(L):

L1 **=** []

**for** i **in** L:

L1**.**append(i**.**replace(" ",""))

**return** L1

In [44]:

movies['cast'] **=** movies['cast']**.**apply(collapse)

movies['crew'] **=** movies['crew']**.**apply(collapse)

movies['genres'] **=** movies['genres']**.**apply(collapse)

movies['keywords'] **=** movies['keywords']**.**apply(collapse)

In [26]:

movies**.**head()

movies['overview'] **=** movies['overview']**.**apply(**lambda** x:x**.**split())

In [46]:

movies['tags'] **=** movies['overview'] **+** movies['genres'] **+** movies['keywords'] **+** movies['cast'] **+** movies['crew']

In [47]:

new **=** movies**.**drop(columns**=**['overview','genres','keywords','cast','crew'])

*#new.head()*

In [48]:

new['tags'] **=** new['tags']**.**apply(**lambda** x: " "**.**join(x))

new**.**head()

**from** sklearn.feature\_extraction.text **import** CountVectorizer

cv **=** CountVectorizer(max\_features**=**5000,stop\_words**=**'english')

In [50]:

vector **=** cv**.**fit\_transform(new['tags'])**.**toarray()

In [ ]:

vector**.**shape

In [51]:

**from** sklearn.metrics.pairwise **import** cosine\_similarity

In [52]:

similarity **=** cosine\_similarity(vector)

In [ ]:

similarity

In [ ]:

new[new['title'] **==** 'The Lego Movie']**.**index[0]

In [53]:

**def** recommend(movie):

index **=** new[new['title'] **==** movie]**.**index[0]

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

**for** i **in** distances[1:6]:

print(new**.**iloc[i[0]]**.**title)

recommend('Gandhi')

**import** pickle

In [56]:

pickle**.**dump(new,open('movie\_list.pkl','wb'))

pickle**.**dump(similarity,open('similarity.pkl','wb'))

**Conclusion:**

The development and implementation of the Movie Recommender System mark a significant stride in enhancing the user experience for movie enthusiasts. This project, rooted in Python and powered by key libraries such as Pandas, NumPy, and Streamlit, manifests as an intuitive and dynamic tool for personalized movie recommendations.

**Achievements:**

User-Centric Design: The system is meticulously designed to prioritize user needs, offering an interactive web interface through Streamlit that simplifies the process of inputting movie preferences and receiving real-time recommendations.

Robust Data Processing: Leveraging Jupyter Notebook, Pandas, and NumPy, the system achieves efficient data processing, ensuring the integrity of the extensive dataset with over 5000 movies. This robust processing lays the foundation for accurate and relevant recommendation generation.

Scalability and Modularity: The modular architecture enables scalability and future enhancements. Each module, from data processing to recommendation algorithms, operates independently, providing flexibility for future iterations and updates.

**Future Considerations:**

Enhanced Recommendation Algorithms: Future iterations could explore advanced recommendation algorithms, incorporating collaborative filtering or deep learning techniques to further enhance the precision of movie suggestions.

User Engagement Metrics: Integrating user engagement metrics could offer valuable insights into the effectiveness of the recommendations, guiding future improvements and optimizations.

Extended Genre Information: Expanding genre information to cover sub-genres or more granular categories could refine the recommendation process, catering to users with specific tastes.

Overall Impact:The Movie Recommender System not only simplifies the movie discovery process but also serves as a testament to the fusion of technology and user-centric design. By seamlessly integrating data processing, machine learning, and a user-friendly interface, the system stands as a valuable tool for movie enthusiasts seeking tailored and enjoyable cinematic experiences.

In conclusion, the Movie Recommender System embodies a successful marriage of technological innovation and user-focused design, setting the stage for continued advancements in the realm of personalized movie recommendations. The project's journey underscores the potential of data-driven solutions to elevate and redefine user interactions within the realm of entertainment and beyond.

**Bibliography:**

**1. McKinney, W. (2010). "Data Structures for Statistical Computing in Python." Proceedings of the 9th Python in Science Conference.**

**2.. Code with Harry. (Year). "Python Programming Tutorials." [https://www.codewithharry.com/](https://www.codewithharry.com/)**

**3. Streamlit. (2021). "The fastest way to create apps." [https://streamlit.io/](https://streamlit.io/)**

**4.. SQLite. (2021). "A software library that provides a relational database management system." [https://www.sqlite.org/](https://www.sqlite.org/)**