PROJECT RED MIND

(A Movie Recommendation System)

A Project Report

Submitted in partial fulfilment of the Requirements for the award of

B.Tech. degree

in

Information Technology

Submitted by

Satyam Seth (1705413050)

Ankit Kumar Gupta (1705413012)

Harshit Dixit (1705413028)

Abhishek Kumar Yadav (1805413801)

Alisha Parveen (1705413011)

Under the guidance of

Prof. Girjesh Kumar Mishra
(Department of Information Technology)

At

BABU BANARASI DAS INSTITUTE OF TECHNOLOGY & MANAGEMENT

(Recognized by AICTE, Govt. of India & Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow)

AKTU College Code - 054

July 2021

CERTIFICATE

This is to certify that the Seminar entitled "Project Red Mind (A Movie Recommendation System)" is a bonafide record of the Seminar work done by **Satyam Seth** (1705413050), **Ankit Kumar Gupta** (1705413012), **Harshit Dixit** (1705413028), **Abhishek Kumar Yadav** (1805413801) and **Alisha Parveen** (1705413011) under my supervision and guidance, in partial fulfillment of the requirements for the Outcome Based Education Paradigm in Information Technology from BBDNITM, Lucknow for the academic year 2020-21

Prof. Girjesh Kumar Mishra
(Guide)

Department of Information Technology

Dr. Manuj Darbari

Head of Department
Department of Information Technology

Place: Lucknow

Date: 05-08-2021

DECLARATION

I hereby declare that the project/dissertation entitled, "Project Red Mind (A Movie Recommendation System)" was carried out and written by me/ us under the guidance of **Prof. Girjesh Kumar Mishra**, Department of Information Technology, B.B.D.I.T.M., Lucknow. This work has not been previously formed the basis for the award of any degree or diploma or certificate nor has been submitted elsewhere for the award of any degree or diploma.

Place: Lucknow
Date: 05-08-2021

Satyam Seth (1705413050)
Ankit Kumar Gupta (1705413012)
Harshit Dixit (1705413028)
Abhishek Kumar Yadav (1805413801)
Alisha Parveen (1705413011)

ACKNOWLEDGEMENT

Like most effective endeavors, preparing this project was a collaborative effort. We owe a great debt to many individuals who helped us in successful completion of this project.

We would not have completed this journey without the help, guidance and constant support and co - operation of certain people who acted as guides and friends along the way. I would like to express my deepest and sincere thanks to **Prof. Girjesh Kumar Mishra** for his valuable guidance and help. It would never be possible for me to take this project to this level without their innovative ideas and their relentless support and encouragement.

In this connection we would like to express my gratitude to my parents and friends who were constant source of inspiration during the project report. At last, we thank to Almighty for giving me the power to complete this project successfully.

Regards,

Satyam Seth 1705413050
Ankit Kumar Gupta 1705413012
Harshit Dixit 1705413028
Abhishek Kumar Yadav 1805413801
Alisha Parveen 1705413011

ABSTRACT

The purpose of the project entitled as *"***Movie Review System***"* is to computerize and recommend movies based on the genre preferences which is user friendly simple, fast, and cost – effective. It deals with the collection of various movie databases provided through movie review module. Traditionally, it was done by critics and the choices of users wasn't taken into account. The main function of the system is register and store movie details and retrieves these details as and when required, and also to use these details meaningfully System input contains user login modules, movie review modules, admin control and system design.

To tackle this issue we have proposed a fully automated system to retrieve data and present it to the user using sentient analysis.

Index

CHAPTER 1

INTRODUCTION

1.1 Introduction to movie review system:

We usually come across movie rating websites where users are not allowed to rate and comment on movies online. These ratings are provided as input to the website rating system. The admin then checks reviews, critic's ratings and displays an online rating for every movie. Here we propose an online system that automatically allows users to post reviews and stores them to rate movies based on user sentiments. The system now analyses this data to check for user sentiments associated with each comment. Our system consists of a sentiment library designed for English as well as Hindi sentiment analysis. The system breaks user comments to check for sentimental keywords and predicts user sentiment associated with it. Once the keywords are found it associates the comment with a sentiment rank. The system now gathers all comments for a particular movie and then calculates an average ranting to score it. This score is generated for every movie in the system. The system also sorts and displays top rating movies as per analysis and calculates a top ten list automatically. This provides an automated movie rating system based on sentiment analysis.

It is a user-friendly, simple, fast and cost effective system designed to recommend user their movies based on their preferences.

Objective:-

1) Define Movie
2) Recording information about the movies that are released.
3) Generating reviews.
4) Recording information related to both critic and user review.
5) Keeping record yearly of all movies.
6) Keeping information about users who register and use the recommendation system.

1.2 EXISTING SYSTEM:

Currently movies are recommended only taking in the pre-provided critic reviews. This is limited to a small group of individuals, where general audience view is often disregarded. The current system requires numerous sources, with data stores spread throughout. Often information is incomplete. Vital information and keywords amiss so it prevents the algorithm from searching movies suited to the needs of the user.

1.3 PROPOSED SYSTEM

The Movie Recommendation System is designed for an easier and smoother way to browse through to replace their existing system. The new system is to control and add the information through data scrapping which will be fully automated using API. These services are to be provided in an efficient manner, with the goal of reducing the time and resources currently required for such tasks. It will fully automated and will bind different pools of data to make it more effective.

1.4 Literature Survey

There are numerous IEEE papers which provides plan concerning the Recommender Systems. For our system we've got referred papers as follows:

A Review Classification of Recommender Systems analysis the primary paper we tend to referred maybe an analysis paper. In this have known 164 articles on recommender systems, that are a unit printed from 2001 to 2009 to know the trend of recommender systems analysis and to produce practitioners and researchers with insight and future direction on recommender systems. Conjointly we've got tacit some important points. So, a lot of researches area unit needed to be studied for this. The approaches exploitation social network analysis ought to be developed within the recommended systems as recently as social analysis has been employed in the varied applications.

1.5 Scope and Feasibility

1) Can be additional extended for generating reviews associated with the product in on-line Shops.
2) Can also be used for generating reviews for the net videos.

3) Can also be used for generating reviews associated with the universities throughout admission.
4) Canalsobeusedforgeneratingreviewsofthecandidateswithintheelection.
5) It can used in medical field to analyze the best consultant and doctors.

1.6 MODULES:

The entire project mainly consists of 3 modules, which are:

- ❖ Admin module
- ❖ Movie Review Module
- ❖ System Design

1.6.1 Admin module:

1. Dashboard: In this section, admin can view total number of users, movies according to years and the genres.
2. Movies: In this section, admin can add any movies specialization and mange according to requirement.(Add/Update).
3. Users: In this section, admin can view users detail and also have right to delete irrelevant user.
4. Contact us Queries: In this section, admin can view queries which are send by users.

Admin can also change his/her own password.

1.6.2 Movie Review Module:

1. Search-Bar: A search box or a search bar is usually a single-line text box or searches with the dedicated function of accepting user input to be searched for in a database.

2. Movie Description: Details about the plot cast and crew members, and critical reviews as well as different user reviews can be seen here.

3. Similar Movies: Similar movies based on user preferences can be seen here which are selected on the basis of sentient analysis.

1.6.3System Design:

1. User Interface Designs: User Interface (UI) focuses on making the user experience to search and browse easier, i.e. it anticipates what users might need to do making easier to

use, access and understand.

2. Data Storage: Data Storage essentially means that files and documents are recorded digitally

and saved in a storage system for future use.

CHAPTER 2

REQUIREMENT-SPECIFICATION

2.1 WHY WE NEED SPECIFICATIONS:

To be used efficiently, all computer software needs certain hardware components or the other software resources to be present on a computer. These pre-requisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements. Still, we have defined the minimum requirements that are needed to accomplish this project.

2.2 HARDWARE REQUIREMENTS:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

HARDWARE REQUIREMENTS FOR PRESENT PROJECT:

PROCESSOR : Intel dual Core,i3

RAM : 1GB

HARDDISK : 80 GB

2.3 SOFTWAREREQUIREMENTS:

Software Requirements deal with defining software resource requirements and pre- requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

SOFTWARE REQUIREMENTS FOR PRESENT PROJECT:

OPERATINGSYSTEM	:	Windows 7/ XP/8/10
FRONTEND	:	HTML, CSS, JavaScript, Bootstrap
BACKEND	:	Python, Django
SERVER	:	Apache Tomcat Server
DATABASE	:	SQLITE3

CHAPTER 3

ANALYSIS

3.1 SOFTWARE SPECIFICATION

3.1.1 FRONT- END SPECIFICATION

I. HTML:

HTML or **Hypertext Markup Language** is the standard markup language used to create web pages.

HTML is written in the form of HTML elements consisting of *tags* enclosed in angle brackets(like <html>). HTML tags most commonly come in pairs
like<h1> and </h1>, although some tags represent *empty elements* and so are unpaired, for example <imp>. The first tag in a pair is the *start tag*, and the second tag is the *end tag* (they are also called *opening tags* and *closing tags*). Though not always necessary, it is best practice to append a slash to tags which are not paired with a closing tag.

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.

HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as Java Script which affect the behavior of HTML web pages.

FEATURES of HTML:

- It is easy to learn and easy to use.
- It is platform-independent.
- Images, videos, and audio can be added to a web page.
- Hypertext can be added to text.
- It is a markup language.

Why use HTML?

- It is a simple markup language. Its implementation is easy.
- It is used to create a website.
- Helps in developing fundamentals about web programming.
- Boost professional career.

II. CASCADING STYLE SHEETS (CSS):

CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation.CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, And fonts.[1]This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content.

It is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their own computer, to override the one the author has specified. However if the author or the reader did not link the document to a specific style sheet the default style of the browser will be applied.

Features of CSS:

- Solves a big problem
- Saves a lot of time

- Provide more attributes
- Pages load faster
- Easier Website maintenance
- Multiple device compatibility

Why use CSS?

CSS saves time: You can write CSS once and reuse the same sheet in multiple HTML pages.

Easy Maintenance : To make a global change simply change the style, and all elements in all the webpages will be updated automatically.

Search Engines: CSS is considered a clean coding technique, which means search engines won't have to struggle to "read" its content.

Superior styles to HTML: CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.

Offline Browsing: CSS can store web applications locally with the help of an offline cache. Using this we can view offline websites.

III. JAVASCRIPT:

JavaScript is the scripting language of the Web. All modern HTML pages are using JavaScript. A scripting language is a lightweight programming language. JavaScript code can be inserted into any HTML page, and it can be executed by all types of web browsers. JavaScript is easy to learn.

WHY TO USE JAVASCRIPT:

JavaScript is one of the 3 languages all web developers must learn:

1.61.1 HTML to define the content of web pages
1.61.2 CSS to specify the layout of web pages
1.61.3 JavaScript to specify the behavior of web pages

Example

x = document.getElementById("demo"); //Find the HTML element with id="demo" x.innerHTML = "Hello JavaScript"; //Change the content of the HTML element

document.getElementById() is one of the most commonly used HTML DOM methods.

JAVASCRIPT CODE:

- JavaScript code (or just JavaScript) is a sequence of JavaScript statements.
- Each statement is executed by the browser in the sequence they are written.
- This example will manipulate two HTML elements:
- Example
- document.getElementById("demo").innerHTML="Hello Dolly"; document.getElementById("myDIV").innerHTML="How are you?";

JAVASCRIPT PROPERTIES:

- Properties are the values associated with a JavaScript object.
- A JavaScript object is a collection of unordered properties.
- Properties can usually be changed, added, and deleted, but some are read only.

IV. BOOTSTRAP*:*

WHAT IS BOOTSTRAP?

Bootstrap is the most popular **CSS Framework** for developing responsive and mobile-first websites.

Bootstrap is a potent front-end framework used to create modern websites and web apps. It's open-source and free to use, yet features numerous HTML and CSS templates for UI interface elements such as buttons and forms. Bootstrap also supports JavaScript extensions.

FEATURES of BOOTSTRAP:

Software engineers use Bootstrap for a number of different reasons:

- It is easy to set up and master
- It has a lot of components and a good grid system.
- Styling for many HTML elements ranging from typography to buttons, as well as support of JavaScript plugins, makes it very flexible.

WHAT ARE THE USES OF BOOTSTRAP:

- Bootstrap is great for creating layouts, as its responsive CSS is designed to conform to different devices.
- It can be employed to ensure consistency, eliminate cross-browser issues, and so on.

3.1.2 BACK-END SPECIFICATION

I. PYTHON:

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

What is Python used for:

- Web development (server-side),
- Software development,
- Mathematics,
- System scripting.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.

- Python can be used for rapid prototyping, or for production-ready software development.

Features of Python:

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

Python can be treated in a procedural way, an object-oriented way or a functional way.

Example

```
print("Hello, World!")
```

Output

```
Hello, World
```

II. Django:

Django is a Python-based web framework that allows you to quickly create efficient web applications. It is also called batteries included framework because Django provides built-in features for everything including Django Admin Interface, default database – SQLlite3, etc. When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use and that too for rapid development.

Feature of Django:

- Secure
- Scalable
- Fully loaded
- Versatile
- Open Source
- Vast and Supported Community

Why use Django?

- Excellent documentation and high scalability.
- Used by Top MNCs and Companies, such as Instagram, Disqus, Spotify, YouTube, Bitbucket, Dropbox, etc. and the list is never-ending.
- Easiest Framework to learn, rapid development and Batteries fully included.
- The last but not least reason to learn Django is Python, Python has huge library and features such as Web Scrapping, Machine Learning, Image Processing, Scientific Computing, etc. One can integrate it all this with web application and do lots and lots of advance stuff.

3.1.3 SERVER

I. APACHE TOMCAT SERVER

Apache Tomcat server is a **web container**. It allows the users to run **Servlet and JAVA** Server Pages that are based on the **web-applications**. It can be used as the HTTP server. The performance of the Tomcat server is not as good as the designated web server. It can be used as **separate product** with its own internal Web-server. It can also be used as **mutually with the others Web-servers** which include Apache, Microsoft Internet Information Server, and Microsoft Personal Web-server.

Features of Apache Tomcat Server:

- Act as a web server rendering static pages.
- Securing by SSL.
- Running server in a virtual IP or in a particular IP in dual NIC environment.
- Access control over directories and files in server root (whether to be accessible through HTTP)
- Executing servlets.
- Executing other dynamic pages (.jsp, .js, etc.,).

Why use Apache Tomcat Server:

- Tomcat acts only as a Web server and Servlet container.
- It doesn't provide the full feature set from the Java EE, but that isn't necessarily a disadvantage.
- You can use Apache Tomcat for production applications that process thousands of
- requests if the features it provides is enough.
- In any case, Tomcat is a production-ready tool.

3.1.4 DATABASE

I. SQLite3

Databases offer numerous functionalities by which one can manage large amounts of information easily over the web and high-volume data input and output over a typical file such as a text file.

SQL is a query language and is very popular in databases. Many websites use MySQL. SQLite is a “light” version that works over syntax very much similar to SQL.

SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine on the world wide web. Python has a library to access SQLite databases, called sqlite3, intended for working with this database which has been included with Python package since version 2.5.

FEATURES OF SQLite3:

2. Serverless
3. Self-Contained
4. Zero-Configuration
5. Transactional
6. Single-Database

Internals and portability:

- Written in C and C++.
- Tested with a broad range of different compilers.
- Works on many different platforms.
- Tested with Purify (a commercial memory leakage detector) as well as with Val grind, a GPL tool.
- Uses multi-layered server design with independent modules.

Security:

- A privilege and password system that is very flexible and secure, and that enables host-based verification.

- Password security by encryption of all password traffic when you connect to a server.

WHY TO USE SQLite3:

- Leading open source RDBMS
- Ease of use – No frills
- Fast
- Robust
- Security
- Multiple OS support
- Free
- Technical support
- Support large database– up to 50 million rows, file size limit up to 8 Million TB

CHAPTER 4

Data Analysis

4.1 Data Collection

To collect movie data here we are using Web API and web scrapping.

API

An **application programming interface** (**API**) is a connection between computers or between computer programs. It is a type of software interface, offering a service to other pieces of software.[1] A document or standard that describes how to build such a connection or interface is called an *API specification*. A computer system that meets this standard is said to *implement* or *expose* an API. The term API may refer either to the specification or to the implementation.

4.1.1 Web-Scrapping

Web scraping is the process of using bots to extract content and data from a website.

Unlike screen scraping, which only copies pixels displayed onscreen, web scraping extracts underlying HTML code and, with it, data stored in a database. The scraper can then replicate entire website content elsewhere.

Web scraping is used in a variety of digital businesses that rely on data harvesting. Legitimate use cases include:

- Search engine bots crawling a site, analysing its content and then ranking it.
- Price comparison sites deploying bots to auto-fetch prices and product descriptions for allied seller websites.
- Market research companies using scrapers to pull data from forums and social media (e.g., for sentiment analysis).

4.1.2 TMDB API

TMDB is a site that provides a free API portal for researchers who are interested in getting access to
movie data. It is free to register for an account and get a free personal API key (an API key is a crucial
piece of information for accessing any API portal, think about it like a digital ticket for an event).

Movie Collection:

For collecting the data of Bollywood movies we are using python modules such as:

- requests: Requests library is an integral part of Python for making HTTP requests to a specified URL. Whether it be REST APIs or Web Scrapping, requests is must to be learned for proceeding further with these technologies. When one makes a request to a URI, it returns a response.
- html5lib: A pure-python library for parsing HTML. It is designed to conform to the WHATWG HTML specification, as is implemented by all major web browsers.
- bs4: BeautifulSoup object is provided by Beautiful Soup which is a web scraping framework for Python. Web scraping is the process of extracting data from the website using automated tools to make the process faster.

And for collecting the data of Hollywood movies we are using the TMDB API:

How to process the API data:

- Before accessing an API, the first step is to read the API documentation; this step is crucial
- because in order to know what data to pull, how to pull it and how to unpack it.
- We need to use the "get movie detail" query to pull the movie detail for each movie.
- By substituting "API key" and providing a movie id of interest, we can form an API query.
- JSON is the format most API portals return their data in, so we need a way to extract the data

we need from this jiberish and put it in a format we can easily digest as humans (say CSV).

- All JSON returns from API portals can be treated as a dictionary when used in Python.
- Finally, start by important all the packages that we need
- Next, use the *requests* package to query the API; this step will return the result in JSON format.
- The last step is to unpack the result and write it to a CSV file.

4.2 Data Pre-Processing

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean

data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

- After getting our initial data from web scraping, where we have used Wikipedia
- The data here is further pre - processed using panda and numpy.

Pandas: It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool,
built on top of the Python programming language.

Numpy: NumPy stands for Numerical Python. NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

4.2.1 How the data is processed:

- The data in the system has to be stored and retrieved from the database.
- Data elements and data structures to be stored have been identified at the analysis stage. They are structured and put together to design the data storage and retrieval system.

Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data storage required, minimizing chances of data inconsistencies, and optimizing for updates.

CHAPTER 5

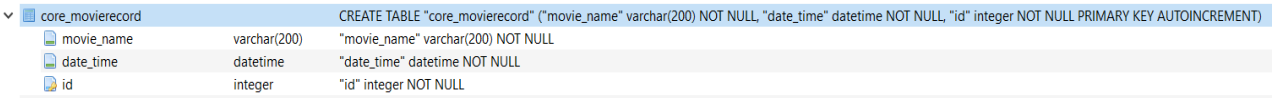
DATABASE DESIGN

The data in the system has to be stored and retrieved from database. Designing the database is part of system design. Data elements and data structures to be stored have been identified at analysis stage. They are structured and put together to design the data storage and retrieval system.

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make database access easy, quick, inexpensive and flexible for the user. Relationships are established between the data items and unnecessary data items are removed. Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data storage required, minimizing chances of data inconsistencies and optimizing for updates. The MS Access database has been chosen for developing the relevant databases.

Movie Recommendation System (MRS) SQLite3 tables:

Table Schema: This table given below stores the fetched data in the form of a table.

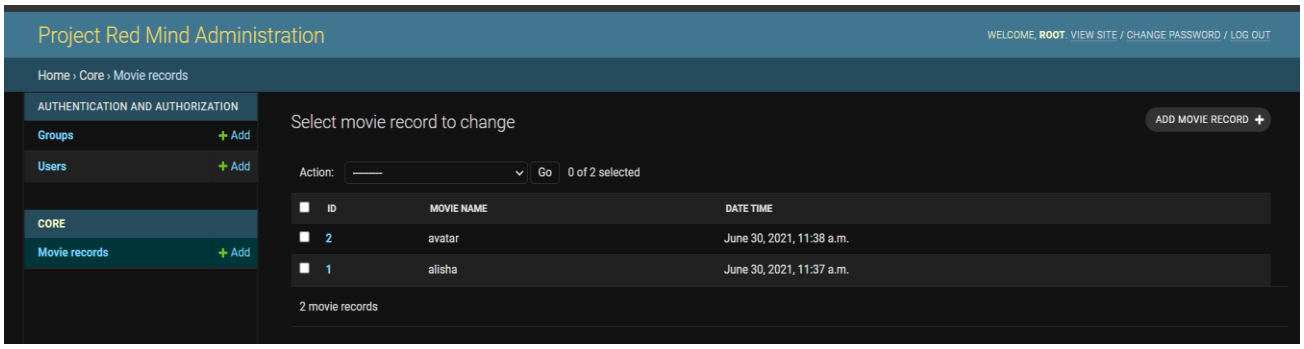


core_movierecord		CREATE TABLE "core_movierecord" ("movie_name" varchar(200) NOT NULL, "date_time" datetime NOT NULL, "id" integer NOT NULL PRIMARY KEY AUTOINCREMENT)
movie_name	varchar(200)	"movie_name" varchar(200) NOT NULL
date_time	datetime	"date_time" datetime NOT NULL
id	integer	"id" integer NOT NULL

Output: The stored data is available the admin in this format. It contains the name of move, data and time when the user has searched about it and the unique id of every user.

	movie_name ▾1	date_time	id
	Filter	Filter	Filter
1	aliens	2021-06-30 20:21:25.373149	5
2	aliens	2021-06-30 21:08:25.054511	7
3	aliens	2021-07-01 06:50:18.520276	19
4	alisha	2021-06-30 06:07:25.764193	1
5	alisha	2021-06-30 21:53:19.367339	11
6	alisha	2021-06-30 21:54:11.731697	12
7	alisha	2021-06-30 21:54:47.424911	13
8	alisha	2021-06-30 21:54:57.207116	14
9	alisha	2021-06-30 21:56:20.084373	15
10	alisha	2021-07-01 06:49:14.353061	17
11	armageddon	2021-06-30 21:10:49.500546	8
12	avatar	2021-06-30 06:08:14.345417	2
13	avatar	2021-06-30 19:42:50.493029	3
14	avatar	2021-06-30 19:57:46.688653	4
15	avatar	2021-06-30 21:07:52.874878	6

Admin Panel Output: The administrator can see all the movie fetched here, and choose to add or update data fetched here.



CHAPTER 6

Technical Aspects

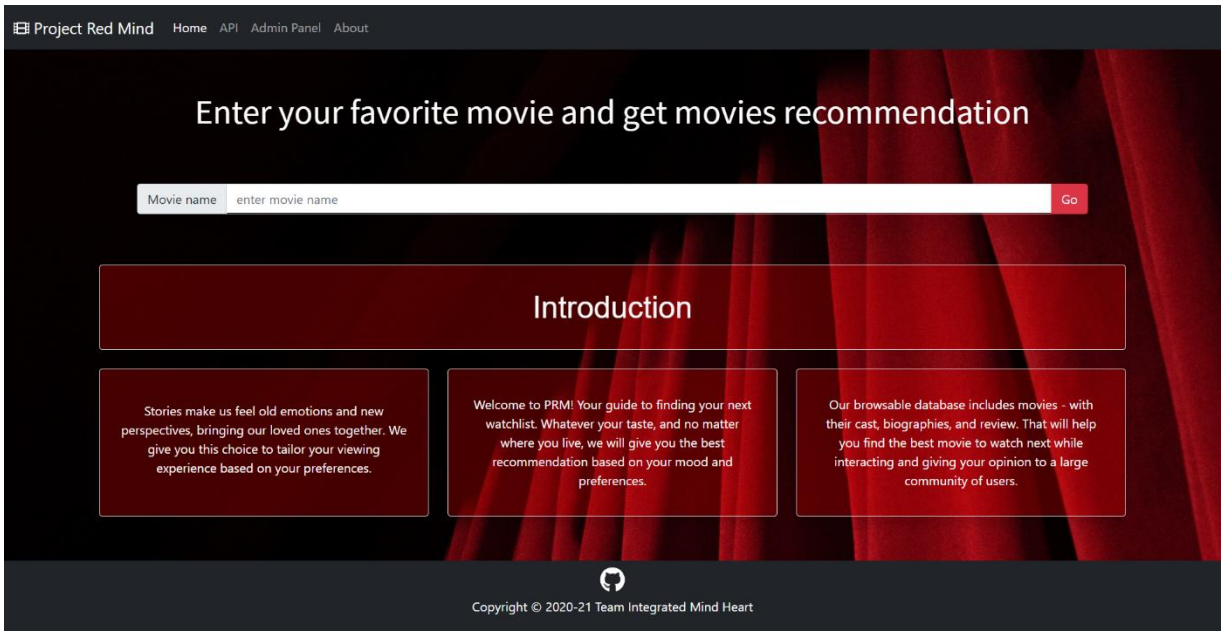
5.1 Frontend

The website is primarily designed using HTML as its base and for styling, we have used cascading style sheets (CSS), furthermore to give it a dynamic approach we have integrated javascript with it. It gives an interactive show to the user making them more engaged.

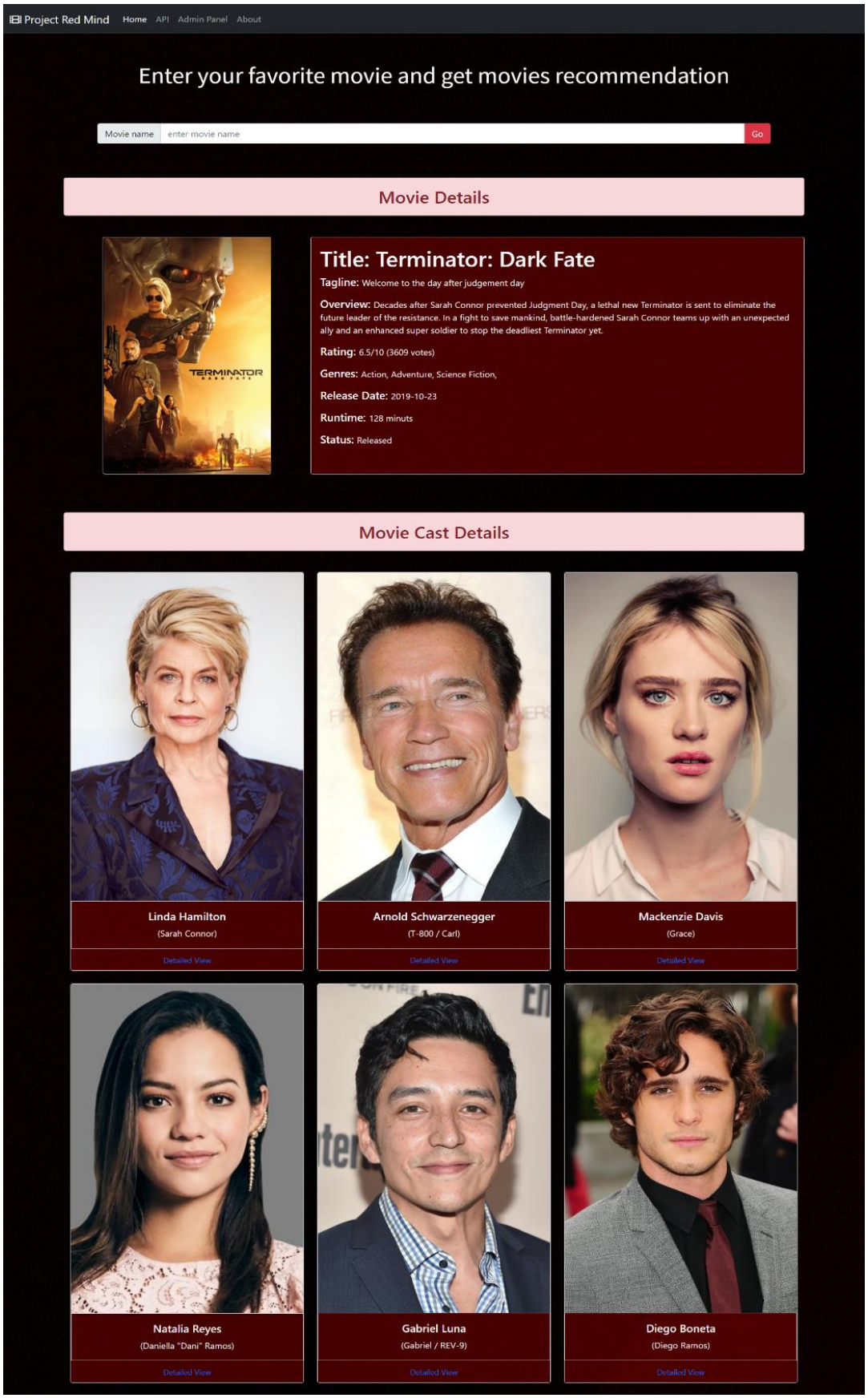
To make the code less bulky jquery is used to wrap texts and call them with a single line of code. Bootstrap makes the site more efficient and responsive by creating better UI/UX templates.

Given below are the sample screenshots of the project.

Home Page:



Searched Movie Details:

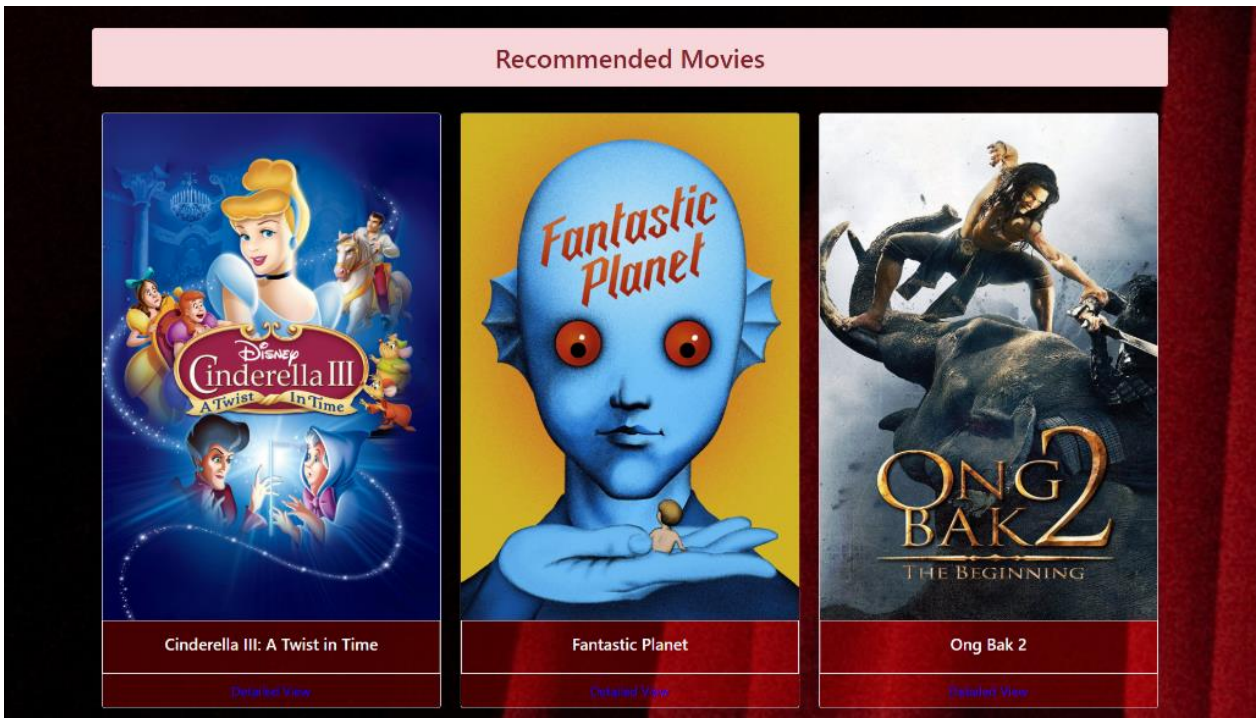


Comment Reviews and Sentient Analysis:

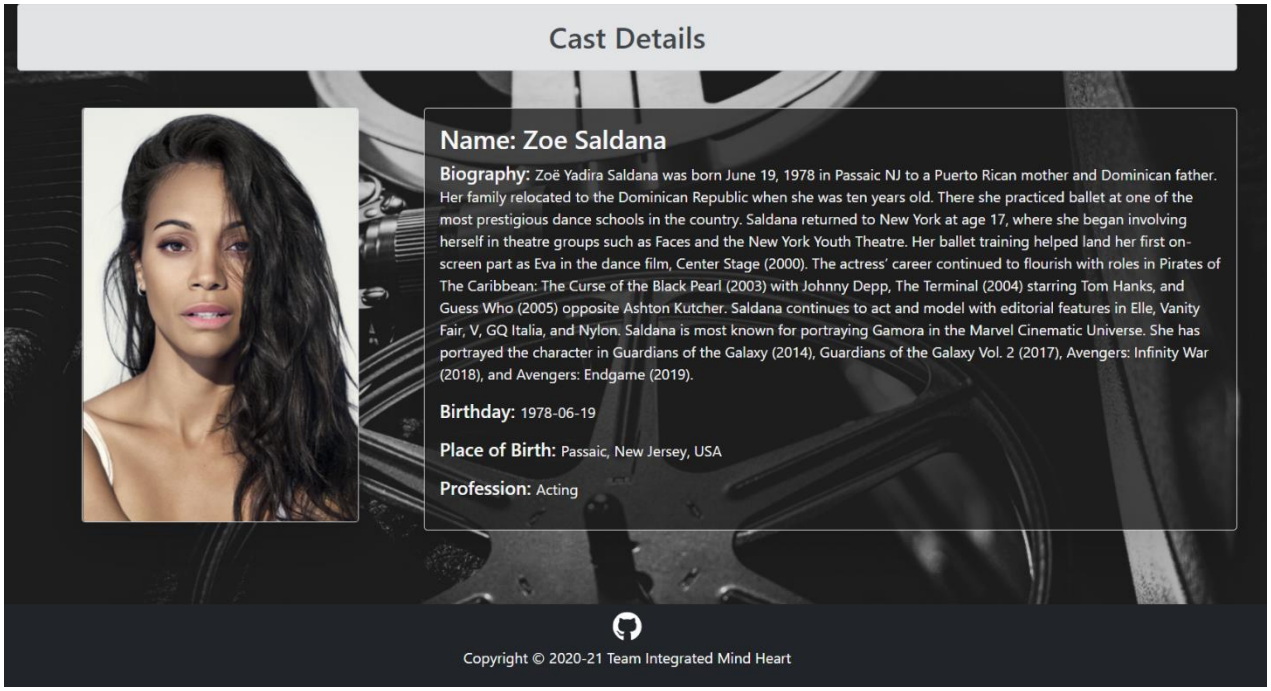
Movie Reviews

S.No	Review	Sentiment
1	I cannot even begin to describe what kind of trash so-called Terminator Dark Fate was except to say this gave Terminator a dark fate. It was a disgrace for everything we love from Terminator and the death of our hopes that some day we will see a good sequel. Then again, this is not even respecting its prequels and alters the story. You will probably like this movie if you are only interested in masculine looking women, stupid dialogue, hairy 'action' and ridiculous situations. Don't see this movie if you actually expect to see a good story, respect for Terminator, pretty girls or anything else that justifies your time or money.	(Bad)
2	What's with the good reviews? I haven't seen the originals but this movie is stupid! It doesn't deserve all the good reviews. These otakus should learn to chill the f out!	(Good)
3	John Connor is killed and Sarah Connor's character is hollowed out.	(Bad)
4	There's nothing original about this apart from killing off John Conner, which gives massive relevance to the flaws in the rest of the film, but, before that, like t2s major flaw in the t1000, lets draw attention to the fact that they very clearly state in the original film that nothing none organic can pass through the time displacement unit, ie, if it doesn't have real skin on it it won't travel back. They broke that rule in t2 straight away and again in this 1 which is unforgivable considering the importance. It's why they come through naked etc. Anyway, the John Conner death effectively changes the timeline completely. It's no longer skynet that became 'aware' but 'legion'. Not important you may say, but if the t800 (arnie) terminator has been sending Sarah Conner details of where all the other terminators sent back by SKYNET appear (also, for what reason ,given that John is dead now. Who would they hunt?) how could he possibly know where the new terminator sent back by a totally different computer with different tech from an alternate timeline would turn up to make sure Sarah Conner turns up to help grace etc? It simply makes no sense. They assume you won't think about the huge plotholes.	(Good)
5	FFS, please stop making Terminator movies already. The first 2 are masterpieces and the movies should have stopped there. Everything after T2 was crap.	(Good)
6	Starts awful ... doesn't get any better , blasphemy to the original story ark , Tim Miller should hang his head in shame , as Should James Cameron ... just watch Terminator and T2 and leave it at that !	(Bad)
7	I just came out of the cinema. I ignored all the negative talk about it and was optimistic this would be good. I mean how could they mess up a 4th time. But they did. The story made no sense, and was very poorly written. It is nothing more than a rehash of all the movies in a pot that doesn't take the story any further if anything it's Terminated it on a epic level. No spoilers from me but do yourself a favour watch T1 and T2 then go watch this. I liked bits of the movie but as a Whole this will sink Tim Miller and make James Cameron look a complete ahole! Linda, Arnold and McKenzie were good with what they had which in honesty wasn't a lot. This is an honest review by a big Terminator fan. I wanted this movie to be good so badly. But it was just very poor. I am sure once Linda watches it she'll cringe and think I should have left my legacy at T2. That movie is epic on so many levels this is not even a scratch. Sorry guys!	(Bad)

Recommended movies:



Cast Details:



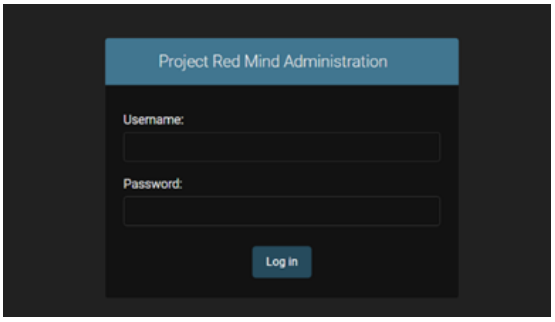
5.2 Backend

In this project, we are using the Django Web Framework. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel.

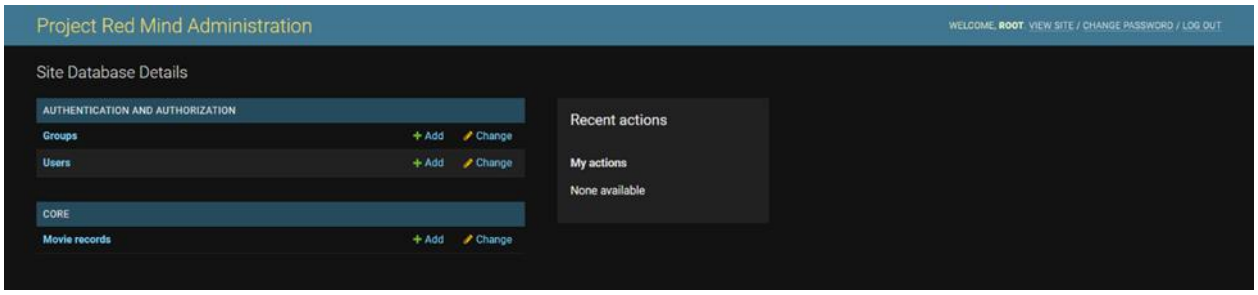
And the database part of this project is implemented using SQLite. SQLite helps to interact with relational databases. SQLite is stored as a single file. This makes sharing databases easier. By default, Django uses the SQLite database.

Here are sample screenshots of backend operations:

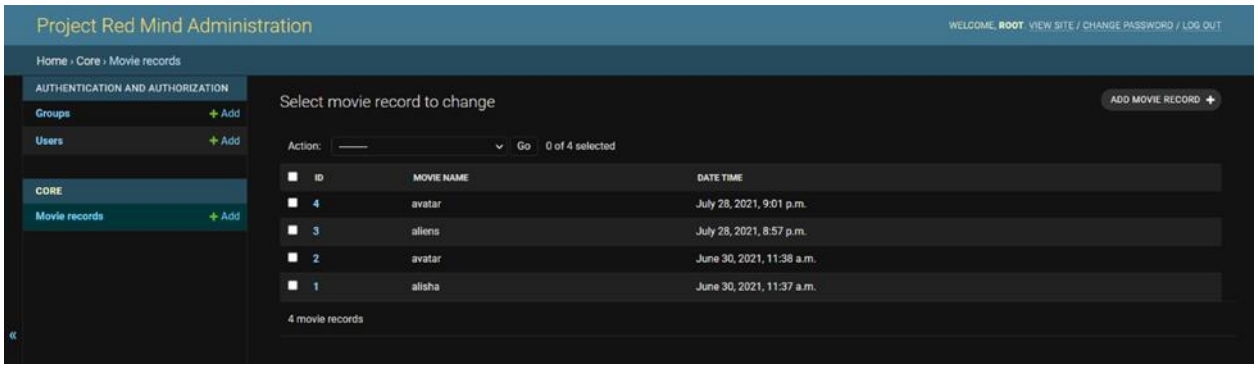
Admin Panel Login Page:



Admin Panel Page:



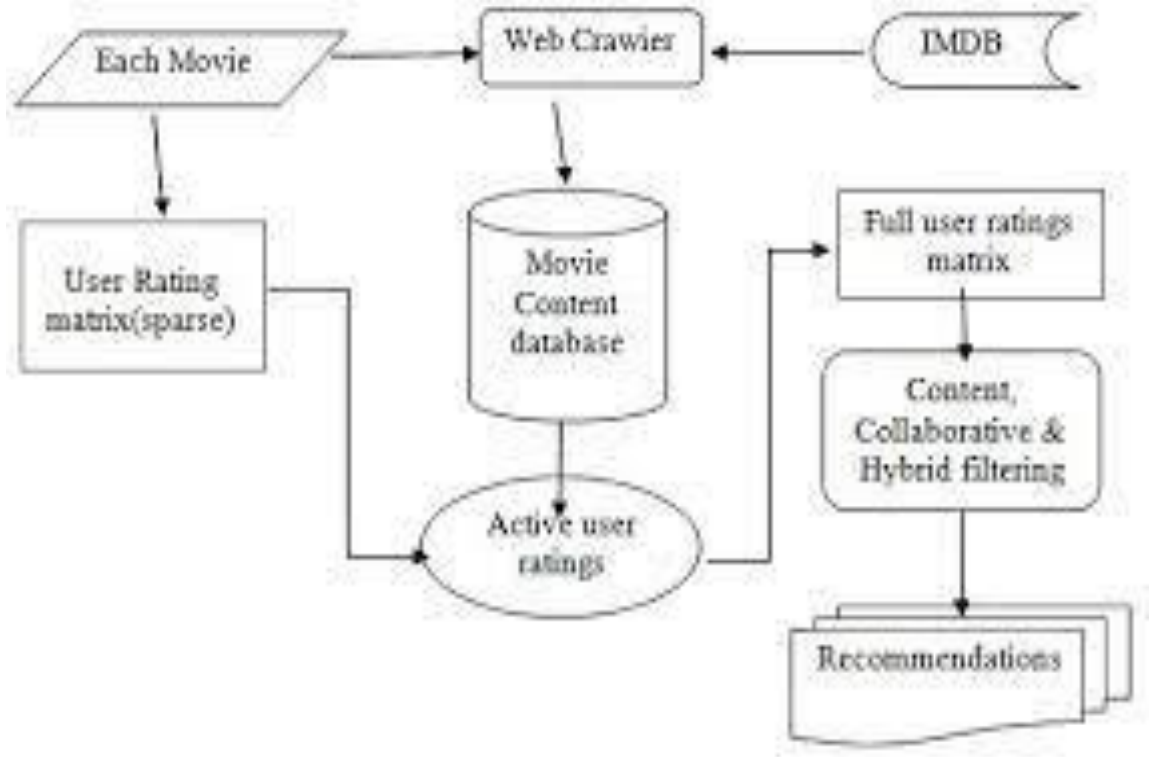
Movies Records Table Details Page:



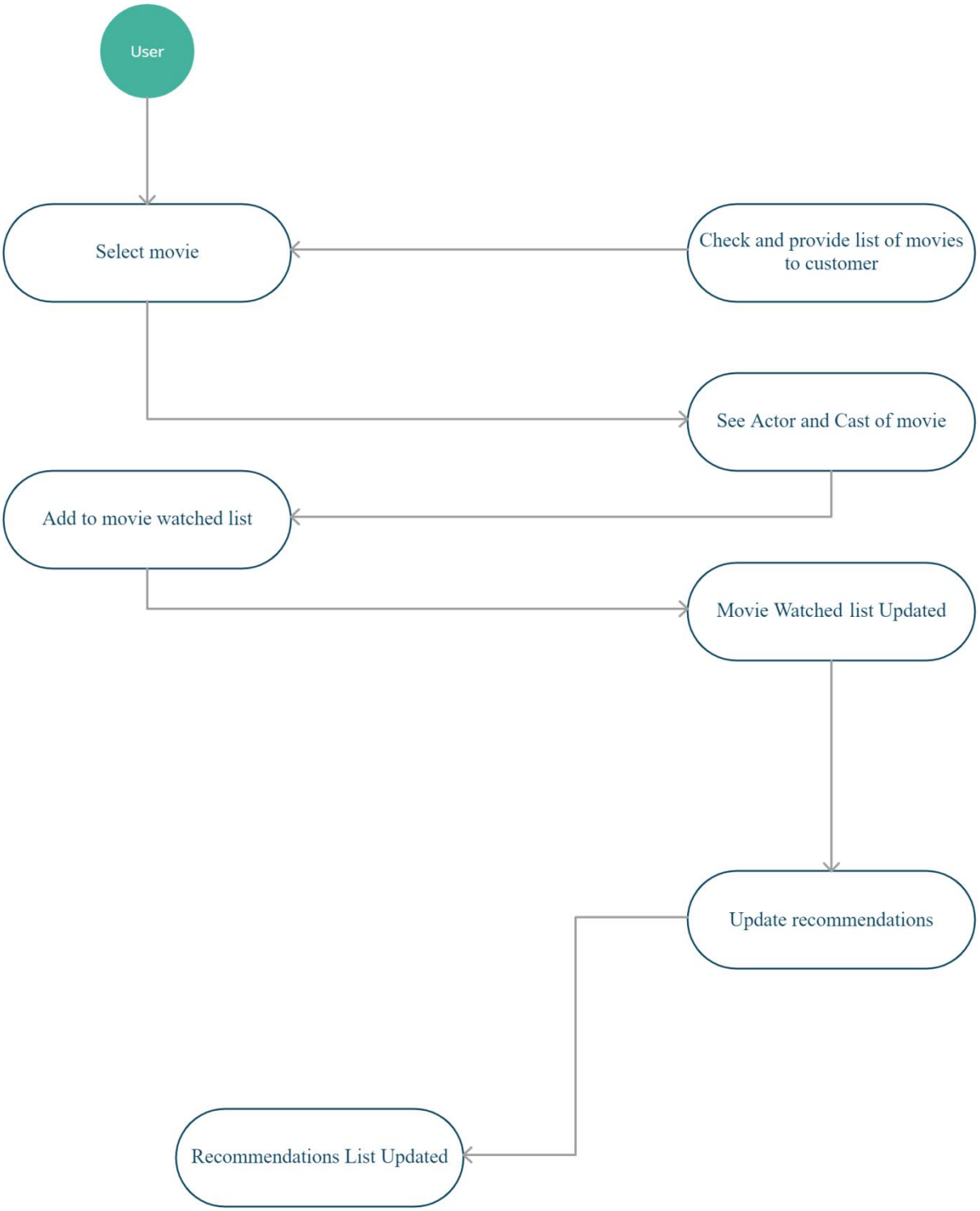
CHAPTER 7

SYSTEM DESIGN

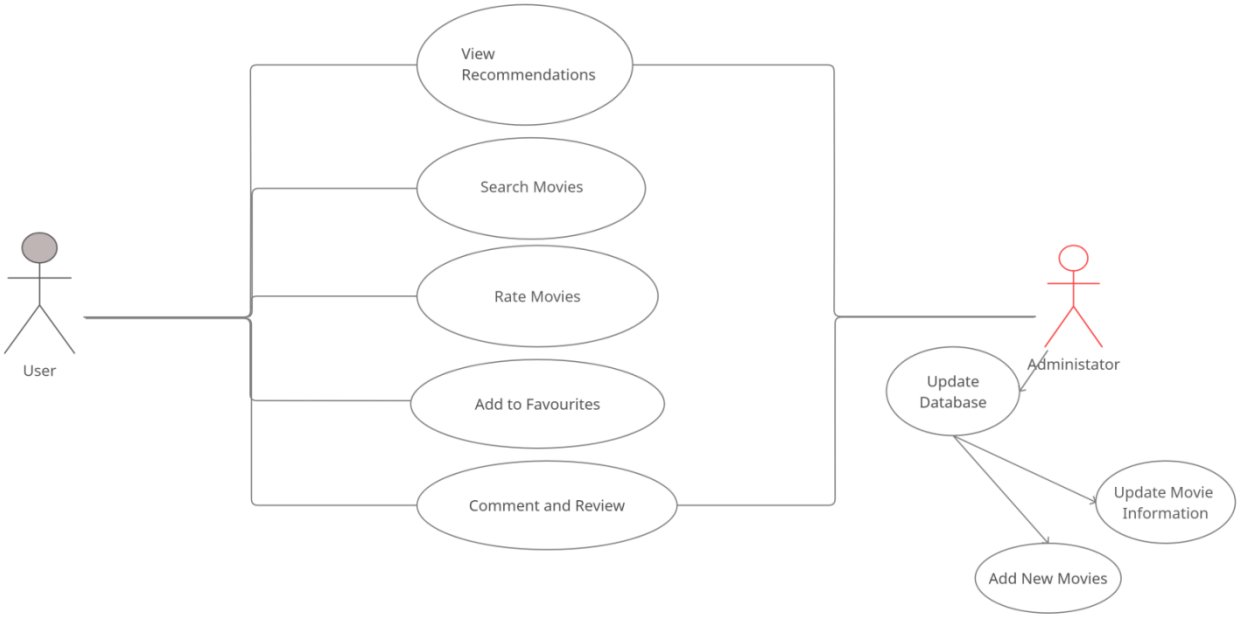
7.1 ER DIAGRAM



7.2 ACTIVITY DIAGRAM



7.3 USE CASE DIAGRAM



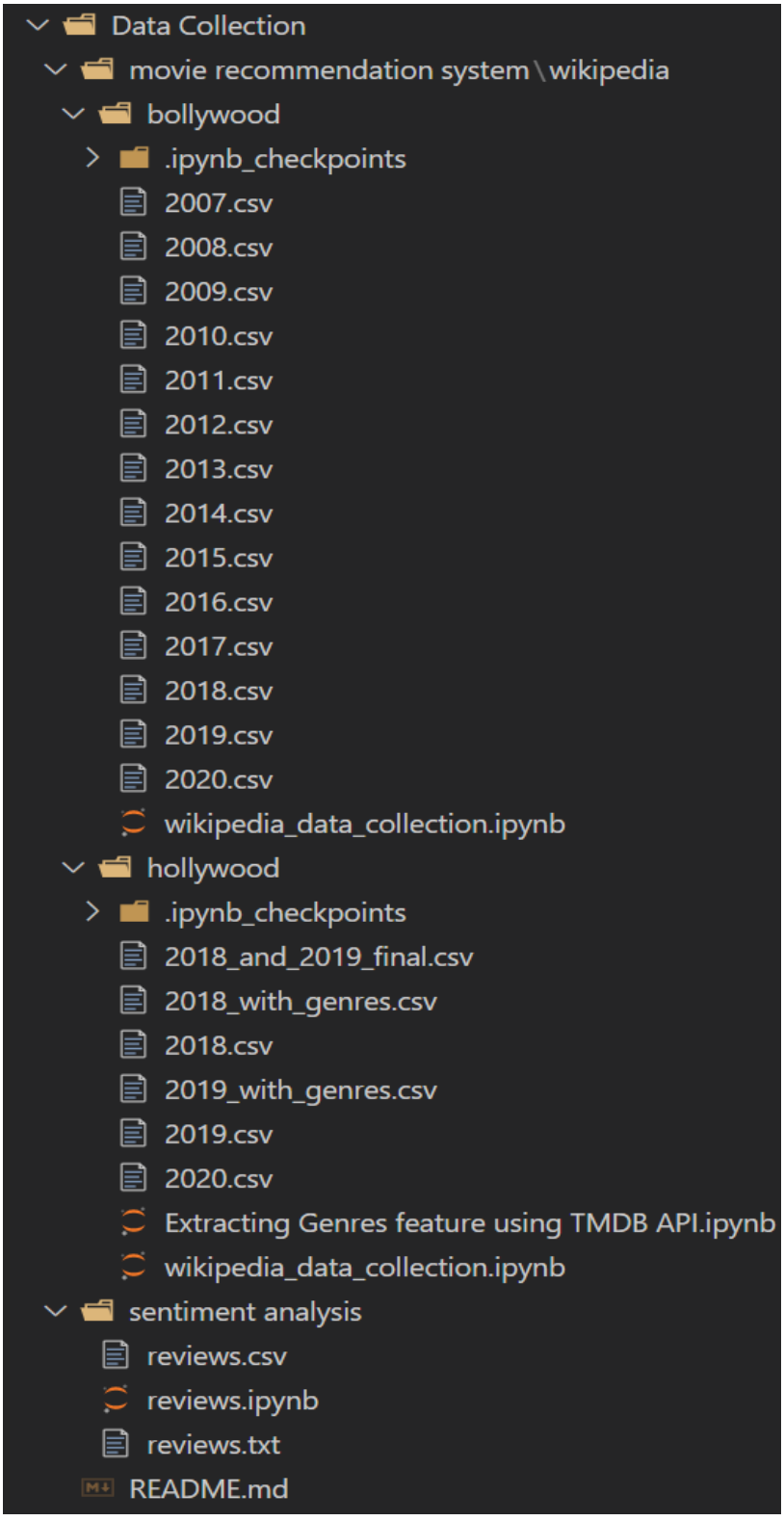
7.4 GANTT CHART

CHAPTER 8

CODE

8.1 DATA COLLECTION CODE

File Structure-



DataCollection\movierecommendationsystem\wikipedia\bollywood\wikipedia_data_collection.ipynb-

jupyter wikipedia_data_collection Last Checkpoint: 16/05/2021 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help | Not Trusted | Python 3

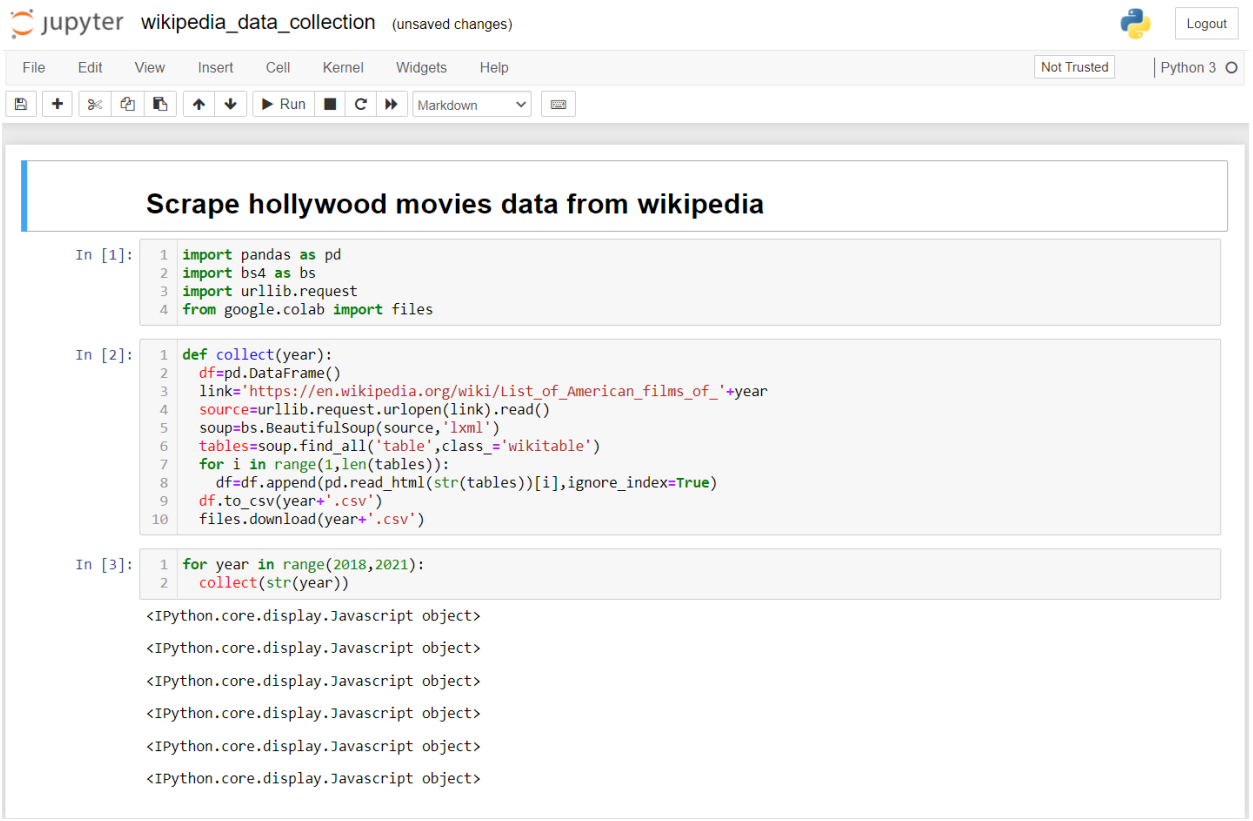
In [ ]:
```python
import pandas as pd
import bs4 as bs
import urllib.request
from google.colab import files
```

In [ ]:
```python
def collect(year):
  df=pd.DataFrame()
  link='https://en.m.wikipedia.org/wiki/List_of_Bollywood_films_of_'+year
  source=urllib.request.urlopen(link).read()
  soup=bs.BeautifulSoup(source,'lxml')
  tables=soup.find_all('table',class_='wikitable')
  for i in range(1,len(tables)):
    df=df.append(pd.read_html(str(tables))[i],ignore_index=True)
  df.to_csv(year+'.csv')
  files.download(year+'.csv')
```

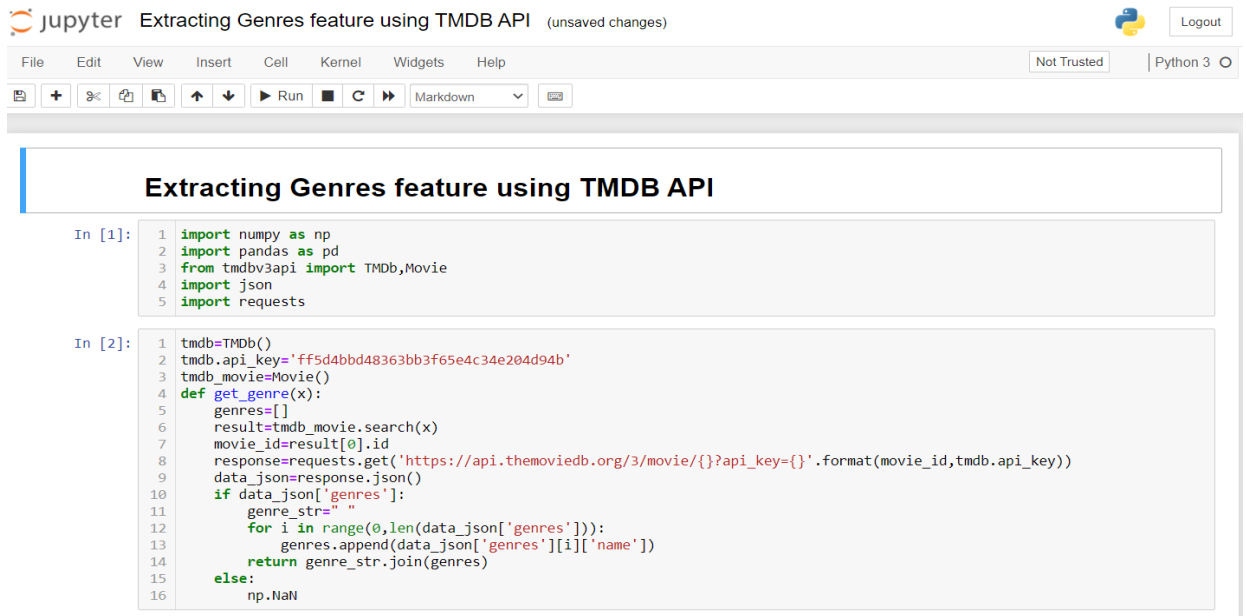
In [ ]:
```python
for year in range(2007,2021):
  collect(str(year))
```

```
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>
```

DataCollection\movierecommendationsystem\wikipedia\hollywood\wikipedia_data_collection.ipynb-



DataCollection\movierecommendationsystem\wikipedia\hollywood\Extracting Genres feature using TMDB API.ipynb-



Extracting Genres of 2018 movies

In [3]:
```
df1=pd.read_csv('2018.csv')
df1.head()
```

Out[3]:

	Unnamed: 0	Opening	Opening.1	Title	Production company	Cast and crew	Ref.	Unnamed: 6
0	0	JANUARY	5	Insidious: The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	[2]	NaN
1	1	JANUARY	5	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	[3]	NaN
2	2	JANUARY	5	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer, Warren...	[4]	NaN
3	3	JANUARY	10	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter, St...	[5]	NaN
4	4	JANUARY	12	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	[6]	NaN

In [4]:
```
df1.drop(['Unnamed: 0','Unnamed: 6','Opening','Opening.1','Ref.'],axis=1,inplace=True)
df1.head()
```

Out[4]:

	Title	Production company	Cast and crew
0	Insidious: The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...
1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...
2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer, Warren...
3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter, St...
4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...

In [5]:
```
df1['genres'] = df1['Title'].map(lambda x: get_genre(str(x)))
df1.head()
```

Out[5]:

	Title	Production company	Cast and crew	genres
0	Insidious: The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	Mystery Horror Thriller
1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	Thriller Drama
2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer, Warren...	Action Thriller
3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter, St...	Drama History Western
4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	Action Thriller

In [6]:
```
df1.to_csv('2018_with_genres.csv')
```

Extracting Genres of 2019 movies

In [7]:
```
df2=pd.read_csv('2019.csv')
df2.head()
```

Out[7]:

	Unnamed: 0	Opening	Opening.1	Title	Production company	Cast and crew	Ref.
0	0	NaN	4	Escape Room	Columbia Pictures	Adam Robitel (director); Bragi F. Schut, Maria...	[2]
1	1	NaN	4	Rust Creek	IFC Films	Jen McGowan (director); Julie Lipson (screenpl...	[3]
2	2	NaN	4	American Hangman	Hangman Justice Productions	Wilson Coneybeare (director/screenplay); Donal...	[4]
3	3	NaN	11	A Dog's Way Home	Columbia Pictures	Charles Martin Smith (director); W. Bruce Came...	[5]
4	4	NaN	11	The Upside	STX Entertainment	Neil Burger (director); Jon Hartmere (screenpl...	[6]

In [8]:
```
df2.drop(['Unnamed: 0','Opening','Opening.1','Ref.'],axis=1,inplace=True)
df2.head()
```

Out[8]:

	Title	Production company	Cast and crew
0	Escape Room	Columbia Pictures	Adam Robitel (director); Bragi F. Schut, Maria...
1	Rust Creek	IFC Films	Jen McGowan (director); Julie Lipson (screenpl...
2	American Hangman	Hangman Justice Productions	Wilson Coneybeare (director/screenplay); Donal...
3	A Dog's Way Home	Columbia Pictures	Charles Martin Smith (director); W. Bruce Came...
4	The Upside	STX Entertainment	Neil Burger (director); Jon Hartmere (screenpl...

In [9]:
```
df2['genres'] = df2['Title'].map(lambda x: get_genre(str(x)))
df2.head()
```

Out[9]:

	Title	Production company	Cast and crew	genres
0	Escape Room	Columbia Pictures	Adam Robitel (director); Bragi F. Schut, Maria...	Thriller Action Mystery Adventure Horror
1	Rust Creek	IFC Films	Jen McGowan (director); Julie Lipson (screenpl...	Thriller Drama
2	American Hangman	Hangman Justice Productions	Wilson Coneybeare (director/screenplay); Donal...	Thriller
3	A Dog's Way Home	Columbia Pictures	Charles Martin Smith (director); W. Bruce Came...	Drama Adventure Family
4	The Upside	STX Entertainment	Neil Burger (director); Jon Hartmere (screenpl...	Comedy Drama

In [10]:
```
df2.to_csv('2019_with_genres.csv')
```

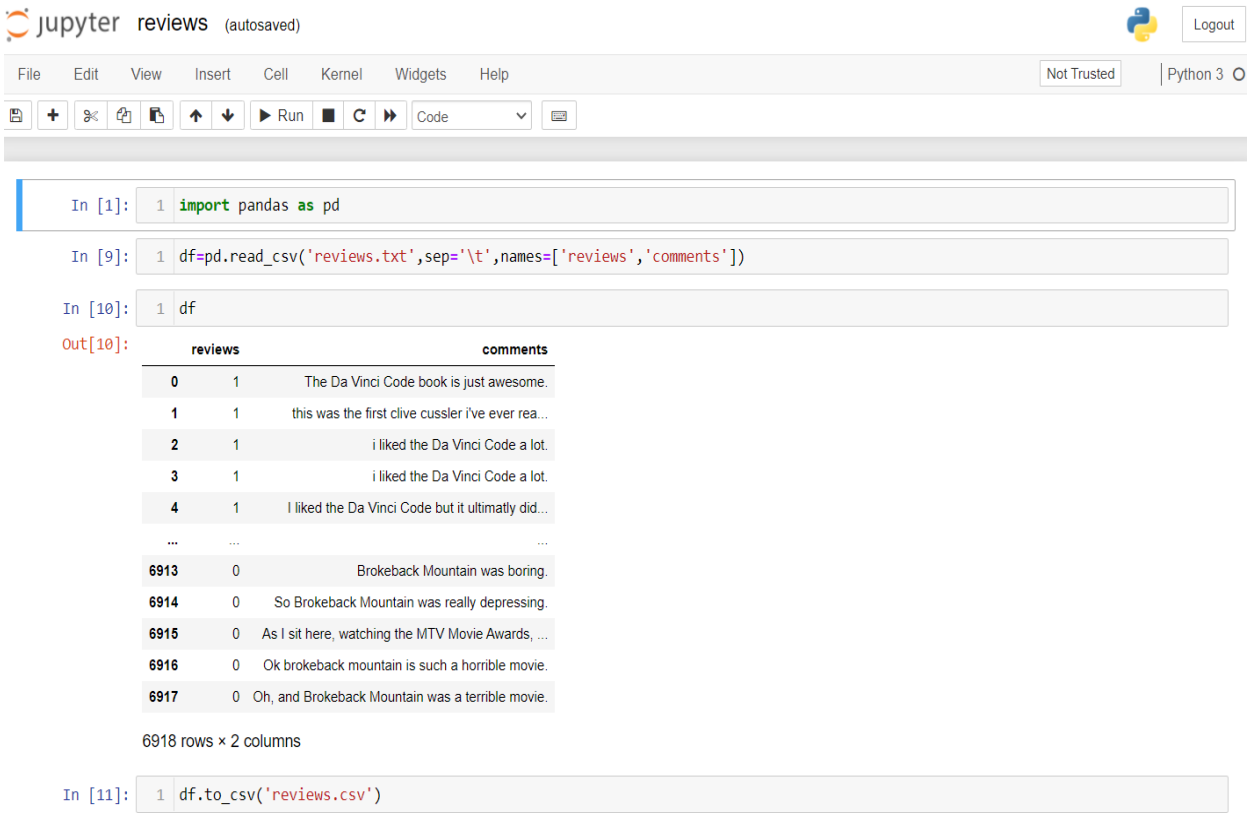
In [11]:
```
df=df1.append(df2,ignore_index=True)
df.head()
```

Out[11]:

	Title	Production company	Cast and crew	genres
0	Insidious: The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	Mystery Horror Thriller
1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	Thriller Drama
2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer, Warren...	Action Thriller
3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter, St...	Drama History Western
4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	Action Thriller

In [12]:
```
df.to_csv('2018_and_2019_final.csv')
```

DataCollection\sentiment analysis\reviews.ipynb-



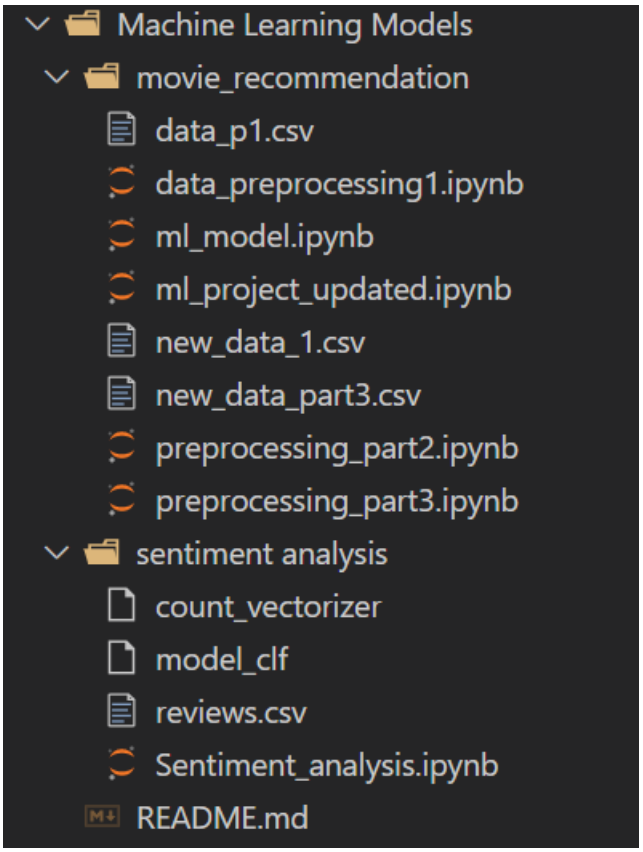
Data Collection/README.md-

 # Project Red Mind Data Collection

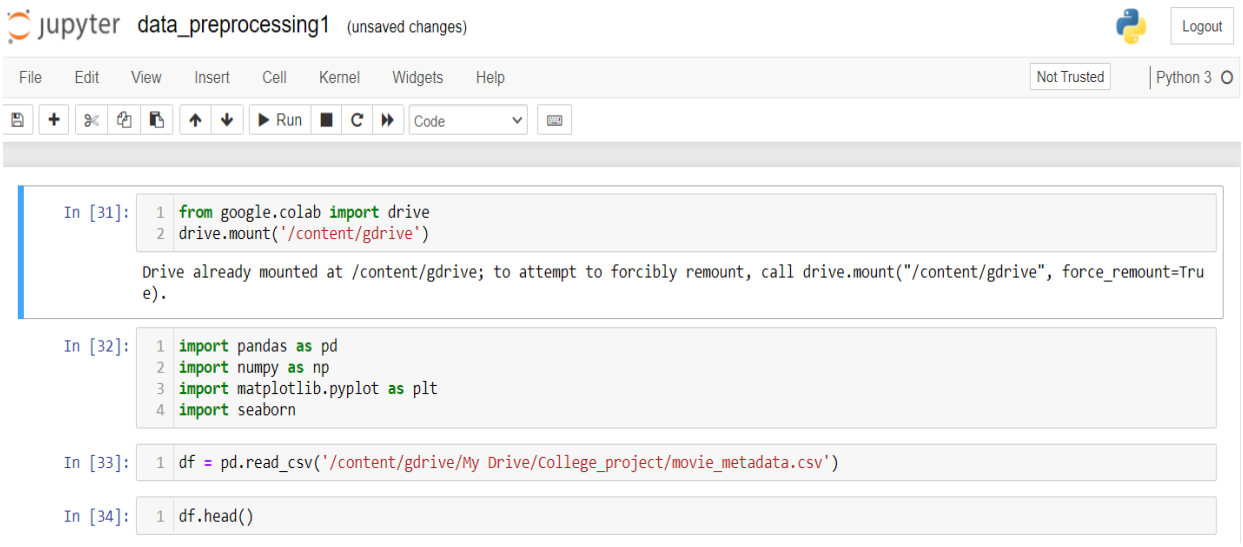
We have used web scraping to collect movies data from Wikipedia and we have also used the TM DB API to get some additional information about the movies.

8.2 MACHINE LEARING MODEL CODE

File Structure-



Machine Learning Models/movie_recommendation/data_preprocessing1.ipynb-



In [35]:
```
df.columns
```

Out[35]:
```
Index(['color', 'director_name', 'num_critic_for_reviews', 'duration',
       'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',
       'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',
       'movie_title', 'num_voted_users', 'cast_total_facebook_likes',
       'actor_3_name', 'facenumber_in_poster', 'plot_keywords',
       'movie_imdb_link', 'num_user_for_reviews', 'language', 'country',
       'content_rating', 'budget', 'title_year', 'actor_2_facebook_likes',
       'imdb_score', 'aspect_ratio', 'movie_facebook_likes'],
      dtype='object')
```

For movie Recomendation we need only few columns like

1. Director_name
2. actor_1_name
3. actor_2_name
4. actor_3_name
5. genrres
6. movie_list

In [36]:
```
df = df.loc[:,['director_name','actor_1_name','actor_2_name','actor_3_name','genres','movie_title']]
```

In [37]:
```
df.head()
```

Out[37]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action|Adventure|Fantasy|Sci-Fi	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action|Adventure|Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action|Adventure|Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action|Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [38]:
```
df['director_name'].value_counts()
```

Out[38]:
```
Steven Spielberg    26
Woody Allen         22
Clint Eastwood      20
Martin Scorsese     20
Ridley Scott        17
                    ..
Ingmar Bergman       1
Edward Hall          1
Michael Winner       1
Sara Newens          1
Kelly Makin          1
Name: director_name, Length: 2398, dtype: int64
```

In [39]:
```
df['director_name'].isna().sum()
```

Out[39]:
```
104
```

In [40]:
```
df['actor_1_name'].isna().sum()
```

Out[40]:
```
7
```

In [41]:
```
df['actor_2_name'].isna().sum()
```

Out[41]:
```
13
```

In [42]:
```
df['actor_3_name'].isna().sum()
```

Out[42]:
```
23
```

In [43]:
```
df['genres'].isna().sum()
```

Out[43]:
```
0
```

In [44]:
```
df['movie_title'].isna().sum()
```

Out[44]:
```
0
```

Observation

We look that we have in the 'director_name' feature we have 104 Nan value

- In the 'actor_1_name' feature we have 7 Nan value
- In the 'actor_2_name' feature we have 13 Nan value
- In the 'actor_3_name' feature we have 23 Nan value
- In the 'genres' feature we have 0 Nan value
- In the 'movie_title' feature we have 0 Nan value

In [45]:
```
df.head()
```

Out[45]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action|Adventure|Fantasy|Sci-Fi	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action|Adventure|Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action|Adventure|Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action|Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [46]:
```
df['genres'] = df['genres'].str.replace('|',' ')
df['genres'] = df['genres'].str.replace('Sci-Fi' , 'science friction')
```

In [47]:
```
df.head()
```

Out[47]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy science friction	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [48]:
```
df['actor_3_name'] = df['actor_3_name'].str.replace('-',' ')
```

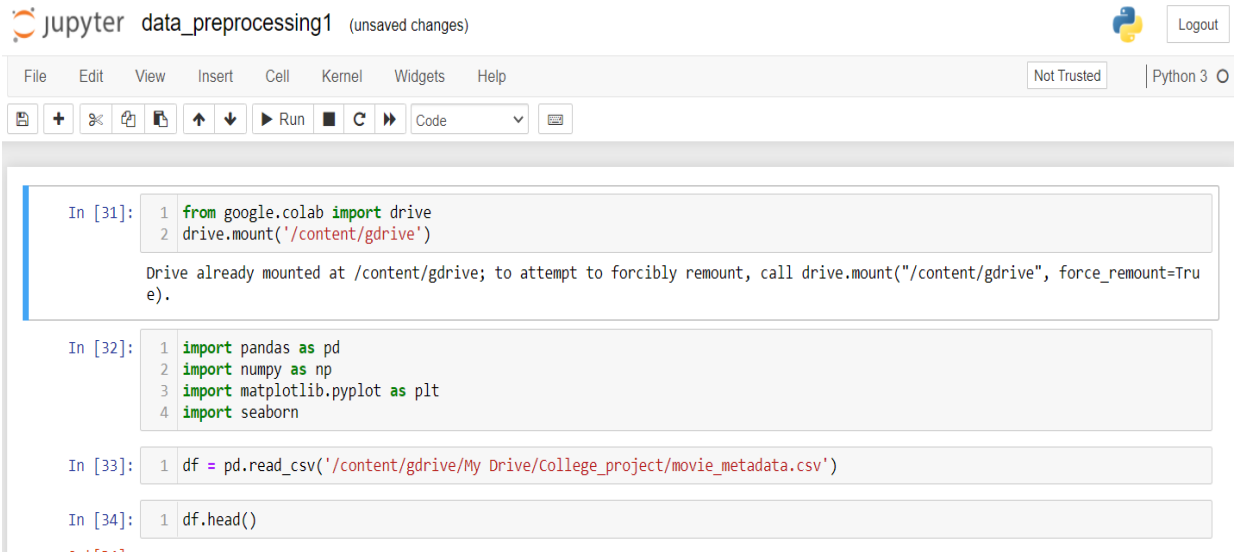
In [49]:
```
df.head()
```

Out[49]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy science friction	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon Levitt	Action Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [52]:
```
df.to_csv('data_p1.csv',index=False)
```

Machine Learning Models/movie_recommendation/preprocessing_part2.ipynb-



Out[34]:

	color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross
0	Color	James Cameron	723.0	178.0	0.0	855.0	Joel David Moore	1000.0	760505847.0
1	Color	Gore Verbinski	302.0	169.0	563.0	1000.0	Orlando Bloom	40000.0	309404152.0
2	Color	Sam Mendes	602.0	148.0	0.0	161.0	Rory Kinnear	11000.0	200074175.0
3	Color	Christopher Nolan	813.0	164.0	22000.0	23000.0	Christian Bale	27000.0	448130642.0
4	NaN	Doug Walker	NaN	NaN	131.0	NaN	Rob Walker	131.0	NaN

In [35]:
```
df.columns
```

Out[35]:
```
Index(['color', 'director_name', 'num_critic_for_reviews', 'duration',
       'director_facebook_likes', 'actor_3_facebook_likes', 'actor_2_name',
       'actor_1_facebook_likes', 'gross', 'genres', 'actor_1_name',
       'movie_title', 'num_voted_users', 'cast_total_facebook_likes',
       'actor_3_name', 'facenumber_in_poster', 'plot_keywords',
       'movie_imdb_link', 'num_user_for_reviews', 'language', 'country',
       'content_rating', 'budget', 'title_year', 'actor_2_facebook_likes',
       'imdb_score', 'aspect_ratio', 'movie_facebook_likes'],
      dtype='object')
```

For movie Recomendation we need only few columns like

1. Director_name
2. actor_1_name
3. actor_2_name
4. actor_3_name
5. genrres
6. movie_list

In [36]:
```
df = df.loc[:,['director_name','actor_1_name','actor_2_name','actor_3_name','genres','movie_title']]
```

In [37]:
```
df.head()
```

Out[37]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action|Adventure|Fantasy|Sci-Fi	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action|Adventure|Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action|Adventure|Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action|Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [38]:
```
df['director_name'].value_counts()
```

Out[38]:
```
Steven Spielberg    26
Woody Allen         22
Clint Eastwood      20
Martin Scorsese     20
Ridley Scott        17
                    ..
Ingmar Bergman       1
Edward Hall          1
Michael Winner       1
Sara Newens          1
Kelly Makin          1
Name: director_name, Length: 2398, dtype: int64
```

In [39]:
```
df['director_name'].isna().sum()
```

Out[39]:
```
104
```

In [40]:
```
df['actor_1_name'].isna().sum()
```

Out[40]:
```
7
```

In [41]:
```
df['actor_2_name'].isna().sum()
```

Out[41]: 13

In [42]:
```
df['actor_3_name'].isna().sum()
```

Out[42]: 23

In [43]:
```
df['genres'].isna().sum()
```

Out[43]: 0

In [44]:
```
df['movie_title'].isna().sum()
```

Out[44]: 0

Observation

We look that we have in the 'director_name' feature we have 104 Nan value

- In the 'actor_1_name' feature we have 7 Nan value
- In the 'actor_2_name' feature we have 13 Nan value
- In the 'actor_3_name' feature we have 23 Nan value
- In the 'genres' feature we have 0 Nan value
- In the 'movie_title' feature we have 0 Nan value

In [45]:
```
df.head()
```

Out[45]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action|Adventure|Fantasy|Sci-Fi	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action|Adventure|Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action|Adventure|Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action|Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [46]:
```
df['genres'] = df['genres'].str.replace('|',' ')
df['genres'] = df['genres'].str.replace('Sci-Fi' , 'science friction')
```

In [47]:
```
df.head()
```

Out[47]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy science friction	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon-Levitt	Action Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [48]:
```
df['actor_3_name'] = df['actor_3_name'].str.replace('-',' ')
```

In [49]:
```
df.head()
```

Out[49]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy science friction	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon Levitt	Action Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [52]:
```
df.to_csv('data_p1.csv',index=False)
```

Machine Learning Models/movie_recommendation/preprocessing_part3.ipynb-

jupyter preprocessing_part3 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

In [33]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=Tru
e).
```

In [34]:
```
import numpy as np
import pandas as pd
```

Loading dataset

In [35]:
```
df = pd.read_csv("/content/gdrive/MyDrive/college_project/2018_with_genres.csv")
```

In [36]:
```
df.head()
```

Out[36]:

	Unnamed: 0	Title	Production company	Cast and crew	genres
0	0	Insidious: The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	Mystery Horror Thriller
1	1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	Thriller Drama
2	2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer, Warren...	Action Thriller
3	3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter, St...	Drama History Western
4	4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	Action Thriller

In [37]:
```
df['Title'] = df['Title'].str.replace(':',' ')
```

In [38]:
```
df.head()
```

Out[38]:

	Unnamed: 0	Title	Production company	Cast and crew	genres
0	0	Insidious The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	Mystery Horror Thriller
1	1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	Thriller Drama
2	2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer, Warren...	Action Thriller
3	3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter, St...	Drama History Western
4	4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	Action Thriller

In [39]:
```
df['Cast and crew'][0]
```

Out[39]:
```
'Adam Robitel (director); Leigh Whannell (screenplay); Lin Shaye, Angus Sampson, Leigh Whannell, Spencer Locke, Caitlin Gerard,
Kirk Acevedo, Bruce Davison'
```

In [40]:
```
df['Cast and crew'] = df['Cast and crew'].str.replace(',' , ';')
```

In [41]:
```
def canvert(data):
  data = data.split(";")
  return data[0]
```

In [42]:
```
actual_data = df['Cast and crew']
df['director_name'] = actual_data.map(canvert)
```

In [43]:
```
df['director_name'] = df['director_name'].str.replace(r"(director)" , '')
df['director_name'] = df['director_name'].str.replace("(" , '')
df['director_name'] = df['director_name'].str.replace(")" , '')
```

In [44]:
```
df.head()
```

Out[44]:

	Unnamed: 0	Title	Production company	Cast and crew	genres	director_name
0	0	Insidious The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	Mystery Horror Thriller	Adam Robitel
1	1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	Thriller Drama	Lauren Wolkstein
2	2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer; Warren...	Action Thriller	Simon West
3	3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter; St...	Drama History Western	Warwick Thornton
4	4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	Action Thriller	Jaume Collet-Serra

In [45]:
```
def canvert(data):
  data = data.split(";")
  try:
    return data[2]
  except:
    return np.nan
```

In [46]:
```
actual_data = df['Cast and crew']
df['actor_1_name'] = actual_data.map(canvert)
```

In [47]:
```
df.head()
```

Out[47]:

	Unnamed: 0	Title	Production company	Cast and crew	genres	director_name	actor_1_name
0	0	Insidious The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	Mystery Horror Thriller	Adam Robitel	Lin Shaye
1	1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	Thriller Drama	Lauren Wolkstein	Alex Pettyfer
2	2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer; Warren...	Action Thriller	Simon West	Warren Davis II (screenplay)
3	3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter; St...	Drama History Western	Warwick Thornton	Steven McGregor (screenplay)
4	4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	Action Thriller	Jaume Collet-Serra	Philip de Blasi (screenplay)

In [48]:
```
df['actor_1_name'] = df['actor_1_name'].str.replace(r"(screenplay)" , '')
df['actor_1_name'] = df['actor_1_name'].str.replace("(" , '')
df['actor_1_name'] = df['actor_1_name'].str.replace(")" , '')
```

In [49]:
```
df.head()
```

Out[49]:

	Unnamed: 0	Title	Production company	Cast and crew	genres	director_name	actor_1_name
0	0	Insidious The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	Mystery Horror Thriller	Adam Robitel	Lin Shaye
1	1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	Thriller Drama	Lauren Wolkstein	Alex Pettyfer
2	2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer; Warren...	Action Thriller	Simon West	Warren Davis II
3	3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter; St...	Drama History Western	Warwick Thornton	Steven McGregor
4	4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	Action Thriller	Jaume Collet-Serra	Philip de Blasi

In [50]:
```
def canvert(data):
  data = data.split(";")
  try:
    return data[3]
  except:
    return np.nan
```

In [51]:
```
actual_data = df['Cast and crew']
df['actor_2_name'] = actual_data.map(canvert)
```

In [52]:
```
df.head()
```

Out[52]:

	Unnamed: 0	Title	Production company	Cast and crew	genres	director_name	actor_1_name	actor_2_name
0	0	Insidious The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	Mystery Horror Thriller	Adam Robitel	Lin Shaye	Angus Sampson
1	1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	Thriller Drama	Lauren Wolkstein	Alex Pettyfer	James Freedson-Jackson
2	2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer; Warren...	Action Thriller	Simon West	Warren Davis II	Dominic Cooper
3	3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter; St...	Drama History Western	Warwick Thornton	Steven McGregor	Bryan Brown
4	4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	Action Thriller	Jaume Collet-Serra	Philip de Blasi	Liam Neeson

In [53]:
```
def canvert(data):
  data = data.split(";")
  try:
    return data[4]
  except:
    return np.nan
```

In [54]:
```
actual_data = df['Cast and crew']
df['actor_3_name'] = actual_data.map(canvert)
```

In [55]:
```
df.head()
```

Out[55]:

	Unnamed: 0	Title	Production company	Cast and crew	genres	director_name	actor_1_name	actor_2_name	actor_3_name
0	0	Insidious The Last Key	Universal Pictures / Blumhouse Productions / S...	Adam Robitel (director); Leigh Whannell (scree...	Mystery Horror Thriller	Adam Robitel	Lin Shaye	Angus Sampson	Leigh Whannell
1	1	The Strange Ones	Vertical Entertainment	Lauren Wolkstein (director); Christopher Radcl...	Thriller Drama	Lauren Wolkstein	Alex Pettyfer	James Freedson-Jackson	Emily Althaus
2	2	Stratton	Momentum Pictures	Simon West (director); Duncan Falconer; Warren...	Action Thriller	Simon West	Warren Davis II	Dominic Cooper	Austin Stowell
3	3	Sweet Country	Samuel Goldwyn Films	Warwick Thornton (director); David Tranter; St...	Drama History Western	Warwick Thornton	Steven McGregor	Bryan Brown	Sam Neill
4	4	The Commuter	Lionsgate / StudioCanal / The Picture Company	Jaume Collet-Serra (director); Byron Willinger...	Action Thriller	Jaume Collet-Serra	Philip de Blasi	Liam Neeson	Vera Farmiga

In [56]:
```
df = df.loc[:,['director_name','actor_1_name','actor_2_name','actor_3_name','genres','Title']]
```

In [57]:
```
df.head()
```

Out[57]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	Title
0	Adam Robitel	Lin Shaye	Angus Sampson	Leigh Whannell	Mystery Horror Thriller	Insidious The Last Key
1	Lauren Wolkstein	Alex Pettyfer	James Freedson-Jackson	Emily Althaus	Thriller Drama	The Strange Ones
2	Simon West	Warren Davis II	Dominic Cooper	Austin Stowell	Action Thriller	Stratton
3	Warwick Thornton	Steven McGregor	Bryan Brown	Sam Neill	Drama History Western	Sweet Country
4	Jaume Collet-Serra	Philip de Blasi	Liam Neeson	Vera Farmiga	Action Thriller	The Commuter

In [58]:
```
df.isna().sum()
```

Out[58]:
```
director_name     0
actor_1_name      8
actor_2_name     12
actor_3_name     30
genres            3
Title             0
dtype: int64
```

In [59]:
```
df = df.dropna(how='any')
```

In [60]:
```
df.isna().sum()
```

Out[60]:
```
director_name    0
actor_1_name     0
actor_2_name     0
actor_3_name     0
genres           0
Title            0
dtype: int64
```

In [61]:
```
df.head()
```

Out[61]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	Title
0	Adam Robitel	Lin Shaye	Angus Sampson	Leigh Whannell	Mystery Horror Thriller	Insidious The Last Key
1	Lauren Wolkstein	Alex Pettyfer	James Freedson-Jackson	Emily Althaus	Thriller Drama	The Strange Ones
2	Simon West	Warren Davis II	Dominic Cooper	Austin Stowell	Action Thriller	Stratton
3	Warwick Thornton	Steven McGregor	Bryan Brown	Sam Neill	Drama History Western	Sweet Country
4	Jaume Collet-Serra	Philip de Blasi	Liam Neeson	Vera Farmiga	Action Thriller	The Commuter

In [62]:
```
df.shape
```

Out[62]:
```
(235, 6)
```

In [63]:
```
df.to_csv('new_data_part3.csv',index=False)
```

Machine Learning Models/movie_recommendation/ml_project_updated.ipynb-

jupyter ml_project_updated (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Code

In [38]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

Importing library

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

Loading Datasets from gdrive

In [2]:
```
# df = pd.read_csv("/content/gdrive/MyDrive/college_project/data_p1.csv")
df = pd.read_csv("data_p1.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy science friction	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon Levitt	Action Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   director_name  4939 non-null   object
 1   actor_1_name   5036 non-null   object
 2   actor_2_name   5030 non-null   object
 3   actor_3_name   5020 non-null   object
 4   genres         5043 non-null   object
 5   movie_title    5043 non-null   object
dtypes: object(6)
memory usage: 236.5+ KB
```

Checking NAN value

In [5]:
```
df.isna().any()
```

Out[5]:
```
director_name     True
actor_1_name      True
actor_2_name      True
actor_3_name      True
genres           False
movie_title      False
dtype: bool
```

Drop NAN value

In [6]:
```
df = df.dropna()
```

In [7]:
```
df.isna().any()
```

Out[7]:
```
director_name    False
actor_1_name     False
actor_2_name     False
actor_3_name     False
genres           False
movie_title      False
dtype: bool
```

In [8]:
```
df.shape
```

Out[8]:
```
(4919, 6)
```

In [9]:
```
df.head()
```

Out[9]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy science friction	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon Levitt	Action Thriller	The Dark Knight Rises
5	Andrew Stanton	Daryl Sabara	Samantha Morton	Polly Walker	Action Adventure science friction	John Carter

In [10]:
```
df['director_name'] = df['director_name'].str.lower()
df['actor_1_name'] = df['actor_1_name'].str.lower()
df['actor_2_name'] = df['actor_2_name'].str.lower()
df['actor_3_name'] = df['actor_3_name'].str.lower()
df['genres'] = df['genres'].str.lower()
df['movie_title'] = df['movie_title'].str.lower()
```

In [11]:
```
df.head()
```

Out[11]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter

In [12]:
```
index = []
for i in range(0 , df.shape[0]):
  index.append(i)
```

In [13]:
```
df["index"] = index
```

In [14]:
```
df.head()
```

Out[14]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	index
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar	0
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end	1
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre	2
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises	3
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter	4

In [15]:
```
df['director_name'] = df['director_name'].str.replace('\d+', '')
df['actor_1_name'] = df['actor_1_name'].str.replace('\d+', '')
df['actor_2_name'] = df['actor_2_name'].str.replace('\d+', '')
df['actor_3_name'] = df['actor_3_name'].str.replace('\d+', '')
df['genres'] = df['genres'].str.replace('\d+', '')
df['movie_title'] = df['movie_title'].str.replace('\d+', '')
```

In [16]:
```
def combined_features(row):
    return row['director_name']+" "+row['actor_1_name']+" "+row['actor_2_name']+" "+row['actor_3_name']+" "+row["genres"]+"
df["combined_features"] = df.apply(combined_features, axis =1)
```

In [17]:
```
df.head()
```

Out[17]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	index	combined_features
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar	0	james cameron cch pounder joel david moore wes...
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end	1	gore verbinski johnny depp orlando bloom jack ...
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre	2	sam mendes christoph waltz rory kinnear stepha...
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises	3	christopher nolan tom hardy christian bale jos...
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter	4	andrew stanton daryl sabara samantha morton po...

Apply countVectorizer for canverting text into vector

In [18]:
```
cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])
print("Count Matrix:", count_matrix.toarray())
```

```
Count Matrix: [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Find cosine Similarity

In [19]:
```
cosine_sim = cosine_similarity(count_matrix)
```

In [20]:
```
df.head()
```

Out[20]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	index	combined_features
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar	0	james cameron cch pounder joel david moore wes...
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end	1	gore verbinski johnny depp orlando bloom jack ...
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre	2	sam mendes christoph waltz rory kinnear stepha...
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises	3	christopher nolan tom hardy christian bale jos...
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter	4	andrew stanton daryl sabara samantha morton po...

In [21]:
```
df['movie_title'] = df['movie_title'].str.strip()
df.head()
```

Out[21]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	index	combined_features
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar	0	james cameron cch pounder joel david moore wes...
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end	1	gore verbinski johnny depp orlando bloom jack ...
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre	2	sam mendes christoph waltz rory kinnear stepha...
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises	3	christopher nolan tom hardy christian bale jos...
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter	4	andrew stanton daryl sabara samantha morton po...

Finding the index of the movie

In [32]:
```
movie_user_likes = "aliens"

def get_index_from_title(title):
    return df[df['movie_title'] == title]["index"].values[0]

movie_index = get_index_from_title(movie_user_likes)
```

In [33]:
```
movie_index
```

Out[33]: 2435

In [34]:
```
similar_movies = list(enumerate(cosine_sim[movie_index]))
```

In [35]:
```
sorted_similar_movies = sorted(similar_movies, key=lambda x:x[1], reverse=True)
```

Similar Movies

In [36]:
```
def get_title_from_index(index):
    return df[df.index == index]["movie_title"].values[0]
i=0
for movie in sorted_similar_movies:
    print(get_title_from_index(movie[0]))
    i=i+1
    if i>15:
        break
```

```
steve jobs
the last temptation of christ
minority report
avp: alien vs. predator
avatar
lucky number slevin
pearl harbor
rush hour
coco before chanel
the amazing spider-man
the chronicles of narnia: the lion, the witch and the wardrobe
ghostbusters
mars attacks!
universal soldier: the return
nancy drew
the last station
```

In [37]:
```
df.to_csv('data.csv')
```

In [38]:
```
df[df['movie_title'] == 'aliens']["index"].values[0]
```

Out[38]: 2435

In [27]:
```
import pickle
```

In [30]:
```
Pkl_Filename = "recommendation.pkl"
```

In [ ]:
```
with open(Pkl_Filename, 'wb') as file:
    pickle.dump(cosine_sim, file)
```

Machine Learning Models/sentiment analysis/Sentiment_analysis.ipynb-

jupyter Sentiment_analysis (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

Run Code

In [1]:
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [2]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

In [3]:
```
df = pd.read_csv("/content/gdrive/MyDrive/college_project/reviews.csv")
```

In [4]:
```
df.head()
```

In [5]:
```
df.shape
```

Out[5]:
```
(6918, 2)
```

In [6]:
```
df.isna().any()
```

Out[6]:
```
comments    False
reviews     False
dtype: bool
```

In [7]:
```
df['comments_1'] = df['comments'].str.lower()
```

In [8]:
```
df.head()
```

Out[8]:

	comments	reviews	comments_1
0	The Da Vinci Code book is just awesome.	1	the da vinci code book is just awesome.
1	this was the first clive cussler i've ever rea...	1	this was the first clive cussler i've ever rea...
2	i liked the Da Vinci Code a lot.	1	i liked the da vinci code a lot.
3	i liked the Da Vinci Code a lot.	1	i liked the da vinci code a lot.
4	I liked the Da Vinci Code but it ultimatly did...	1	i liked the da vinci code but it ultimatly did...

In [9]:
```
df['comments_1'] = df['comments_1'].str.replace('\d+', '')
```

In [10]:
```
df.head()
```

Out[10]:

	comments	reviews	comments_1
0	The Da Vinci Code book is just awesome.	1	the da vinci code book is just awesome.
1	this was the first clive cussler i've ever rea...	1	this was the first clive cussler i've ever rea...
2	i liked the Da Vinci Code a lot.	1	i liked the da vinci code a lot.
3	i liked the Da Vinci Code a lot.	1	i liked the da vinci code a lot.
4	I liked the Da Vinci Code but it ultimatly did...	1	i liked the da vinci code but it ultimatly did...

In [11]:
```
df['comments_1'] = df['comments_1'].str.strip()
df.head()
```

Out[11]:

	comments	reviews	comments_1
0	The Da Vinci Code book is just awesome.	1	the da vinci code book is just awesome.
1	this was the first clive cussler i've ever rea...	1	this was the first clive cussler i've ever rea...
2	i liked the Da Vinci Code a lot.	1	i liked the da vinci code a lot.
3	i liked the Da Vinci Code a lot.	1	i liked the da vinci code a lot.
4	I liked the Da Vinci Code but it ultimatly did...	1	i liked the da vinci code but it ultimatly did...

In [12]:
```
df.head()
```

Out[12]:

	comments	reviews	comments_1
0	The Da Vinci Code book is just awesome.	1	the da vinci code book is just awesome.
1	this was the first clive cussler i've ever rea...	1	this was the first clive cussler i've ever rea...
2	i liked the Da Vinci Code a lot.	1	i liked the da vinci code a lot.
3	i liked the Da Vinci Code a lot.	1	i liked the da vinci code a lot.
4	I liked the Da Vinci Code but it ultimatly did...	1	i liked the da vinci code but it ultimatly did...

In [13]:
```
X = df['comments_1']
y = df['reviews']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [15]:
```
print("Shape of X-train : ",X_train.shape[0])
print("Shape of X-test : ",X_test.shape[0])
```

```
Shape of X-train :  4635
Shape of X-test :  2283
```

In [16]:
```
count_vect = CountVectorizer() #in scikit-learn
count_vect.fit(X_train)
print("some feature names ", count_vect.get_feature_names()[:10])
print('='*50)

final_counts = count_vect.transform(X_train)
test_data = count_vect.transform(X_test)

print("the type of count vectorizer ",type(final_counts))
print("the shape of out text BOW vectorizer ",final_counts.get_shape())
print("the number of unique words ", final_counts.get_shape()[1])
```

```
some feature names  ['able', 'abortion', 'about', 'absolute', 'absolutely', 'absurd', 'academy', 'acceptable', 'accompaniment',
'according']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (4635, 1685)
the number of unique words  1685
```

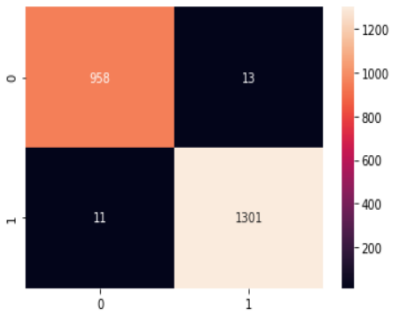
In [17]:
```
clf = LogisticRegression(random_state=0)
clf.fit(final_counts, y_train)
predict = clf.predict(test_data)
accuracy_score(y_test, predict)
```

Out[17]: 0.9894875164257556

In [18]:
```
result = confusion_matrix(y_test, predict)
sns.heatmap(result, annot=True, fmt="d")

```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7f0cac09c150>



In [33]:
```
import pickle
```

In [36]:
```
pickle.dump(count_vect, open("/content/gdrive/MyDrive/college_project/count_vectorizer", 'wb'))
```

In [37]:
```
pickle.dump(clf, open("/content/gdrive/MyDrive/college_project/model_clf", 'wb'))
```

Loading Model

In [38]:
```
# load the model from disk
Count_vectorizer = pickle.load(open("/content/gdrive/MyDrive/college_project/count_vectorizer", 'rb'))
model_clf = pickle.load(open("/content/gdrive/MyDrive/college_project/model_clf" , 'rb'))
```

In [42]:
```
reviews = input("Enter a review : ")
```

```
Enter a review : This is very bad movies
```

In [43]:
```
test = pd.Series(reviews)
type(test)
```

Out[43]: pandas.core.series.Series

In [44]:
```
vector = Count_vectorizer.transform(test)
model_clf.predict(vector)
```

Out[44]: array([0])

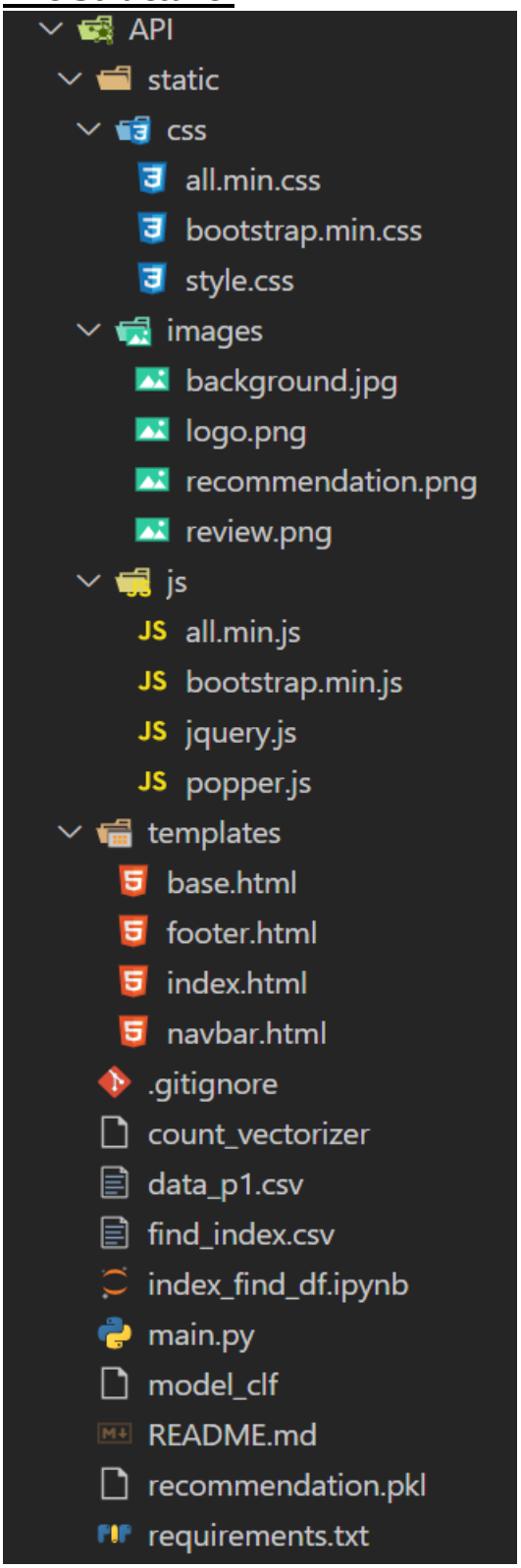
Machine Learning Models/README.md

Project Red Mind ML Models

We have used the Countvectorizer to convert text into a vector, Cosine Similarity for the recommendation and for review sentiment analysis we have used the Logistic Regression algorithm.

8.3 API CODE

File Structure-



API/static/css/style.css-

```
.trans{
  background-color: rgba(225, 0, 0, 0.3);
}

.bg {
  background-image: url("../images/background.jpg");
  height: 100%;
  background-color: rgb(139, 33, 33);
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}
```

API/templates/base.html-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="{{url_for('static',filename='css/bootstrap.min.css')}}" rel="stylesheet">
    <link href="{{url_for('static',filename='css/all.min.css')}}" rel="stylesheet">
    <link href="{{url_for('static',filename='css/style.css')}}" rel="stylesheet">
    <link rel="shortcut icon" href="{{url_for('static',filename='images/logo.png')}}" type="image/
x-icon">
    <title>Project Red Mind : API</title>
</head>
<body>
    {% include 'navbar.html' %}
    <div class="container-fluid bg">
        {% block content %}
        {% endblock content %}
    </div>
    {% include 'footer.html' %}
    <script src="{{url_for('static',filename='js/bootstrap.min.js')}}"></script>
    <script src="{{url_for('static',filename='js/popper.js')}}"></script>
    <script src="{{url_for('static',filename='js/all.min.js')}}"></script>
</body>
</html>
```

API/templates/footer.html-

```
<footer class="container-fluid p-0">
   <div class="bg-dark text-center p-1 text-white">
      <div>
         <a href="https://youtu.be/3-Ko3rMPN38" target="_blank">
            <i class="fab fa-youtube fa-2x m-1 text-white"></i>
         </a>
         <a href="https://github.com/IntegratedMindHeart" target="_blank">
            <i class="fab fa-github fa-2x m-1 text-white"></i>
         </a>
         <a>
            <i class="fab fa-linkedin fa-2x m-1 text-white"></i>
         </a>
      </div>
      <p>Copyright &#169; 2020-21 Team Integrated Mind Heart</p>
   </div>
</footer>
```

API/templates/index.html-

```
{% extends "base.html" %}
{% block content %}
<div class="container pt-4">
   <div class="row text-center py-4">
      <div class="alert alert-danger h2" role="alert">
         How to use movie recommendation API
      </div>
   </div>
   <div class="row pb-5">
      <div class="col-md-4 col-sm-12 text-center my-2">
         <img width="320px" class="border rounded shadow-
lg" src="{{url_for('static',filename='images/recommendation.png')}}" alt="recommendation">
      </div>
      <div class="col-md-8 col-sm-12 my-2 trans p-3 border rounded shadow-lg text-light text-
break">
         <h1>Send GET request -</h1>
         <h4>http://127.0.0.1:5000/api/recommendation/{<i>movie_name</i>}</h4>
         <h5>where, movie_name your movie name</h5>
         <br>
         <h1>Example -</h1>
         <a href="http://127.0.0.1:5000/api/recommendation/avatar" target="_blank" class="h4">h
ttp://127.0.0.1:5000/api/recommendation/avatar</a>
      </div>
   </div>
</div>
<div class="container">
   <div class="row text-center py-4">
      <div class="alert alert-danger h2" role="alert">
```

```
            How to use movie review sentiment analysis API
        </div>
    </div>
    <div class="row pb-5">
        <div class="col-md-4 col-sm-12 text-center my-2">
            <img width="320px" class="border rounded shadow-
lg" src="{{url_for('static',filename='images/review.png')}}" alt="recommendation">
        </div>
        <div class="col-md-8 col-sm-12 my-2 trans p-3 border rounded shadow-lg text-light text-
break">
            <h1>Send GET request -</h1>
            <h4>http://127.0.0.1:5000/api/review/{<i>movie_review</i>}</h4>
            <h5>where, movie_review your movie review</h5>
            <br>
            <h1>Example -</h1>
            <a href="http://127.0.0.1:5000/api/review/good" target="_blank" class="h4">http://127.0.
0.1:5000/api/review/good</a>
        </div>
    </div>
</div>
{% endblock content %}
```

API/templates/navbar.html-

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
        <a class="navbar-brand" href="http://127.0.0.1:8000/"><i class="fas fa-
film"></i> Project Red Mind</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent"
            aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                    <a class="nav-link" href="http://127.0.0.1:8000/">Home <span class="sr-
only">(current)</span></a>
                </li>
                <li class="nav-item">
                    <a class="nav-link active disabled">API <span class="sr-
only">(current)</span></a>
                </li>
                <li class="nav-item">
                    <a class="nav-
link" href="http://127.0.0.1:8000/admin" target="_blank">Admin Panel <span class="sr-
only">(current)</span></a>
```

```
            </li>
            <li class="nav-item">
               <a class="nav-link" href="http://127.0.0.1:8000/about">About <span class="sr-
only">(current)</span></a>
            </li>
         </ul>
      </div>
   </div>
</nav>
```

API/.gitignore-

```
recommendation.pkl
```

API/main.py-

```
from flask import Flask,render_template,jsonify
import pandas as pd
import pickle

from flask import Flask
app = Flask(__name__)

@app.route('/')
def api_help():
   return render_template('index.html')

@app.route('/api/recommendation/<string:movie>')
def get_recommendation(movie):
   df=pd.read_csv('find_index.csv')
   index=df[df['movie_title'] == movie]["index"].values[0]

   with open('recommendation.pkl', 'rb') as file:
      LOAD_MODEL = pickle.load(file)

   similar_movies = list(enumerate(LOAD_MODEL[index]))

   sorted_similar_movies = sorted(similar_movies, key=lambda x:x[1], reverse=True)

   def get_title_from_index(index):
      return df[df.index == index]["movie_title"].values[0]

   l=[]

   i=0
   for movie in sorted_similar_movies:
      m=get_title_from_index(movie[0])
      if type(m) is float:
```

```
            continue
        l.append(m)
        i=i+1
        if i>15:
            break

    return jsonify({'similar_movies':l[1:]})


@app.route('/api/review/<string:review>')
def get_review(review):
    count_vectorizer=pickle.load(open("count_vectorizer",'rb'))
    model_clf=pickle.load(open("model_clf" ,'rb'))
    test=pd.Series(review)
    vector=count_vectorizer.transform(test)
    result=int(model_clf.predict(vector)[0])
    return jsonify({'result':result})

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

API/requirements.txt-

```
Flask==2.0.1
pandas==1.2.4
scikit-learn==0.24.2
gunicorn==20.1.0
```

API/README.md-

```
# Project Red Mind API
We have created this API using a flask that helps our users to create movie recommendation syste
ms and sentiment analysis. This API can be used by any developer who wants to create a movie r
ecommendation system for their application he/she will just need to call our API.

## See Project

visit- https://porjectredmindapi.herokuapp.com/

## Installation

Use the package manager [pip](https://pip.pypa.io/en/stable/) to install all python packages on yo
ur system which is listed in **[requirements.txt](https://github.com/IntegratedMindHeart/api/bl
ob/master/requirements.txt)** file.

## How to run

1. Open cmd or any other terminal

2. Go to file location were **[main.py](https://github.com/IntegratedMindHeart/api/blob/master/
main.py)** is located

3. Run the command *python main.py*

4. Open your browser and open url http://127.0.0.1:5000/

#### Demo Video - https://youtu.be/3-Ko3rMPN38
```

API/index_find_df.ipynb-

Out[5]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy science friction	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon Levitt	Action Thriller	The Dark Knight Rises
4	Doug Walker	Doug Walker	Rob Walker	NaN	Documentary	Star Wars: Episode VII - The Force Awakens ...

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5043 entries, 0 to 5042
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   director_name  4939 non-null   object
 1   actor_1_name   5036 non-null   object
 2   actor_2_name   5030 non-null   object
 3   actor_3_name   5020 non-null   object
 4   genres         5043 non-null   object
 5   movie_title    5043 non-null   object
dtypes: object(6)
memory usage: 236.5+ KB
```

Checking NAN value

In [7]:
```
df.isna().any()
```

Out[7]:
```
director_name     True
actor_1_name      True
actor_2_name      True
actor_3_name      True
genres           False
movie_title      False
dtype: bool
```

Drop NAN value

In [8]:
```
df = df.dropna()
```

In [9]:
```
df.isna().any()
```

Out[9]:
```
director_name    False
actor_1_name     False
actor_2_name     False
actor_3_name     False
genres           False
movie_title      False
dtype: bool
```

In [10]:
```
df.shape
```

Out[10]:
```
(4919, 6)
```

In [11]:
```
df.head()
```

Out[11]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	James Cameron	CCH Pounder	Joel David Moore	Wes Studi	Action Adventure Fantasy science friction	Avatar
1	Gore Verbinski	Johnny Depp	Orlando Bloom	Jack Davenport	Action Adventure Fantasy	Pirates of the Caribbean: At World's End
2	Sam Mendes	Christoph Waltz	Rory Kinnear	Stephanie Sigman	Action Adventure Thriller	Spectre
3	Christopher Nolan	Tom Hardy	Christian Bale	Joseph Gordon Levitt	Action Thriller	The Dark Knight Rises
5	Andrew Stanton	Daryl Sabara	Samantha Morton	Polly Walker	Action Adventure science friction	John Carter

In [12]:
```
df['director_name'] = df['director_name'].str.lower()
df['actor_1_name'] = df['actor_1_name'].str.lower()
df['actor_2_name'] = df['actor_2_name'].str.lower()
df['actor_3_name'] = df['actor_3_name'].str.lower()
df['genres'] = df['genres'].str.lower()
df['movie_title'] = df['movie_title'].str.lower()
```

In [13]:
```
df.head()
```

Out[13]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter

In [14]:
```
index = []
for i in range(0 , df.shape[0]):
  index.append(i)
```

In [15]:
```
df["index"] = index
```

In [16]:
```
df.head()
```

Out[16]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	index
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar	0
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end	1
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre	2
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises	3
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter	4

In [17]:
```
df['director_name'] = df['director_name'].str.replace('\d+', '')
df['actor_1_name'] = df['actor_1_name'].str.replace('\d+', '')
df['actor_2_name'] = df['actor_2_name'].str.replace('\d+', '')
df['actor_3_name'] = df['actor_3_name'].str.replace('\d+', '')
df['genres'] = df['genres'].str.replace('\d+', '')
df['movie_title'] = df['movie_title'].str.replace('\d+', '')
```

```
<ipython-input-17-5876ff70f3b1>:1: FutureWarning: The default value of regex will change from True to False in a future versio
n.
  df['director_name'] = df['director_name'].str.replace('\d+', '')
<ipython-input-17-5876ff70f3b1>:2: FutureWarning: The default value of regex will change from True to False in a future versio
n.
  df['actor_1_name'] = df['actor_1_name'].str.replace('\d+', '')
<ipython-input-17-5876ff70f3b1>:3: FutureWarning: The default value of regex will change from True to False in a future versio
n.
  df['actor_2_name'] = df['actor_2_name'].str.replace('\d+', '')
<ipython-input-17-5876ff70f3b1>:4: FutureWarning: The default value of regex will change from True to False in a future versio
n.
  df['actor_3_name'] = df['actor_3_name'].str.replace('\d+', '')
<ipython-input-17-5876ff70f3b1>:5: FutureWarning: The default value of regex will change from True to False in a future versio
n.
  df['genres'] = df['genres'].str.replace('\d+', '')
<ipython-input-17-5876ff70f3b1>:6: FutureWarning: The default value of regex will change from True to False in a future versio
n.
  df['movie_title'] = df['movie_title'].str.replace('\d+', '')
```

In [18]:
```
def combined_features(row):
    return row['director_name']+" "+row['actor_1_name']+" "+row['actor_2_name']+" "+row['actor_3_name']+" "+row["genres"]+" "+row
df["combined_features"] = df.apply(combined_features, axis =1)
```

In [19]:
```
df.head()
```

Out[19]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	index	combined_features
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar	0	james cameron cch pounder joel david moore wes...
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end	1	gore verbinski johnny depp orlando bloom jack ...
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre	2	sam mendes christoph waltz rory kinnear stepha...
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises	3	christopher nolan tom hardy christian bale jos...
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter	4	andrew stanton daryl sabara samantha morton po...

In [20]:
```
df['movie_title'] = df['movie_title'].str.strip()
df.head()
```

Out[20]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	index	combined_features
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar	0	james cameron cch pounder joel david moore wes...
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end	1	gore verbinski johnny depp orlando bloom jack ...
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre	2	sam mendes christoph waltz rory kinnear stepha...
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises	3	christopher nolan tom hardy christian bale jos...
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter	4	andrew stanton daryl sabara samantha morton po...

In [21]:
```
df
```

Out[21]:

	director_name	actor_1_name	actor_2_name	actor_3_name	genres	movie_title	index	combined_features
0	james cameron	cch pounder	joel david moore	wes studi	action adventure fantasy science friction	avatar	0	james cameron cch pounder joel david moore wes...
1	gore verbinski	johnny depp	orlando bloom	jack davenport	action adventure fantasy	pirates of the caribbean: at world's end	1	gore verbinski johnny depp orlando bloom jack ...
2	sam mendes	christoph waltz	rory kinnear	stephanie sigman	action adventure thriller	spectre	2	sam mendes christoph waltz rory kinnear stepha...
3	christopher nolan	tom hardy	christian bale	joseph gordon levitt	action thriller	the dark knight rises	3	christopher nolan tom hardy christian bale jos...
5	andrew stanton	daryl sabara	samantha morton	polly walker	action adventure science friction	john carter	4	andrew stanton daryl sabara samantha morton po...
...	...	...	...	...	...	...	...	...

...	...	...	...	...	...	...	...	...
5037	edward burns	kerry bishé	caitlin fitzgerald	daniella pineda	comedy drama	newlyweds	4914	edward burns kerry bishé caitlin fitzgerald da...
5038	scott smith	eric mabius	daphne zuniga	crystal lowe	comedy drama	signed sealed delivered	4915	scott smith eric mabius daphne zuniga crystal ...
5040	benjamin roberds	eva boehnke	maxwell moody	david chandler	drama horror thriller	a plague so pleasant	4916	benjamin roberds eva boehnke maxwell moody dav...
5041	daniel hsia	alan ruck	daniel henney	eliza coupe	comedy drama romance	shanghai calling	4917	daniel hsia alan ruck daniel henney eliza coup...
5042	jon gunn	john august	brian herzlinger	jon gunn	documentary	my date with drew	4918	jon gunn john august brian herzlinger jon gunn...

4919 rows × 8 columns

In [22]:
```
new_df=pd.DataFrame()
```

In [23]:
```
new_df['index']=df['index']
```

In [24]:
```
new_df['movie_title']=df['movie_title']
```

In [25]:
```
new_df
```

Out[25]:

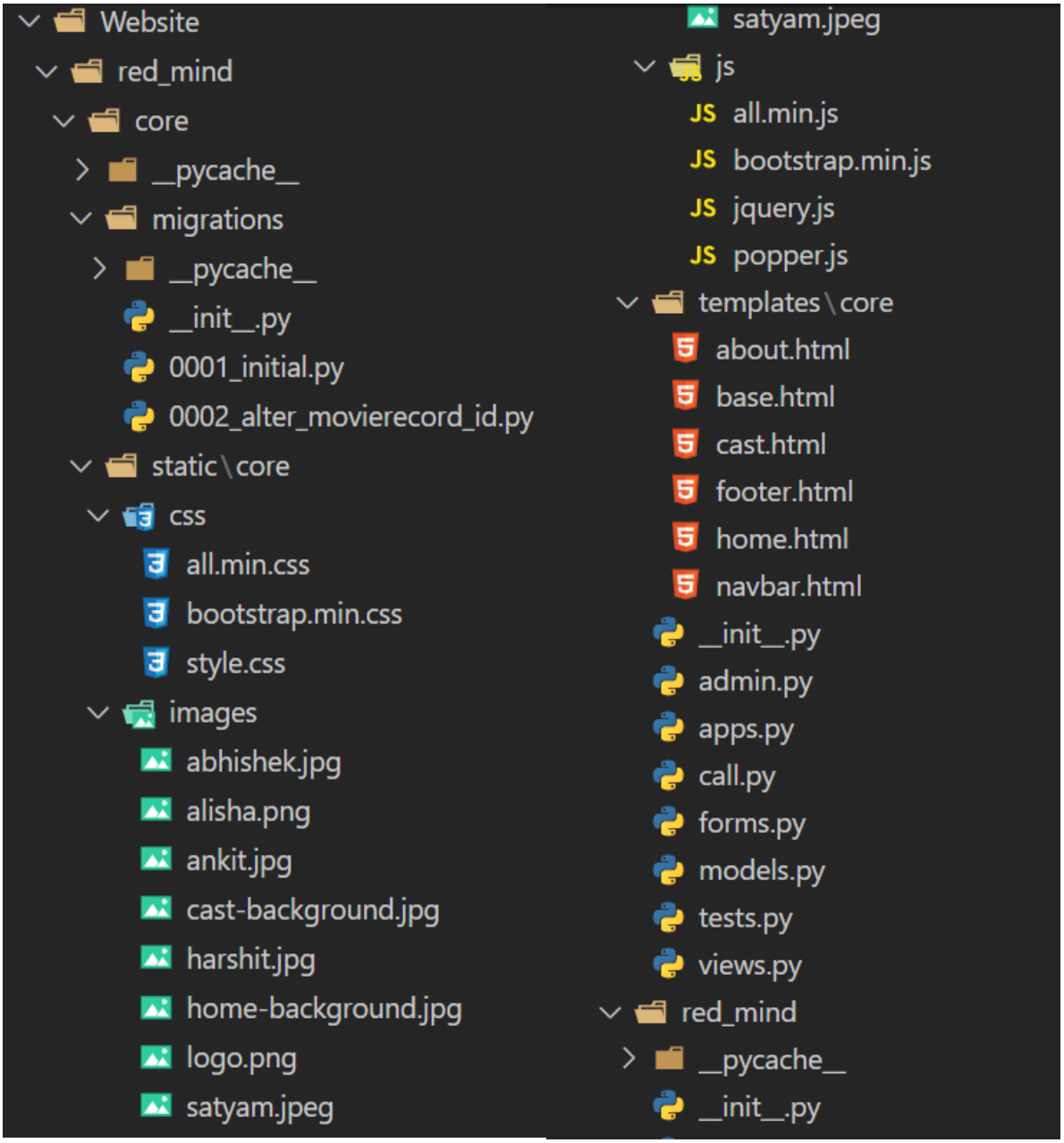
	index	movie_title
0	0	avatar
1	1	pirates of the caribbean: at world's end
2	2	spectre
3	3	the dark knight rises
5	4	john carter
...	...	...
5037	4914	newlyweds
5038	4915	signed sealed delivered
5040	4916	a plague so pleasant
5041	4917	shanghai calling
5042	4918	my date with drew

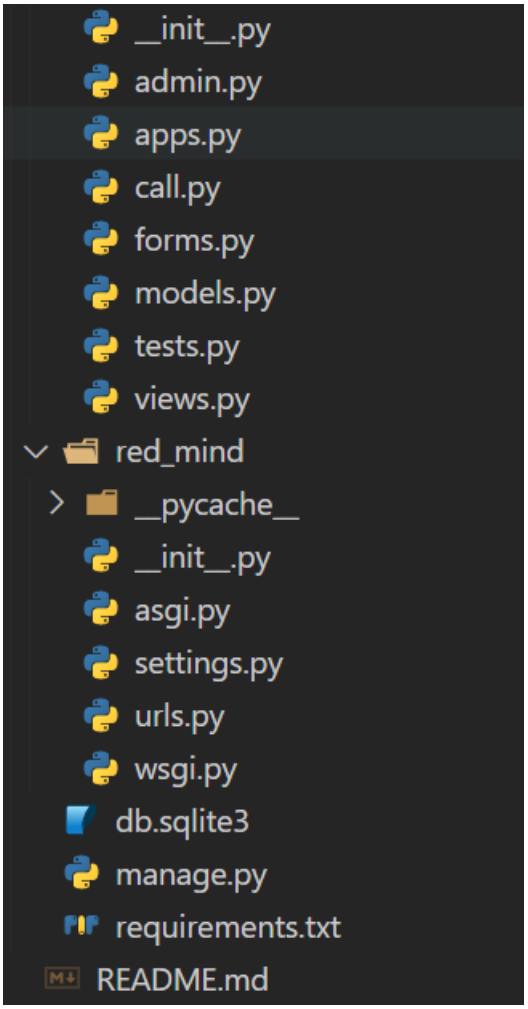
4919 rows × 2 columns

In [26]:
```
new_df.to_csv('find_index.csv')
```

8.4 WEBSITE CODE

File Structure-





Website/red_mind/core/migrations/0001_initial.py-

```
# Generated by Django 3.2.4 on 2021-06-30 05:56

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='MovieRecord',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose
_name='ID')),
```

```
                ('movie_name', models.CharField(max_length=200)),
                ('date_time', models.DateTimeField(auto_now_add=True)),
            ],
        ),
    ]
```

Website/red_mind/core/migrations/0002_alter_movierecord_id.py-

```
# Generated by Django 3.2.4 on 2021-06-30 05:59

from django.db import migrations, models


class Migration(migrations.Migration):

    dependencies = [
        ('core', '0001_initial'),
    ]

    operations = [
        migrations.AlterField(
            model_name='movierecord',
            name='id',
            field=models.AutoField(primary_key=True, serialize=False),
        ),
    ]
```

Website/red_mind/core/static/core/style.css-

```
.text-responsive {
  font-size: calc(100% + 1vw + 1vh);
  font-family: "Rubik", sans-serif;
}

.trans{
  background-color: rgba(225, 0, 0, 0.3);
}

.trans2{
  background-color: #1d1d1dc9;
}

.bg-home {
  background-image: url("../images/home-background.jpg");
  height: 100%;
  background-color: rgb(139, 33, 33);
```

```
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}

.bg-cast {
  background-image: url("../images/cast-background.jpg");
  height: 100%;
  background-color: rgb(17, 10, 10);
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
}

.card, table tr td, table tr th{
  background-color: rgba(0, 0, 0, 0) !important;
}

.astext {
  background:none;
  border:none;
  margin:0;
  padding:0;
  cursor: pointer;
  color: blue;
}

.fstyle {
  font-family: 'Noto Sans KR', sans-serif;
}
```

Website/red_mind/core/templates/core/about.html-

```
{% extends 'core/base.html' %}
{% load static %}
{% block title %}About{% endblock title %}
{% block bg %}bg-cast{% endblock bg %}
{% block content %}
<div class="container">
    <div class="row text-center pt-5">
        <div class="alert alert-secondary h2" role="alert">
            About the Team Integrated Mind Heart
        </div>
    </div>
    <div class="row row-cols-1 row-cols-md-3 g-4 pb-5 mt-2 justify-content-center">
        <div class="col">
            <div class="card h-100 border text-white text-center">
```

```
            <img src="{% static 'core/images/satyam.jpeg' %}" class="card-img-
top" alt="satyam_seth">
            <div class="card-body trans2">
              <h3 class="card-title">Satyam Seth</h3>
              <h4>Data Collection, Creating API, Backend, Deployment</h4>
              <p class="card-text">
                <a href="mailto:satyam1998.1998@gmail.com" target="_blank">
                  <i class="fas fa-envelope fa-2x mr-2 text-light"></i>
                </a>
                <a href="https://www.linkedin.com/in/satyam-seth-39069b17a" target="_blank">
                  <i class="fab fa-linkedin fa-2x mx-2 text-light"></i>
                </a>
                <a href="https://github.com/satyam-seth" target="_blank">
                  <i class="fab fa-github fa-2x ml-2 text-light"></i>
                </a>
              </p>
            </div>
          </div>
        </div>
        <div class="col">
          <div class="card h-100 border text-white text-center">
            <img src="{% static 'core/images/ankit.jpg' %}" class="card-img-
top" alt="ankit_kumar_gupta">
            <div class="card-body trans2">
              <h3 class="card-title">Ankit Kumar Gupta</h3>
              <h4>Creating Recommendation and Sentiment Snalysis ML models</h4>
              <p class="card-text">
                <a href="mailto:ankitgupta6564@gmail.com" target="_blank">
                  <i class="fas fa-envelope fa-2x mr-2 text-light"></i>
                </a>
                <a href="https://www.linkedin.com/in/ankit-gupta-02376b179" target="_blank">
                  <i class="fab fa-linkedin fa-2x mx-2 text-light"></i>
                </a>
                <a href="https://github.com/ankit700771" target="_blank">
                  <i class="fab fa-github fa-2x ml-2 text-light"></i>
                </a>
              </p>
            </div>
          </div>
        </div>
        <div class="col">
          <div class="card h-100 border text-white text-center">
            <img src="{% static 'core/images/alisha.png' %}" class="card-img-
top" alt="alisha_parveen">
            <div class="card-body trans2">
              <h3 class="card-title">Alisha Parveen</h3>
              <h4>Frontend and Project Documentation</h4>
              <p class="card-text">
```

```
                    <a href="mailto:alishakhan1999@gmail.com" target="_blank">
                      <i class="fas fa-envelope fa-2x mr-2 text-light"></i>
                    </a>
                    <a href="https://www.linkedin.com/in/alisha-parveen-" target="_blank">
                      <i class="fab fa-linkedin fa-2x mx-2 text-light"></i>
                    </a>
                    <a href="https://github.com/shali-lalee" target="_blank">
                      <i class="fab fa-github fa-2x ml-2 text-light"></i>
                    </a>
                  </p>
                </div>
              </div>
            </div>
            <div class="col">
              <div class="card h-100 border text-white text-center">
                <img src="{% static 'core/images/harshit.jpg' %}" class="card-img-
top" alt="harshit_dixit">
                <div class="card-body trans2">
                  <h3 class="card-title">Harshit Dixit</h3>
                  <h4>Frontend</h4>
                  <p class="card-text">
                    <a href="mailto:harshitdixit65@gmail.com" target="_blank">
                      <i class="fas fa-envelope fa-2x mr-2 text-light"></i>
                    </a>
                    <a href="https://www.linkedin.com/in/harshit-dixit-38b27a149" target="_blank">
                      <i class="fab fa-linkedin fa-2x mx-2 text-light"></i>
                    </a>
                    <a href="https://github.com/harshitdixit69" target="_blank">
                      <i class="fab fa-github fa-2x ml-2 text-light"></i>
                    </a>
                  </p>
                </div>
              </div>
            </div>
            <div class="col">
              <div class="card h-100 border text-white text-center">
                <img src="{% static 'core/images/abhishek.jpg' %}" class="card-img-
top" alt="abhishek_kumar_yadav">
                <div class="card-body trans2">
                  <h3 class="card-title">Abhishek Kumar Yadav</h3>
                  <h4>Project Testing</h4>
                  <p class="card-text">
                    <a href="mailto:abhi991926@gmail.com" target="_blank">
                      <i class="fas fa-envelope fa-2x mr-2 text-light"></i>
                    </a>
                    <a href="https://www.linkedin.com/in/abhishek-kumar-yadav-
780262194" target="_blank">
                      <i class="fab fa-linkedin fa-2x mx-2 text-light"></i>
```

```
                </a>
                <a href="https://github.com/Abhishek991926" target="_blank">
                    <i class="fab fa-github fa-2x ml-2 text-light"></i>
                </a>
            </p>
        </div>
    </div>
  </div>
</div>
{% endblock content %}
```

Website/red_mind/core/templates/core/base.html-

```
<!DOCTYPE html>
<html lang="en">
{% load static %}
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="shortcut icon" href="{% static 'core/images/logo.png' %}" type="image/x-icon">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Akaya+Kanadaka&family=Noto+Sans+
KR&display=swap" rel="stylesheet">
  <link rel="stylesheet" href="{% static 'core/css/bootstrap.min.css' %}">
  <link rel="stylesheet" href="{% static 'core/css/all.min.css' %}">
  <link rel="stylesheet" href="{% static 'core/css/style.css' %}">
  <title>Poject Red Mind : {% block title %}{% endblock title %}</title>
</head>
<body>

  {% include 'core/navbar.html' %}

  <div class="container-fluid {% block bg %}bg-home{% endblock bg %}">
    {% block content %}
    {% endblock content %}
  </div>

  {% include 'core/footer.html' %}

  <script src="{% static 'core/js/bootstrap.min.js' %}"></script>
  <script src="{% static 'core/js/popper.js' %}"></script>
  <script src="{% static 'core/js/all.min.js' %}"></script>
</body>
</html>
```

Website/red_mind/core/templates/core/cast.html-

```
{% extends 'core/base.html' %}
{% load static %}
{% block title %}Cast{% endblock title %}
{% block bg %}bg-cast{% endblock bg %}
{% block content %}
<div class="container pt-5 pb-2">
  <div class="row text-center mt-2">
    <div class="alert alert-secondary h2" role="alert">
      Cast Details
    </div>
  </div>
  {% if name %}
  <div class="row my-3 pb-5">
    <div class="col-md-4 col-sm-12 text-center my-2">
      <img width="300px" class="border rounded shadow-
lg" src={{profile_path}} alt="profile">
    </div>
    <div class="col-md-8 col-sm-12 my-2 trans2 p-3 border rounded shadow-lg text-light">
      <p class="h3">Name: {{name}}</p>
      {% if biography %}
        <p><span class="h5">Biography: </span>{{biography}}</p>
      {% endif %}
      {% if birthday %}
        <p><span class="h5">Birthday: </span>{{birthday}}</p>
      {% endif %}
      {% if place_of_birth %}
        <p><span class="h5">Place of Birth: </span>{{place_of_birth}}</p>
      {% endif %}
      {% if deathday %}
        <p><span class="h5">Deathday: </span>{{deathday}}</p>
      {% endif %}
      {% if profession %}
        <p><span class="h5">Profession: </span>{{profession}}</p>
      {% endif %}
    </div>
  </div>
  {% else %}
    <div class="pb-5">
      <p class="trans2 text-responsive border rounded text-center text-light p-
5">Introduction</p>
    </div>
  {% endif %}
</div>
{% endblock content %}
```

Website/red_mind/core/templates/core/footer.html-

```
<footer class="container-fluid p-0">
  <div class="bg-dark text-center p-1 text-white">
    <div>
      <a href="https://youtu.be/3-Ko3rMPN38" target="_blank">
        <i class="fab fa-youtube fa-2x m-1 text-white"></i>
      </a>
      <a href="https://github.com/IntegratedMindHeart" target="_blank">
        <i class="fab fa-github fa-2x m-1 text-white"></i>
      </a>
      <a>
        <i class="fab fa-linkedin fa-2x m-1 text-white"></i>
      </a>
    </div>
    <p>Copyright &#169; 2020-21 Team Integrated Mind Heart</p>
  </div>
</footer>
```

Website/red_mind/core/templates/core/home.html-

```
{% extends 'core/base.html' %}
{% load static %}
{% block title %}Home{% endblock title %}
{% block content %}
<div class="container">
  <div class="row pt-5">
    <div class="col-12">
      <p class="text-responsive text-center text-
white fstyle">Enter your favorite movie and get movies recommendation
      </p>
    </div>
  </div>
  <div class="row justify-content-center p-md-5">
    <form action="{% url 'home' %}" method="POST">
      {% csrf_token %}
      <div class="input-group">
        <div class="input-group mb-3 shadow-lg">
          <span class="input-group-
text" id="{{form.movie_name.id_for_label}}">{{form.movie_name.label}}</span>
            {{form.movie_name}}
            <button type="submit" class="btn btn-danger">Go</button>
        </div>
      </div>
    </form>
  </div>
  {% if movie_title %}
  <div class="row text-center">
    <div class="alert alert-danger h2" role="alert">
      Movie Details
```

```
    </div>
  </div>
  <div class="row my-3 pb-5">
    <div class="col-md-4 col-sm-12 text-center my-2">
      <img width="300px" class="border rounded shadow-lg" src={{poster}} alt="poster">
    </div>
    <div class="col-md-8 col-sm-12 my-2 trans p-3 border rounded shadow-lg text-light">
      <p class="h1">Title: {{movie_title}}</p>
      <p><span class="h5">Tagline: </span>{{tagline}}</p>
      <p><span class="h5">Overview: </span>{{overview}}</p>
      <p><span class="h5">Rating: </span>{{rating}}/10 ({{vote_count}} votes)</span>
      <p><span class="h5">Genres: </span>
        {% for genre in genre_list %}
        {{genre}},
        {% endfor %}</p>
      <p><span class="h5">Release Date: </span>{{release_date}}</p>
      <p><span class="h5">Runtime: </span>{{runtime}} minuts</p>
      <p><span class="h5">Status: </span>{{status}}</p>
    </div>
  </div>
  {% if casts %}
  <div class="row text-center pb-4">
    <div class="alert alert-danger h2" role="alert">
      Movie Cast Details
    </div>
  </div>
  <div class="row row-cols-1 row-cols-md-3 g-4 pb-5">
    {% for cast in casts %}
    <div class="col">
      <div class="card h-100 text-light text-center border">
        <img src="{{cast.image}}" class="card-img-top" alt="cast-img">
        <div class="card-body trans border">
          <h5 class="card-title">{{cast.original_name}}</h5>
          <p class="card-text">({{cast.character}})</p>
        </div>
        <div class="card-footer trans">
          <a href="{% url 'cast' cast.cast_id %}" style="text-decoration:none">
            <small>Detailed View</small>
          </a>
        </div>
      </div>
    </div>
    {% endfor %}
  </div>
  {% endif %}
  {% if reviews %}
  <div class="row text-center">
    <div class="alert alert-danger h2" role="alert">Movie Reviews</div>
```

```
    </div>
    <table class="table table-striped table-condensed table-bordered rounded trans my-2">
        <tr class="text-light">
            <th>S.No</th>
            <th>Review</th>
            <th>Sentiment</th>
        </tr>
        {% for review in reviews %}
        <tr class="text-light">
            <th class="active">{{forloop.counter}}</th>
            <td class="active">{{review.review}}</td>
            {% if review.pred %}
            <td class="active"><i class="fas fa-thumbs-up"></i> (Good)</td>
            {% else %}
            <td class="active"><i class="fas fa-thumbs-down"></i> (Bad)</td>
            {% endif %}
        </tr>
        {% endfor %}
    </table>
    {% endif %}
    {% if movie_posters %}
    <div class="row text-center mt-5 mb-3">
        <div class="alert alert-danger h2" role="alert">Recommended Movies</div>
    </div>
    <div class="row row-cols-1 row-cols-md-3 g-4 pb-5">
        {% for movie,poster in movie_posters %}
        <div class="col">
            <div class="card h-100 text-light text-center border">
                <img src="{{poster}}" class="card-img-top" alt="movie-img">
                <div class="card-body trans border">
                    <h5 class="card-title">{{movie|capfirst}}</h5>
                </div>
                <div class="card-footer trans">
                    <form action="{% url 'home' %}" method="POST">
                        {% csrf_token %}
                        <input hidden type="text" name="movie_name" value={{movie}}>
                        <input class="astext" type="submit" value="Detailed View">
                    </form>
                </div>
            </div>
        </div>
        {% endfor %}
    </div>
    {% else %}
    <div class="pb-5">
        <p class="trans text-responsive border rounded text-center text-light p-
5">Some thing went worng</p>
    </div>
```

```
    {% endif %}
    {% else %}
    {% if sorry %}
    <div class="pb-5">
        <p class="trans text-responsive border rounded text-center text-light p-5">{{sorry}}</p>
    </div>
    {% endif %}
    <div class="pb-5">
        <p class="trans text-responsive border rounded text-center text-light p-4">Introduction</p>
        <div class="row pt-2 text-white text-center">
            <div class="col-12 col-md-4 pb-2">
                <div class="border rounded p-4 shadow trans">
                    <div class="pt-3 pb-
4">Stories make us feel old emotions and new perspectives, bringing our loved ones together. We
 give you this choice to tailor your viewing experience based on your preferences.</div>
                </div>
            </div>
            <div class="col-12 col-md-4 pb-2">
                <div class="border rounded p-4 shadow trans">
                    <div class="py-
2">Welcome to PRM! Your guide to finding your next watchlist. Whatever your taste, and no ma
tter where you live, we will give you the best recommendation based on your mood and preferenc
es.</div>
                </div>
            </div>
            <div class="col-12 col-md-4 pb-2">
                <div class="border rounded p-4 shadow trans">
                    <div class="py-2">Our browsable database includes movies -
 with their cast, biographies, and review. That will help you find the best movie to watch next whi
le interacting and giving your opinion to a large community of users.</div>
                </div>
            </div>
        </div>
    </div>
    {% endif %}
</div>
{% endblock content %}
```

Website/red_mind/core/templates/core/navbar.html-

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="{% url 'home' %}"><i class="fas fa-
film"></i> Project Red Mind</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="Toggle navigation">
```

```
    <span class="navbar-toggler-icon"></span>
   </button>
   <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav me-auto mb-2 mb-lg-0">
     <li class="nav-item">
      <a class="nav-
link {{home_active}} {{home_disabled}}" href="{% url 'home' %}">Home <span class="sr-
only">(current)</span></a>
     </li>
     {% if cast_active %}
     <li class="nav-item">
      <a class="nav-link {{cast_active}} {{cast_disabled}}">Cast <span class="sr-
only">(current)</span></a>
     </li>
     {% endif %}
     <li class="nav-item">
      <a class="nav-link" href="http://127.0.0.1:5000/">API <span class="sr-
only">(current)</span></a>
     </li>
     <li class="nav-item">
      <a class="nav-link" href="/admin" target="_blank">Admin Panel <span class="sr-
only">(current)</span></a>
     </li>
     <li class="nav-item">
      <a class="nav-
link {{about_active}} {{about_disabled}}" href="{% url 'about' %}">About <span class="sr-
only">(current)</span></a>
     </li>
    </ul>
   </div>
  </div>
</nav>
```

Website/red_mind/core/admin.py-

```
from django.contrib import admin
from .models import MovieRecord

# Register your models here.

@admin.register(MovieRecord)
class MovieRecordAdmin(admin.ModelAdmin):
   list_display=('id','movie_name','date_time')
```

Website/red_mind/core/apps.py-

```
from django.apps import AppConfig
```

```
class CoreConfig(AppConfig):
    name = 'core'
```

Website/red_mind/core/forms.py-

```
import requests
import bs4 as bs

KEY='ff5d4bbd48363bb3f65e4c34e204d94b'

def get_movie_cast(api_key,movie_id):
    try:
        casts=[]
        r=requests.get(f'https://api.themoviedb.org/3/movie/{movie_id}/credits?api_key={api_key}'
)
        for i in r.json()['cast'][:11]:
            cast={}
            if i['profile_path']:
                cast['cast_id']=i['id']
                cast['original_name']=i['original_name']
                cast['character']=i['character']
                cast['image']='https://image.tmdb.org/t/p/original'+i['profile_path']
            else:
                continue
            casts.append(cast)
        return casts
    except:
        return

def get_details(api_key,title):
    context={}
    try:
        r1=requests.get(f'https://api.themoviedb.org/3/search/movie?api_key={api_key}&query={tit
le}')
        movie_details=r1.json()['results'][0]
        movie_id=movie_details['id']
        movie_title=movie_details['original_title']

        r2=requests.get(f'https://api.themoviedb.org/3/movie/{movie_id}?api_key={api_key}')
        movie_details=r2.json()

        tagline=movie_details['tagline']
        imdb_id=movie_details['imdb_id']
        poster='https://image.tmdb.org/t/p/original'+movie_details['poster_path']
        overview=movie_details['overview']
        genres=movie_details['genres']
        rating=movie_details['vote_average']
```

```
        vote_count=movie_details['vote_count']
        release_date=movie_details['release_date']
        runtime=movie_details['runtime']
        status=movie_details['status']
        casts=get_movie_cast(KEY,movie_id)

        genre_list=[]
        for genre in genres:
            genre_list.append(genre['name'])

        context['movie_id']=movie_id
        context['movie_title']=movie_title
        context['tagline']=tagline
        context['imdb_id']=imdb_id
        context['poster']=poster
        context['overview']=overview
        context['genres']=genres
        context['rating']=rating
        context['vote_count']=vote_count
        context['release_date']=release_date
        context['runtime']=runtime
        context['status']=status
        context['genre_list']=genre_list
        context['casts']=casts

    except:
        context['sorry']='movie not found'

    return context

def get_similar_movies(movie_title,movie_id,api_key):
    try:
        r=requests.get(f'http://127.0.0.1:5000/api/recommendation/{movie_title}')
        return r.json()
    except:
        r=requests.get(f'https://api.themoviedb.org/3/movie/{movie_id}/similar?api_key={api_key}
&language=en-US&page=1')
        results=r.json()['results']
        similar_movies=[]
        for result in results:
            similar_movies.append(result['title'])
        return {'similar_movies':similar_movies}

def get_movie_posters(api_key,similar_movies):
    posters=[]
    try:
        for movie in similar_movies:
```

```
            r=requests.get(f'https://api.themoviedb.org/3/search/movie?api_key={api_key}&query={
movie}')
            poster_path=r.json()['results'][0]['poster_path']
            posters.append((movie,f'https://image.tmdb.org/t/p/original{poster_path}'))
        return {'movie_posters':posters}
    except:
        pass

def get_individual_cast(api_key,cast_id):
    try:
        details={}
        r=requests.get(f'https://api.themoviedb.org/3/person/{cast_id}?api_key={api_key}')
        cast=r.json()
        details['biography']=cast['biography']
        details['birthday']=cast['birthday']
        details['deathday']=cast['deathday']
        details['profession']=cast['known_for_department']
        details['name']=cast['name']
        details['place_of_birth']=cast['place_of_birth']
        details['profile_path']='https://image.tmdb.org/t/p/original'+cast['profile_path']
        return details
    except:
        return {'sorry':'details not found'}

def get_sentiment(review):
    r=requests.get(f'http://127.0.0.1:5000/api/review/{review}')
    return r.json()['result']

def get_reviews(imdb_id):
    sauce=requests.get('https://www.imdb.com/title/{}/reviews?ref_=tt_ov_rt'.format(imdb_id)).te
xt
    soup=bs.BeautifulSoup(sauce,"html.parser")
    soup_result=soup.find_all("div",{"class":"text show-more__control"})
    reviews=[]
    for review in soup_result:
        if review.string:
            temp={}
            try:
                temp['pred']=get_sentiment(review.string)
                temp['review']=review.string
            except:
                continue
            reviews.append(temp)
    return reviews
```

Website/red_mind/core/forms.py-

```
from django import forms
```

```
from .models import MovieRecord

class MovieRecordForm(forms.ModelForm):
    class Meta:
        model=MovieRecord
        fields=('movie_name',)
        widgets={
            'movie_name':forms.TextInput(attrs={'class':'form-
control','placeholder':'enter movie name'})
        }
```

Website/red_mind/core/models.py-

```
from django.db import models

# Create your models here.

class MovieRecord(models.Model):
    id=models.AutoField(primary_key=True)
    movie_name=models.CharField(max_length=200)
    date_time=models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.movie_name
```

Website/red_mind/core/views.py-

```
from django.shortcuts import redirect, render
from django.views import View
from .models import MovieRecord
from .forms import MovieRecordForm
from .call import KEY,get_details,get_similar_movies,get_reviews,get_movie_posters,get_indivi
dual_cast

# Create your views here.

class HomeView(View):
    def get(self,request,*args,**kwargs):
        fm=MovieRecordForm()
        context={
            'home_active':'active',
            'home_disabled':'disabled',
            'form':fm
        }
        return render(request,'core/home.html',context)

    def post(self,request,*args,**kwargs):
        fm=MovieRecordForm(request.POST)
```

```
        title=''
        if fm.is_valid():
            title=fm.cleaned_data['movie_name'].lower()
            rec=MovieRecord(movie_name=title)
            rec.save()
        fm=MovieRecordForm()
        context=get_details(KEY,title)
        if 'movie_id' in context.keys():
            context.update(get_similar_movies(title,context['movie_id'],KEY))
        if 'imdb_id' in context.keys():
            context['reviews']=get_reviews(context['imdb_id'])
        if 'similar_movies' in context.keys():
            context.update(get_movie_posters(KEY,context['similar_movies']))
        context['home_active']='active'
        context['home_disabled']='disabled'
        context['form']=fm
        return render(request,'core/home.html',context)

class CastView(View):
    def get(self,request,*args,**kwargs):
        context=get_individual_cast(KEY,kwargs['cast_id'])
        if not context:
            return redirect('home')
        context['cast_active']='active'
        context['cast_disabled']='disabled'
        return render(request,'core/cast.html',context)

class AboutView(View):
    def get(self,request,*args,**kwargs):
        context={
            'about_active':'active',
            'about_disabled':'disabled'
        }
        return render(request,'core/about.html',context)
```

Website/red_mind/red_mind/asgi.py-

```
"""
ASGI config for red_mind project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/asgi/
"""

import os
```

```
from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'red_mind.settings')

application = get_asgi_application()
```

Website/red_mind/red_mind/wsgi.py-

```
"""
WSGI config for red_mind project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/howto/deployment/wsgi/
"""

import os

from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'red_mind.settings')

application = get_wsgi_application()
```

Website/red_mind/red_mind/settings.py-

```
"""
Django settings for red_mind project.

Generated by 'django-admin startproject' using Django 3.1.1.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
```

```
SECRET_KEY = 'kuhmic47((8x3#_*p97ww@m8-*ede(+dw22cgc4ebp)*euj-p*'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'core',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'red_mind.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'red_mind.wsgi.application'
```

```
# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Asia/Kolkata'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/
```

```
STATIC_URL = '/static/'
```

Website/red_mind/red_mind/urls.py-

```
"""red_mind URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.1/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from core import views

admin.site.site_header = 'Project Red Mind Administration'
admin.site.index_title = 'Site Database Details'
admin.site.site_title = 'Project Red Mind Site Admin'

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.HomeView.as_view(), name='home'),
    path('about/', views.AboutView.as_view(), name='about'),
    path('cast/<int:cast_id>', views.CastView.as_view(), name='cast')
]
```

Website/red_mind/manage.py-

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'red_mind.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
```

```
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Website/red_mind/requirements.txt-

```
bs4==0.0.1
Django==3.2.4
gunicorn==20.1.0
requests==2.25.1
```

Website/red_mind/README.md-

```
# Project Red Mind
We have created Django based website to get information about any movie and it also provides th
e movie recommendation service with review sentiment analysis.

## See Project

visit- https://projectredmind.herokuapp.com/

## Installation

1.Use the package manager [pip](https://pip.pypa.io/en/stable/) to install all python packages on y
our system which is listed in [requirements.txt](https://github.com/IntegratedMindHeart/project_r
ed_mind/blob/main/red_mind/requirements.txt) file.

2.Open cmd or terminal

3.Go to file location were [manage.py](https://github.com/IntegratedMindHeart/project_red_min
d/blob/main/red_mind/manage.py) is located

4.Run the command *python manage.py runserver*

5.Open your browser and open url http://127.0.0.1:8000/

#### Demo Video - https://youtu.be/3-Ko3rMPN38
```

CHAPTER 9

CONCLUSION

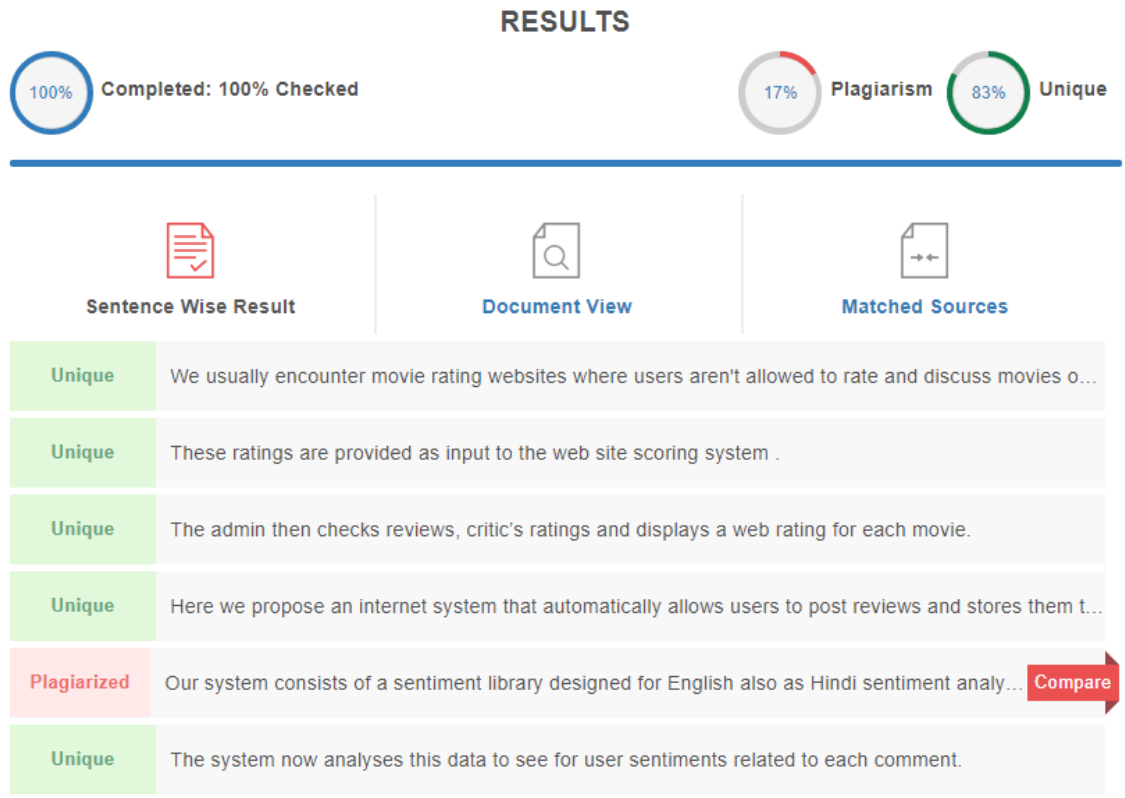
UNCOVERED KEY RESEARCH TRENDS

This solution is carefully analysed and after the evaluation of the possible solutions, the most feasible solution for this project is identified and selected, so the project turns to be cost-effective, vital and practical. Thus processing information will be faster. It guarantees accurate maintenance of details. It easily reduces the book keeping task and thus reduces the human effort and increases accuracy speed.

The Movie Review System can be entered using a username and password. It is accessible either by an administrator or registered user. Only the admin can add data into the database. The data can be retrieved easily. The data are well protected for personal use and makes the data processing very fast.

Appendix/Appendices

Plagiarism Check Report



References:

- Adomavicius G., Tuzhilin A., Extending Recommender Systems: A Multidimensional Approach (2001)
- Gencay R., Qi M., Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging, IEEE Transactions on Neural Networks, July 2001, 12(4): 726-734.
- Cai C.L., Shi Z.Z., A modular neural network architecture with approximation capability and its applications, Proceedings of the 2nd IEEE International Conference on Cognitive Informatics, 2003, pp. 60.
- Sun, Z.H., Sun, Y.X., Fuzzy support vector machine for regression estimation. IEEE International Conference on Systems, Man and Cybernetics, Oct 5-8 2003,Vol. 4, pp. 3336-3341.
- Jung S., Harris K., Webster J., Herlocker, J.L. SERF – Integrating Human Recommendations with search (2004)
- Thanh-Nghi D., Fekete J.D., Large Scale Classification with Support Vector Machine Algorithms, ICMLA 2007, Sixth International Conference on Machine Learning and Applications, Dec. 13-15 2007, pp. 7-12.
- Debnath, S., Ganguly, N., Mitra, P.: Feature weighting in content based recommendation system using social network analysis. In: Proceedings of the 17th International Conference on World Wide Web. ACM (2008)
- JCDL '09: Proceedings of the 9th ACM/IEEE-CS joint conference on Digital librariesJune 2009 Pages 213–216

GitHub & Website Link

GitHub Organization Link –

https://github.com/IntegratedMindHeart

GitHub Website Repository Link –

https://github.com/IntegratedMindHeart/project_red_mind

GitHub API Repository Link –

https://github.com/IntegratedMindHeart/api

GitHub Data Collection Repository Link –

https://github.com/IntegratedMindHeart/data_collection

GitHub ML Repository Link –

https://github.com/IntegratedMindHeart/ml_model

GitHub Documentation Repository Link –

https://github.com/IntegratedMindHeart/documentation

Project Website URL –

https://projectredmind.herokuapp.com/

Project API URL –

https://porjectredmindapi.herokuapp.com/

Babu Banarasi Das

Institute of Technology and Management

(Formerly known as Babu Banarasi Das National Institute of Technology and Management)

Plagiarism Check Report by Supervisor

1.	**Project Title Name**	Project Red Mind	
2.	**Supervisor(s) Name**	Prof. Girjesh Kumar Mishra	
3.	**Department**	Information Technology	
4.	**Year & Session**		
5.	**Students Name**	**ROLL NO.**	**Name**
		1705413050	Satyam Seth
		1705413012	Ankit Kumar Gupta
		1705413011	Alisha Parveen
		1705413028	Harshit Dixit
		1805413801	Abhishek Kumar Yadav
6.	**Similarity Content (%) (Up to 40% acceptable)**	17%	
7.	**Plagiarism detection tool applied**	Smallseotools.com	
8.	**Date of Plagiarism Check**	05/07/2021	
9.	**Research Paper published (Yes / No)**		
10.	**Title of Research Paper**		
11.	**Publisher Name**		
12.	**Impact Factor with ISBN No.**		

Supervisor Signature **Head Signature**

Officer Incharge (Exam) **Director**

College Seal