PROJECT REPORT

ON

MOVIES RECOMMENDATION SYSTEM

SUBMITTED TO

ROURKELA INSTITUTE OF MANAGEMENT STUDIES

"MASTER OF COMPUTER APPLICATION"

(2022-2024)

SUBMITTED BY

NITYA GOPAL JENA

Registration Roll No: 2205260015

MCA 4th SEMESTER

ROURKELA INSTITUTE OF MANAGEMENT STUDIES

(Affiliated to Biju Patnaik University of Technology, Odisha)

Rourkela-769015

Rourkela Institute of Management Studies

Rourkela

Department of Master in Computer Application

Rourkela Institute of Management Studies

Chhend, Rourkela-15, Odisha

Phone: 0661 2480482

Fax: 91-0661-1480665

Mail: admission_rims@rims-edu.com

Visit: RIMS (rimsedu.ac.in)

CERTIFICATE OF EXAMINATON

This is to certifies that the project report, "Movie Recommendation System," submitted by **NITYA GOPAL JENA** during the 4th semester at the Rourkela Institute of Management Studies, Rourkela, has been verified by us as original and is accepted in partial fulfilment of the requirements for the Master of Computer Applications degree at the Biju Patnaik University of Technology, Rourkela.

Internal Examiner

Rourkela Institute of Management Studies

Rourkela

Department of Master in Computer Application

Rourkela Institute of Management Studies

Chhend, Rourkela-15, Odisha

Phone: 0661 2480482

Fax: 91-0661-1480665

Mail: admission_rims@rims-edu.com

Visit: RIMS (rimsedu.ac.in)

CERTIFICATE

This is to certifies that the project report, "Movie Recommendation System," submitted by **NITYA GOPAL JENA** of MCA 4th semester of session 2022-2024, at the **Rourkela Institute of Management Studies**, Rourkela, has been verified by us as original and is accepted in partial fulfilment of the requirements for the Master of Computer Applications degree at the Biju Patnaik University of Technology, Rourkela.

He is found fit and approved for the award of **"Master in Computer Application Degree.**

I wish all success in his life.

Dean Academic

RIMS, Rourkela

Prof. Bibhudendu Panda

Head of the Department, MCA

Rourkela Institute of Management Science

Rourkela

CERTIFICATE

This is to certify that **NITYA GOPAL JENA** student of **MCA Rourkela Institute of Management Studies, Rourkela, Odisha** of session 2022-2024 has completed the project successfully.

I wish all success in his life.

(Prof. Bibhudendu Panda)

DECLARATION

I, **NITYA GOPAL JENA**, here by declare that the project report entitled "**Movie Streaming and Recommendation System**" is of my work. The above work I submitted to **"Biju Patnaik University of Technology Rourkela"** for the award of **"Master in Computer Applications" Degree.**

NITYA GOPAL JENA

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Bibhudendu Panda (H.O.D MCA) and Prof. Damodar Nayak for his guidance and support during the project work.

For giving me the chance to work on the "Movie Streaming and Recommendation System" project, I am incredibly grateful to the Rourkela Institute of Management Studies in Chhend, Rourkela.

I acknowledge the help and co-operation received from my team member in making this project.

I consider myself fortunate that I have successfully completed this project. I acknowledge my team partner gratitude to all those works and ideas that had helped me in creating this project.

NITYA GOPAL JENA

ROLL NO: 2205260015

MCA(2022-2024)

ROURKELA INSTITUTE OF MANAGEMENT STUDIES

GANTT CHART

ID	TASK NAME	NO. OF DAYS	BAR REPRESENTATION
1	Project Management		
1.1	Project Initiation	5	
1.2	Project Planning	9	
2	Analysis	15	
3	Design	25	
4	Implementation	35	
5	Testing	4	
6	Evaluation	2	

TABLE OF CONTENT

ABSTRACT

In the era of digital media, the demand for personalized content has seen a significant rise. Our project, "Movie Streaming and Recommendation System", aims to cater to this need by providing a platform that not only streams movies but also offers personalized recommendations to its users.

The system leverages advanced machine learning algorithms to analyse user behaviour and search. Based on this analysis, it generates a list of recommended movies tailored to each user's unique taste.

Furthermore, the system incorporates a user-friendly interface for seamless navigation and a robust streaming service to ensure a high-quality viewing experience. It also includes features like search functionality, categorization by genre, and user ratings, enhancing the overall usability of the system.

This project represents a step towards more personalized and user-centric digital media platforms. It combines the power of machine learning with the convenience of online streaming, paving the way for the future of digital entertainment.

This project aims to develop a personalized movie recommendation system that suggests movies to users based on their preferences and searches. The system utilizes collaborative filtering and content-based filtering techniques to provide accurate and relevant movie recommendations. This report presents the design, implementation, and evaluation of the movie recommendation system, along with an analysis of its performance and user satisfaction.

INTRODUCTION

1.1 Preface

Online streaming services have been incredibly popular in recent years, giving consumers access to an enormous selection of movies. By making recommendations, a well-designed recommendation system can improve user experience and engagement.

films that fit with their hobbies. This project's main objective is to create a reliable and accurate recommendation system that can accommodate a wide range of user preferences.

Our machine is trained in such a manner that it will break the users input into singular.

words, and then the machine will search whether the data (movie name) exists in its

memory and if not, it will web scrap it and display in the output screen.

To further our machine is trained in such a manner that it will recommend similar to any particular movie name given by the user.

1.2 Problem Statements of Project

1. Movie Streaming:

- **Quality of Streaming**: Ensuring high-quality streaming without buffering or lagging is a significant challenge, especially when users have varying internet speeds and bandwidths.
- **Scalability:** The system must be able to handle a large number of users simultaneously without any degradation in performance.
- **Security:** Protecting the content from piracy and ensuring only authorized users have access to the content is crucial.
- **User Interface:** Providing an intuitive and user-friendly interface that allows users to easily search for and select movies.

2. Movie Recommendation:

- **Personalization:** The system should be able to provide personalized recommendations based on the user's viewing history and preferences.
- **Diversity:** The recommendations should not only focus on popular movies but also include diverse and less-known movies that match the user's preferences.
- **Cold Start Problem:** The system should be able to provide relevant recommendations to new users with no viewing history.
- **Real-time Updates:** The recommendations should be updated in real-time as the user's preferences change.

Objective(s):

The main objectives of the Movie Recommendation System project are:

- Provide a seamless and high-quality movie streaming experience.
- Build a scalable system that can handle a large number of users.
- Ensure the security of the content and restrict access to authorized users.
- Design an intuitive and user-friendly interface.

Recommendation:

- Develop a personalized recommendation system.
- Ensure diversity in recommendations.
- Address the cold start problem by providing relevant recommendations to new users.
- Implement real-time updates in the recommendation system.

Personalized Movie Recommendations: Develop a recommendation system that provides personalized movie suggestions to users based on their searching and preferences.

Accuracy and Diversity: Create a recommendation engine that is both accurate and diversified in order to serve a broad spectrum of user interests.

Enhanced User Experience: Improve user experience and engagement on the online streaming platform by simplifying the movie selection process and presenting relevant movie choices.

Documentation and Reporting: Make a project report that includes all the information on the design, implementation, assessment, and upcoming improvements of the movie recommendation system.

Potential for Expansion: Design the system with scalability in mind, allowing for future enhancements, such as incorporating deep learning models, real-time user interactions, and sentiment analysis.

1.3 Motivation:

The development of the Movie Recommendation System is motivated by several factors:

Enhancing User Experience: The main goal of developing a recommendation system is to improve the streaming platforms' user experience. Despite the large movie collection, consumers frequently have trouble locating.

information that suits their objectives. Users can find movies they are likely to appreciate with the help of tailored and relevant Page | 20 movie suggestions, which increases user satisfaction and engagement with the site.

Improving Movie Discovery: There are so many great films out there that they are often overlooked. Well-crafted recommendation systems can bring to light underappreciated films and hidden treasures that viewers might not otherwise discover. This may result in a more inclusive and varied movie-watching experience for consumers.

Increasing Platform Engagement: Engagement on a platform can be greatly impacted by a well-functioning movie suggestion system. Customers are more inclined to act upon recommendations that are personalized for them and speak to their interests. extend your exploration and content consumption time on the platform. Increased user retention and loyalty may result from higher engagement.

Understanding User Preferences: The data collected through the recommendation system offers valuable insights into user preferences and behaviour. This data can be leveraged by the platform to better understand its user base, tailor marketing strategies, and optimize the content library.

Research and Innovation: Using different algorithms and strategies, such collaborative filtering, is necessary to develop a recommendation system. It offers a chance for academics and developers to investigate novel ideas and make advancements in the recommender system industry.

User Empowerment: By providing users with personalized movie recommendations, the recommendation system empowers users to discover content that resonates with their individual tastes and interests. It gives users more control over their movie-watching choices and creates a more user-centric platform. Overall, the motivation behind the Movie Recommendation System is to create a more enjoyable and efficient movie-watching experience for users, promote movie diversity, and leverage technology to better understand and cater to user preferences. By achieving these goals, the project aims to contribute to the success of online streaming platforms and enrich the overall movie-watching ecosystem.

1.4 Project Overview/ Specification:

Project Overview:

Certainly! The project, named "Movie Streaming and Recommendation System", aims to develop a platform that enhances the user's movie-watching experience by providing seamless streaming and personalized movie recommendations. The key features of this system include high-quality movie streaming capable of handling a large number of users simultaneously, and a recommendation engine that suggests movies based on users' searching and preferences. The recommendations will be diverse and updated in real-time as the user's preferences change. The system will also feature an intuitive and user-friendly interface for easy navigation. Various technologies such as machine learning algorithms, web development frameworks, and possibly cloud services will be leveraged in this project. The expected outcome is a fully functional movie streaming and recommendation system that not only allows users to stream movies seamlessly but also recommends movies that align with their tastes. This project provides a great opportunity to apply and learn various concepts of machine learning, data analysis, web development, and cloud computing. It also holds high industry relevance as recommendation systems are widely used in various sectors including entertainment, e-commerce, and more.

1.5 Project Specifications:

Data Collection:

Data Scraping:

Data scraping, also known as web scraping, is a method used in machine learning to gather large amounts of data from the internet. It plays a crucial role in various stages of machine learning, including data collection, preprocessing, and model training.

Here we scraping the data from Netflix using python in machine learning.

Dataset:

The project will collect movie data, user ratings, and user movie interactions from a reliable source or dataset. The dataset will be pre-processed to handle missing values, duplicates, and outliers. We are doing the machine learning on dataset.

Collaborative Filtering: For the purpose of recommending movies, the system will incorporate both item-based and user-based collaborative filtering techniques. The algorithms are designed to identify movies that people with similar preferences have enjoyed by taking into account user-to-user and item-to-item similarity.

Content-Based Filtering: The system will employ the content-based filtering algorithm to recommend movies based on the similarity between movie attributes and user preferences.

Cosine similarity has been used to recommend a movie.

Recommendation Engine: A hybrid recommendation engine will be developed to combine collaborative filtering and content-based filtering approaches. The engine will generate personalized and diverse movie recommendations to cater to individual user preferences.

User Interface: The project will include a user-friendly web interface or application through which users can interact with the system. The interface will allow users to input their preferences, view recommended movies, and access movie details and trailers.

Evaluation and Testing: The Recommendations are made by computing similarity scores for movies using cosine similarity. For each movie tags are created by combining various details like genre of the movie, title, top cast, director and then they are converted to vectors using which similarity matrix is formed. Then for any searched movie the movies with the largest similarity score with it are sorted and then recommended.

Documentation and Reporting: Comprehensive documentation will be created for each module, including algorithms, functions, and data structures used. A project report will be written, detailing the design, implementation, evaluation, and future enhancements of the movie recommendation system.

Scalability and Future Enhancements: The system will be designed with scalability in mind, allowing for potential future enhancements. Future possibilities may include integrating deep learning models, incorporating real-time user interactions.

Deployment and Integration: The movie recommendation system will be deployed to a production environment or a cloud-based platform. Integration with an existing online streaming platform or the creation of an independent movie recommendation service will be considered.

User Support and Maintenance: Post-deployment, user support will be provided to address any issues that arise. Regular maintenance and updates will be performed to ensure the system runs smoothly and remains up to date.

1.6 Operating System:

The Movie Recommendation System is a web-based application, and hence it is platform-independent.

It can run on any operating system that supports a modern web browser, including but not limited to:

- **Windows 7 or later**
- **macOS Sierra (10.12) or later**
- **Ubuntu 16.04 or later**
- **Fedora 24 or later**

Web Browser:

The system is designed to work with modern web browsers, where we use React JS, MongoDB, SQL, Python3, CSS, and JavaScript ES6 and flask.

The following browsers are recommended:

- **Google Chrome 58 or later**
- **Mozilla Firefox 54 or later**
- **Safari 10 or later**
- **Microsoft Edge 14 or later**

Internet Connection:

As the system is web-based, a stable internet connection is required for accessing the movie database and processing user requests.

Hardware:

- **Processor: 1GHz or faster**

- **RAM: 1GB or more**
- **Screen Resolution: 1024 x 768 or higher.**

1.7 Programming Languages:

Python: Python will be the primary programming language used for developing the recommendation algorithms, data processing, and other backend functionalities.

React.js, SCSS, : These web technologies will be used to create the user interface for the web application.

Dependencies used in React.js:

- @reduxjs/toolkit": "^1.9.1
- axios": "^1.2.2
- dayjs": "^1.11.7
- react": "^18.2.0
- react-circular-progressbar": "^2.1.0
- react-dom": "^18.2.0
- react-icons": "^4.7.1
- "react-infinite-scroll-component ": "^6.1.0
- react-lazy-load-image-component": "^1.5.6"
- react-player": "^2.11.0
- react-redux": "^8.0.5
- react-router-dom": "^6.6.2
- react-select": "^5.7.0
- sass": "^1.57.1

Libraries used in Python:

- import pandas as pd
- from flask.helpers import send_from_directory
- from flask_cors import CORS, cross_origin
- from sklearn.feature_extraction.text import CountVectorizer
- from sklearn.metrics.pairwise import cosine_similarity
- from flask import Flask, request, render_template, jsonify

User Interface and Frontend:

React.js/SCSS : The frontend will be designed using React.js for structuring the content, SCSS for styling.

Version Control:

Git: Git will be used for version control to track changes, manage codebase collaboration, and facilitate the integration of new features. Deployment.

Documentation:

Markdown: Markdown can be used for creating documentation files and the project report, making it easy to maintain and share project information.

The technology and techniques that will be utilized to implement the various components of the Movie Recommendation System are described in the software specifications.

These specs meet the goals and specifications of the project and guarantee the successful creation, implementation, and upkeep of the recommendation system.

1.8 Organization of the project:

The organization of the Movie Recommendation System project involves dividing the development process into several logical components or modules. Each module Page | 23 is responsible for specific functionalities, making the project more manageable and easier to maintain. Here's a suggested organization for the project.

1.9 Data Collection and Preprocessing:

Obtain movie data from reliable sources or datasets and API.

Preprocess the data to handle missing values, duplicates, and outliers.

Perform feature engineering to extract relevant movie attributes.

2. Content-Based Filtering:

Implement the content-based filtering algorithm.

Calculate similarity scores based on movie attributes.

Evaluate the content-based filtering approach.

2.1 Recommendation:

Develop a recommendation engine that content-based filtering. Fine-tune the recommendation algorithm to provide personalized and diverse recommendations.

2.2 User Support and Maintenance:

After deployment, assist users and resolve any problems that may come up. To maintain the system current and functioning properly, perform routine updates and maintenance. Recall that the size of the team, the development environment, and certain needs can all affect how the project is organized. For the project to be completed successfully, team members must continue to collaborate and communicate clearly at all times.

LITERATURE STUDY

3.1 Why Do We Need Movie Streaming Services:

Movie Streaming Services have become an integral part of our digital lives for several reasons:

1) **Immediate Access:** Streaming is immediate, allowing content to start playing more or less instantly, regardless of how large the audio or video file is.

2) **No Storage Required:** Streaming doesn't required storage space. You don't need a large hard drive to hold content you want to watch or listen to.

3) **Diverse Content:** Streaming services facilitate the production and distribution of more diverse and niche content. They offer a vast library of movies, TV shows, documentaries, and more from various genres and languages.

4) **Convenience:** streaming services provide unparalleled access to a vast library of content, changing the dynamics of how movies are produced, distributed, and consumed.

5) **Cost-Effective:** Compared to traditional cable or satellite TV subscriptions, most streaming services are more affordable.

6) **Personalized Recommendations:** streaming services use advanced algorithm to analysed viewing habits and provide personalized content recommendations.

3.2 Why Do We Need Recommendation Systems:

It has been said that we are living in the "era of abundance." There are potentially thousands of options available for any given product. Consider the aforementioned instances: social media, internet commerce, streaming videos—the list is endless. Recommender systems aid in platform personalization and assist users in finding content they enjoy.

To do this, it is easiest and simplest to suggest the most well-liked products. Personalized recommendations can improve user experience, but only with specialized recommender systems.

From a business perspective, user engagement increases with more relevant products found on the site. The platform itself frequently sees an increase in revenue as a result. According to a number of reports, recommendations account for as much as 35–40% of the money generated by internet companies.

Now that we understand the importance of recommender systems, let's have a look at types of recommendation systems, then build our own with open-source data.

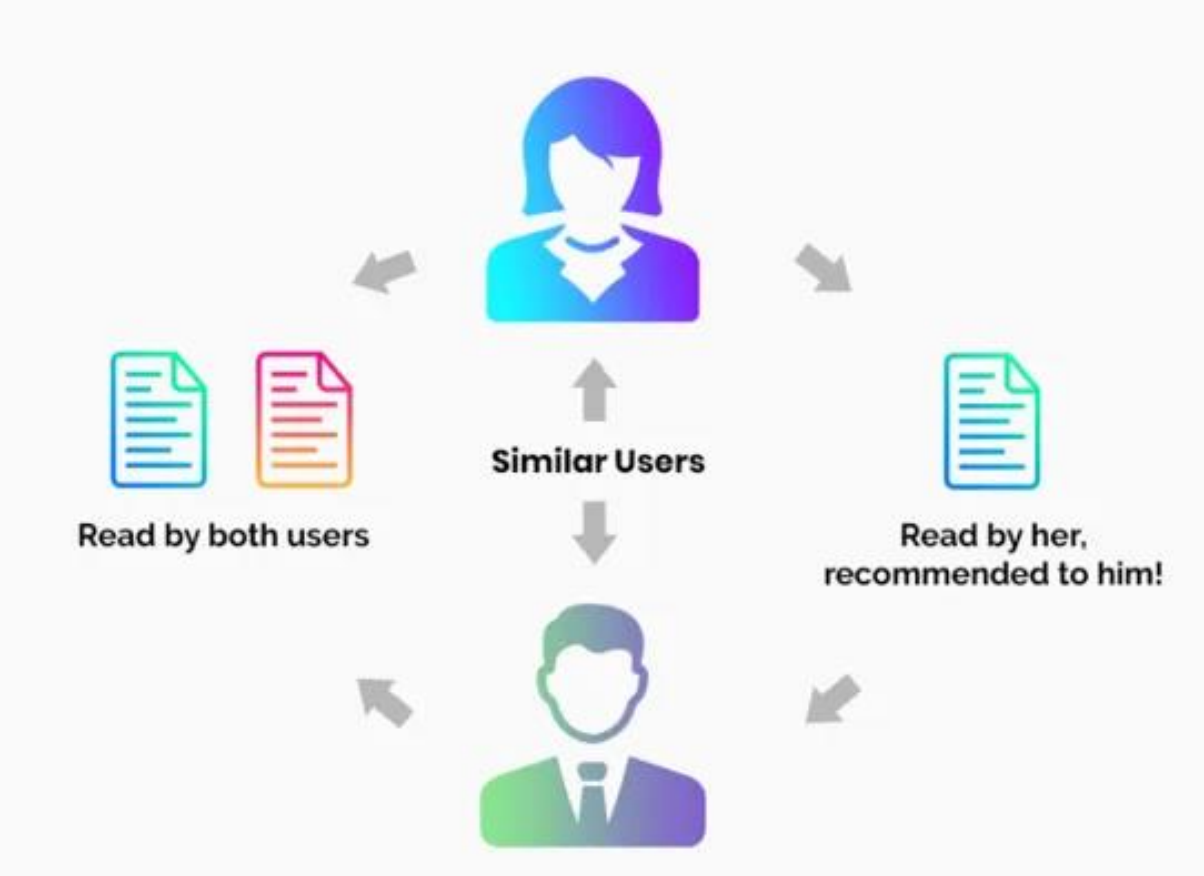
Types Of Recommendations Systems:

Making product suggestions is one of the well-known uses of machine learning, however it can handle many other issues as well. Recommendation systems fall into three categories –

Collaborative Filtering:

Items are recommended via Collaborative Filtering based on similarity metrics between users and/or items. The algorithm operates under the fundamental premise that people who share similar interests would have comparable preferences.

Concept: CF is based on the idea that people who agreed in their evaluation of certain items are likely to agree again in the future.



Example:

If user A likes Apples, Bananas, and Mango while user B likes Apples, Bananas, and Jackfruit, they have similar interests. So, it is highly likely that A would like Jackfruit and B would enjoy Mango. This is how collaborative filtering takes place.

Two kinds of collaborative filtering techniques used are:

User-User collaborative filtering is a type of recommendation system that makes predictions for a user based on the preferences of similar users. It works by finding users with similar tastes and recommending items they liked to the target user.

Item-Item collaborative filtering, on the other hand, recommends items to a user based on the preferences for similar items. It works by identifying items that are similar to the ones a user has liked in the past and recommending them to the user.

Content-Based Filtering:

Content-based recommendation systems recommend items to users based on their past preferences and behaviours. This type of system analyses the user's historical data, such as their search history, browsing history, or purchase history, and recommends items that are similar to the ones the user has interacted with before.

Some key components of a Content-Based Filtering system:

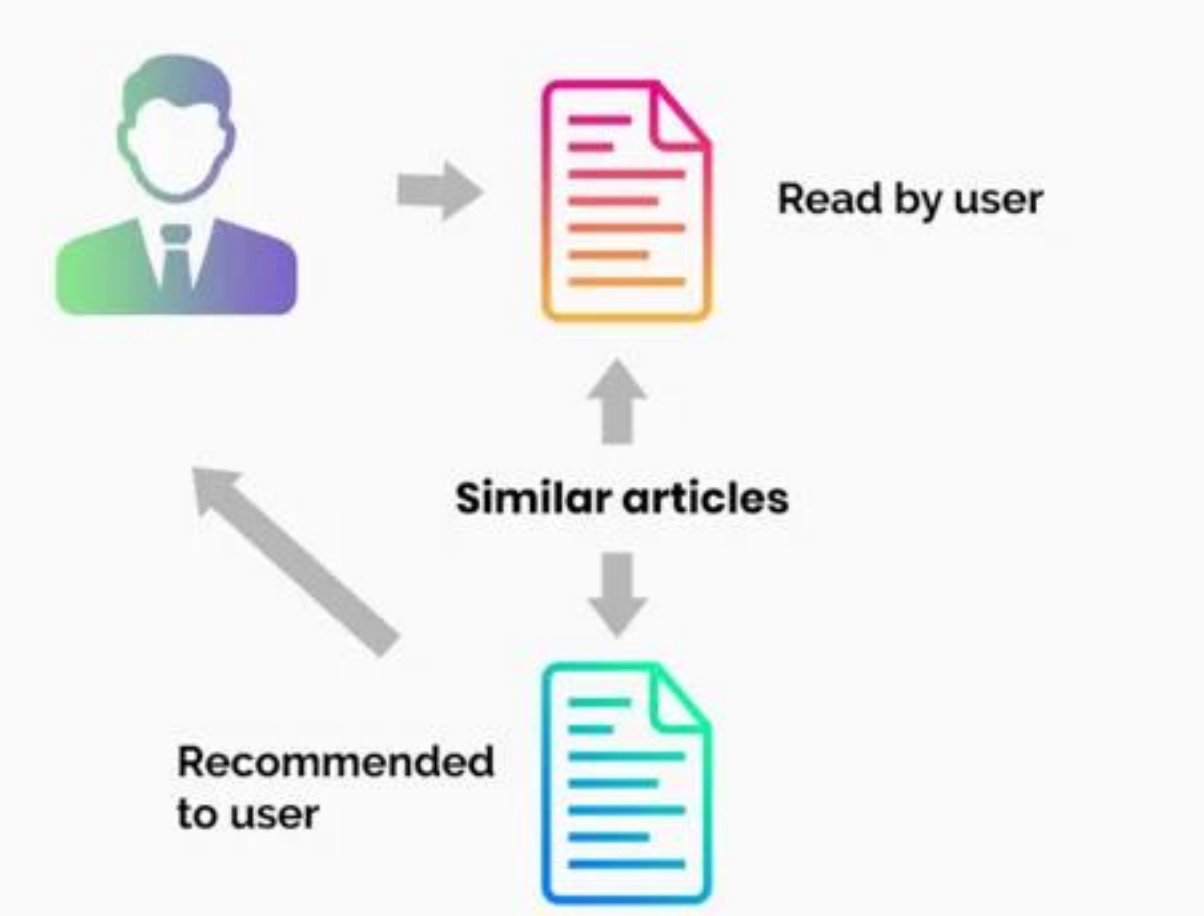
User Profile: We build vectors in the User Profile to represent the user's preferences. The utility matrix, which explains the link between the user and the object, is used in the building of the user profile.

Item Profile: In Content-Based Recommender, we must build a profile for each item, which will represent the important characteristics of that item.

Utility Matrix: Utility Matrix signifies the user's preference with certain items. In the data gathered from the user, we have to find some relation between the items which are liked by the user and those which are disliked, for this purpose we use the utility matrix.

Example:

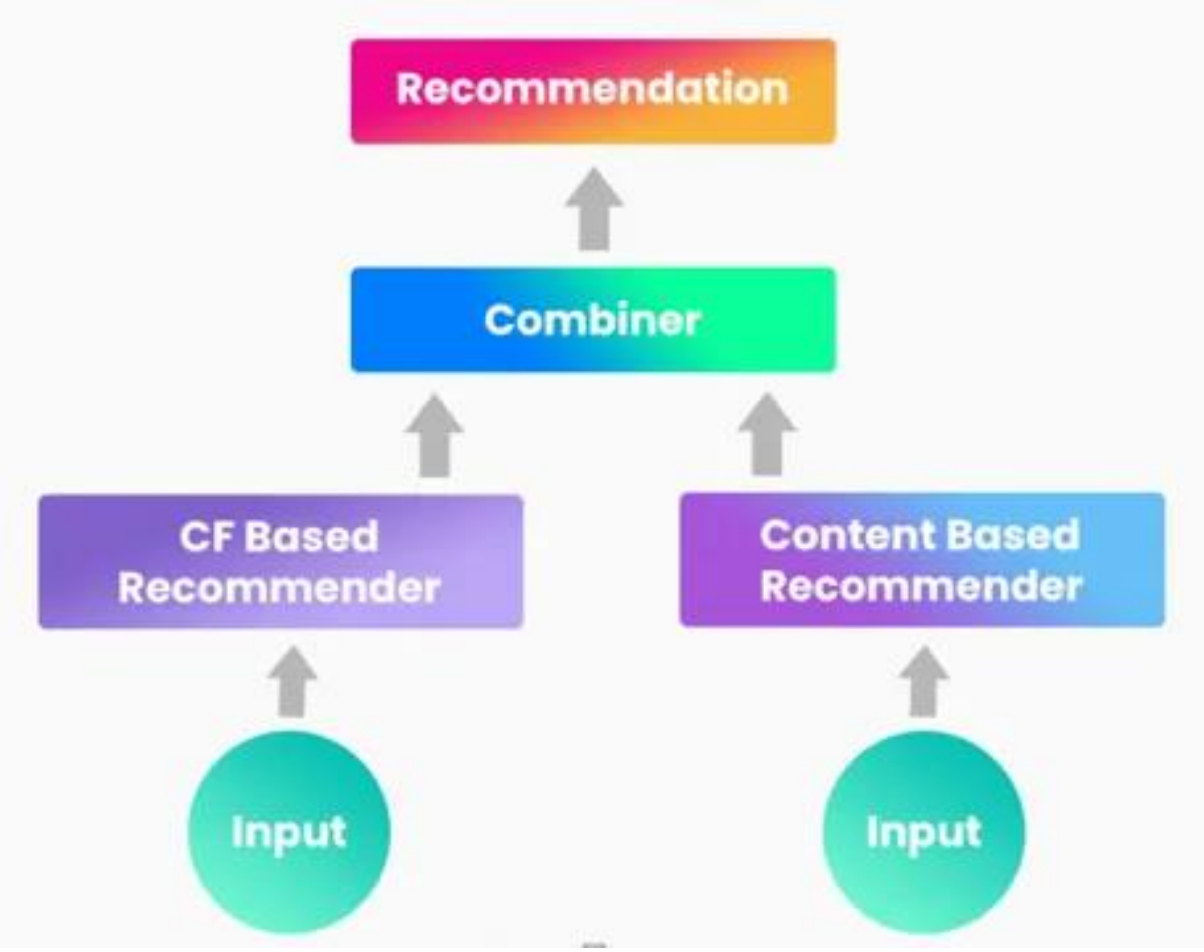
if a user has watched several action movies in the past, a content-based recommendation system might recommend similar action movies to the user. if a user likes to watch movies such as Iron Man, the recommender system recommends movies of the superhero genre or films describing Tony Stark.



Hybrid-Recommendation System:

Hybrid recommendation systems combine both content-based and collaborative filtering techniques to provide more accurate and diverse recommendations. This type of system uses a combination of user data, item data, and other contextual information to generate recommendations.

Hybrid recommendation system might use content-based filtering to recommend items that are similar to the ones the user has interacted with before, and collaborative filtering to recommend items that other similar users have liked or interacted with. By combining the strengths of both approaches, hybrid recommendation systems can provide more accurate and diverse recommendations than either content-based or collaborative filtering alone.



Netflix is an excellent case in point for a hybrid recommendation system. It makes recommendations by juxtaposing users' watching and searching habits and finding similar users on that platform. This way, Netflix uses collaborative filtering.

By recommending such shows/movies that share similar traits with those rated highly by the user, Netflix uses content-based filtering. They can also veto the common issues in recommendation systems, such as cold start and data insufficiency issues.

Disadvantages:

Increased Complexity: The integration of multiple techniques can lead to increased complexity in system design and implementation.

Computational Resources and Processing Time: Depending on the specific methods used, hybrid recommendation systems may require increased computational resources and processing time.

Cold Start Problem: Hybrid recommendation systems may struggle with the "cold start" problem, where the system cannot recommend products to new users who have not had any interaction yet, or recommend an item that users never selected before.

Large Database of Ratings: Maintaining a large database of ratings can be challenging and resource intensive.

Unintended Consequences: Research shows that recommendation systems do more than just reflect consumer preferences — they actually shape them. This can fuel biases and affect sales in unexpected ways.

Disadvantages of Content-Based Filtering:

Domain Knowledge Requirement: Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domains knowledge.

Limited to Existing Interests: The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

Overspecialization: Content-based filtering algorithms are prone to overspecialization2. This means they tend to recommend items that are too similar, limiting the diversity of the recommendations.

Detailed Information Requirement: Content-based filtering algorithms require detailed information about items and content, which is sometimes not available.

Lack of Trust: Users may not trust the recommendations if they perceive the system as being too intrusive or not understanding their preferences.

Disadvantages of User-Based Collaborative Filtering:

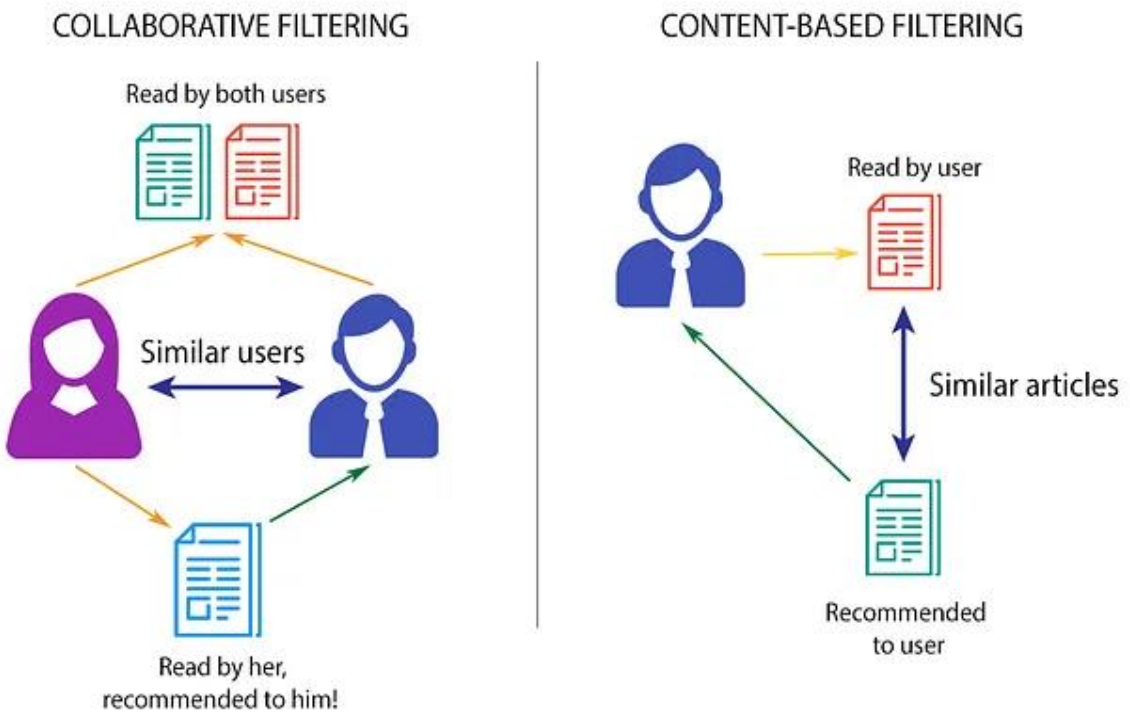
Fickle User Preferences: User preferences can change over time, leading to initial similarity patterns between users becoming outdated. This can result in inaccurate recommendations as users' tastes evolve.

Large Matrices: As the number of users is typically much larger than the number of items, maintaining large matrices becomes challenging and resource intensive. Regular recomputation is required to keep the data up to date.

Vulnerability to Shilling Attacks: Shilling attacks involve creating fake user profiles with biased preference patterns to manipulate the recommendation system. User-based collaborative filtering is susceptible to such attacks, potentially leading to biased and manipulated recommendations.

Advantages over User-based Collaborative:

Filtering includes stable movie preferences, as movies do not change like people's tastes do. Additionally, maintaining, and computing matrices is easier as there are usually fewer items than users. Shilling attacks are also more challenging since items cannot be faked, making this approach more robust.



METHODOLOGY

4.1 Data Collection:

For this project, we utilized a dataset containing movie information, user ratings, and user-movie interactions. The dataset was obtained from [source name or link]. The dataset consisted of X movies and Y users.

4.1.1 Data Processing:

Data preprocessing was performed to clean the dataset, handle missing values, and normalize user ratings. Additionally, feature engineering was carried out to extract relevant information from movie metadata.

4.2 Collaborative Filtering:

Collaborative filtering is a technique that recommends movies based on user behavior and preferences. We implemented user-based collaborative filtering and item-based collaborative filtering algorithms to generate personalized recommendations.

4.3Content-Based Filtering:

Content-Based Filtering is a type of recommendation system that personalizes suggestions based on a user's activities1. It works by using the data that we take from the user, either explicitly (rating) or implicitly. This data is used to create a user profile, which is then used to suggest items to the user.

The process of how a Content-Based Recommendation Engine works:

Creating User Profile: In the User Profile, vectors are created that describe the user's preference. The utility matrix, which describes the relationship between user and item, is used in the creation of a user profile.

Creating Item Profile: In Content-Based Recommender, a profile for each item is built, which represents the important characteristics of that item.

Utility Matrix: Utility Matrix signifies the user's preference with certain items. In the data gathered from the user, some relation is found between the items which are liked by the user and those which are disliked. For this purpose, the utility matrix is used.

Recommending Items to User Based on Content: There are different methods to recommend items to the user based on content.

Method 1: The cosine distance between the vectors of the item and the user can be used to determine its preference to the user.

Method 2: A classification approach can be used in the recommendation systems too, like the Decision Tree for finding out whether a user wants to watch a movie or not.

Three key design goals are efficiency, scalability, and modularity in the Movie Recommendation System architecture. The system is comprised of multiple interrelated elements that collaborate to furnish customers with precise and customized movie suggestions. The system architecture at a high level is shown below:

4.3.1 User Interface(Frontend):

The user interface will be developed using React.js, SCSS, providing a user-friendly web application for users to interact with the recommendation system.
Users can input their preferences, view recommended movies, and access movie details and trailers.

4.3.2 Web server (Backend):

The web server will be implemented using Flask, serving as the backend of the web application.

It handles HTTP requests from the frontend and interacts with other components of the system.

4.3.3 Data Collection and processing:

This component is responsible for collecting movie data, user ratings, and user-movie interactions from the database or external sources. Data preprocessing is performed to handle missing values, duplicates, and outliers. Feature engineering is carried out to extract relevant movie attributes.

4.3.4 Collaborative Filtering Module:

User-based collaborative filtering and item-based collaborative filtering are the two algorithms included in the collaborative filtering module.
These algorithms make movie recommendations based on consumers' past tastes by utilizing user-to-user and item-to-item similarity.

4.3.5 Content-Based Filtering Module:

The content-based filtering algorithm is utilized by the content-based filtering module. To suggest movies that users would enjoy, it computes similarity scores between user preferences and movie attributes.

4.4 Hybrid Recommendations Engine:

Hybrid suggestions are produced by the recommendation engine by combining the results of content-based and collaborative filtering modules.
It makes use of both content-based and collaborative methods to provide users with accurate and varied movie recommendations.

4.4.1 Documentation and Reporting:

The project documentation includes detailed information on each component, algorithm, and function used in the system. A comprehensive project report details the design, implementation, evaluation, and future enhancements of the movie recommendation system.

4.4.2 Deployment and Integration:

The deployed system can run on a cloud-based platform (optional) like AWS, Google Cloud, or Azure for scalability and accessibility.

The recommendation system can be integrated with an existing online streaming platform or function as an independent movie recommendation service.

The proposed architecture ensures that the Movie Recommendation System is modular, flexible, and easy to maintain. It effectively combines collaborative filtering and content-based filtering to deliver personalized and accurate movie recommendations, ultimately enhancing user experience and engagement on the online streaming platform.

4.5 Feasibility Study :

4.5.1 Economic Feasibility

1. This project is economically feasible as it doesn't require much. The only requirement is of a web browser along with any operating system.

2. This is free for viewing on World Wide Web as a registered domain name. 3. The people who use this can connect to the browser and this is the only thing that acts as a cost that is incurred in this application.

4.5.2 Technical Feasibility

For a successful development of this application all we need is an internet connection, a database server, and a supporting web server. This can be successfully implemented in Microsoft Edge (or any browser) environment and Internet Explorer.

About ReactJS

5. What is React:

React is the most popular JavaScript library for building user interfaces. It is fast, flexible and it also has a strong community sitting online to help you every time. The coolest thing about React is it's based on components, you break down your complex code into individual pieces, components and that helps developers in organizing their code.

5.1 Concept of React:

Let's say one of your friends posted a photo on Facebook. If you like the photo and then you started checking out the comments too. Now as you are browsing over comments you can see that the likes count has increased by 100 since you liked the picture, even without reloading the page. This magical change in count is due to ReactJS.

5.1.1 Uses of React:

ReactJS, a flexible JavaScript library, is widely used to create dynamic user interfaces. Its main application is in building single-page applications(SPAs), allowing smooth content updates without page reloads. With a focus on reusable components and user-friendly syntax, React simplifies UI development, excelling in real-time applications and seamlessly integrating with backends.

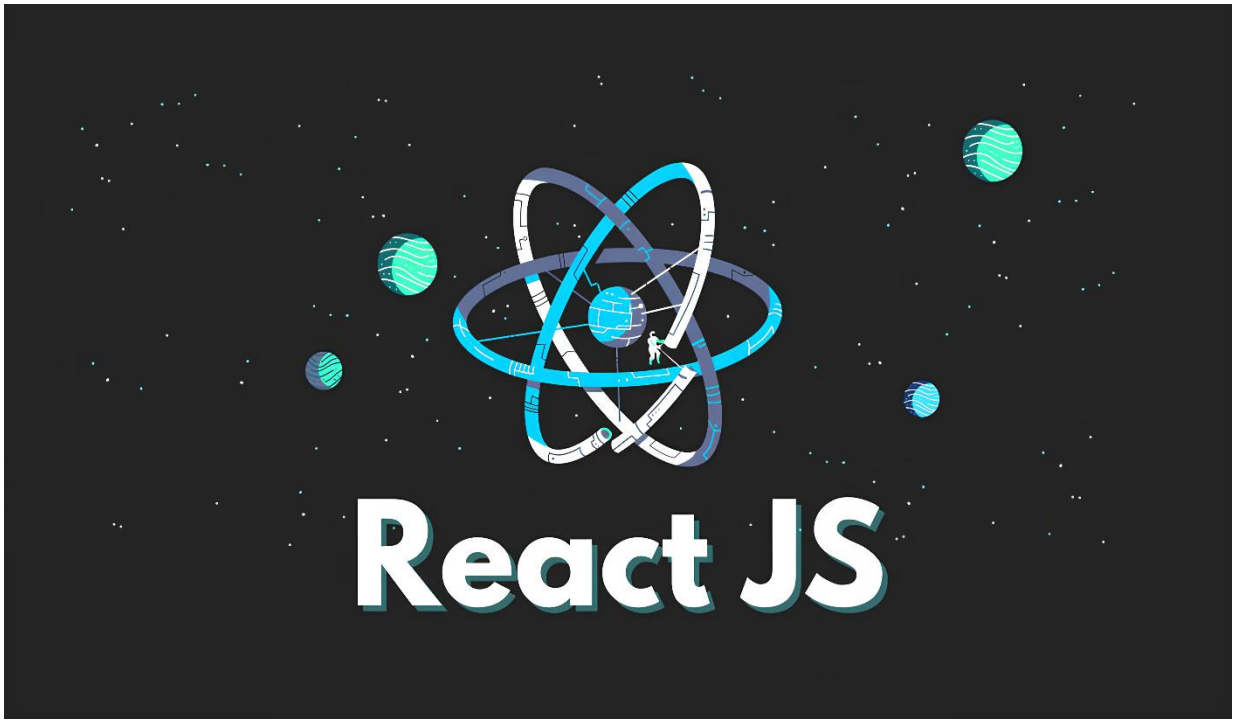
With the help of React to change data in the page without reloading the page. It is very easy to use and quick. It stands as view in MVC template. It uses virtual DOM rather than real DOM which is faster.

5.1.2 Main Features of React:

React is one of the most demanding JavaScript frameworks because it is equipped with a ton of features which makes it faster and production ready.

5.1.3 Is React a framework or library ?:

Its is very confusing to a lot of people that React is a framework of a library. React is considered as a library rather than a framework. While I the framework there is a controlled way of structure to write the code whether in library you are free to write without any structural restriction.



5.2 Why we use ReactJS ?:

The main objective of ReactJS is to develop User Interface (UI) that improves the speed of the apps. It uses virtual DOM (JavaScript object), which improves the performance of the app. The JavaScript virtual DOM is faster than the regular DOM. We can use ReactJS on the client and server-side as well as with other frameworks. It uses component and data patterns that improve readability and helps to maintain larger apps.

REACT ENVIRONMENT SETUP:

Pre-requisite for ReactJS:

1. NodeJS and NPM
2. React and React DOM
3. Webpack
4. Babel

Ways to install ReactJS:

There are two ways to set up an environment for successful ReactJS application. They are given bellow.

- Using the npm command.
- Using the create-react-app command.

What is NodeJS:

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project.

Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant.

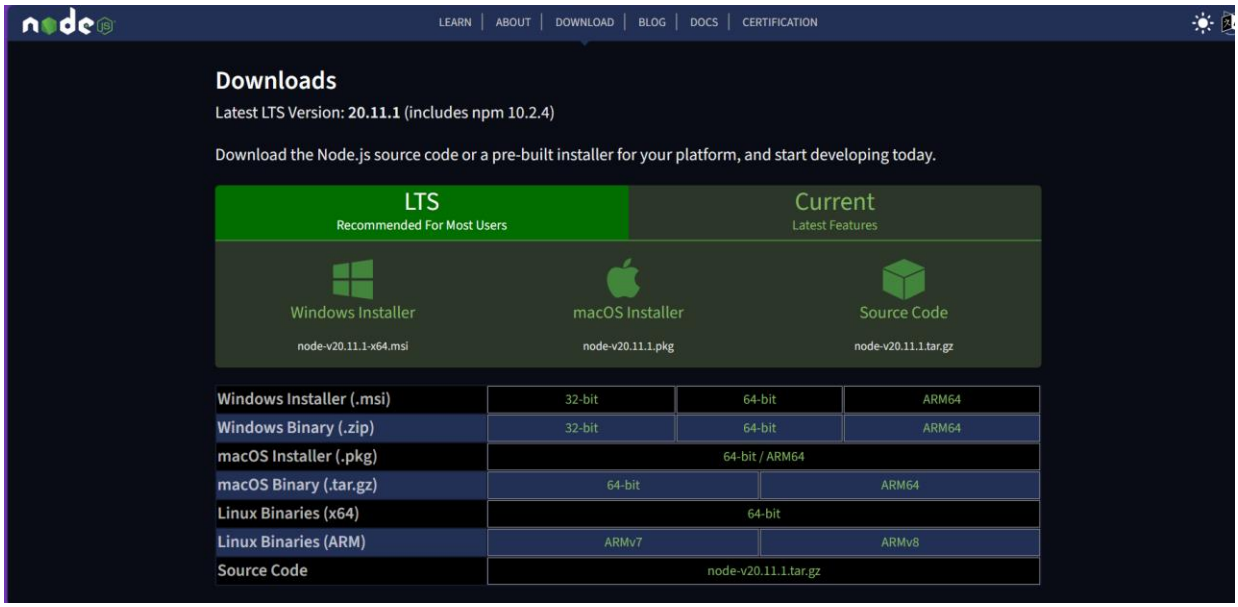
A Node.js app runs in a single process, without creating a new thread for every request. Node.js provides a set of asynchronous I/O primitives in its standard library that prevent JavaScript code from blocking and generally, libraries in Node.js are written using non-blocking paradigms, making blocking behavior the exception rather than the norm.

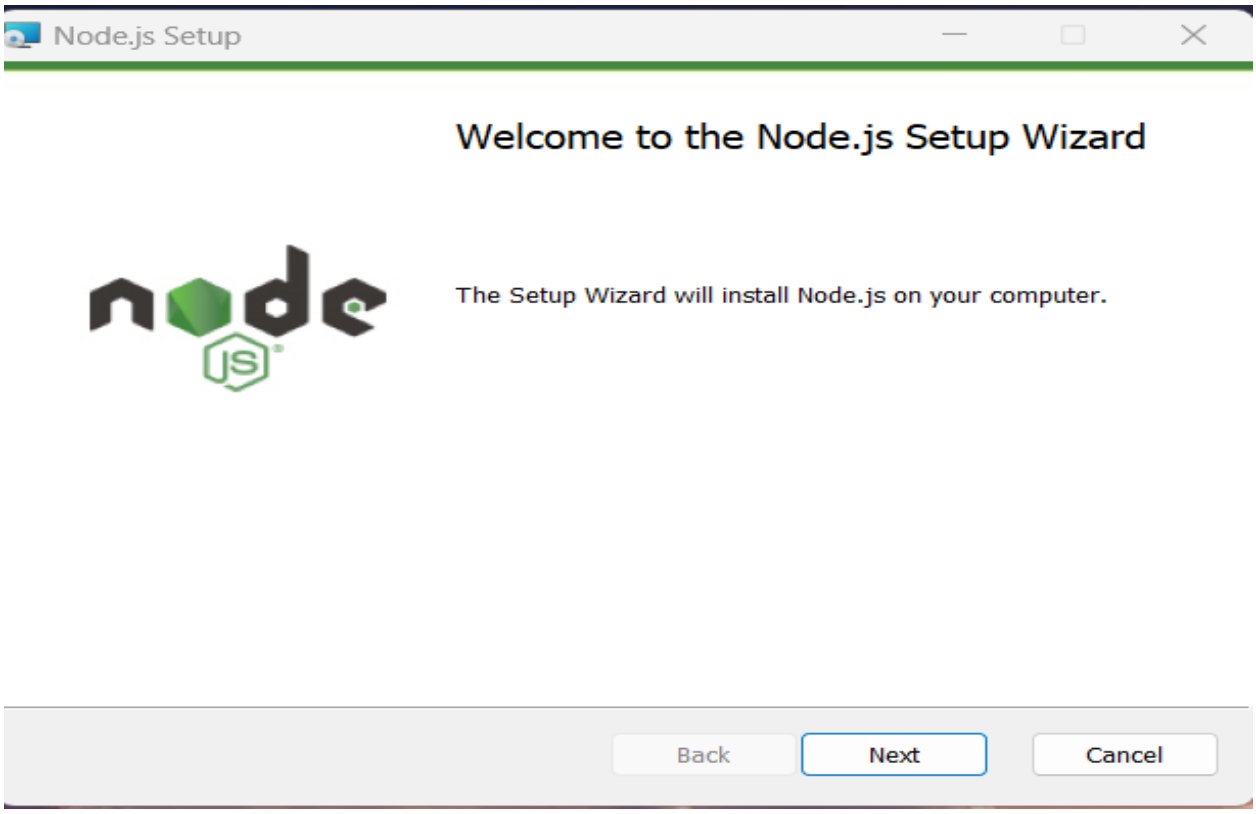
When Node.js performs an I/O operation, like reading from the network, accessing a database or the filesystem, instead of blocking the thread and wasting CPU cycles waiting, Node.js will resume the operations when the response comes back.

This allows Node.js to handle thousands of concurrent connections with a single server without introducing the burden of managing thread concurrency, which could be a significant source of bugs.

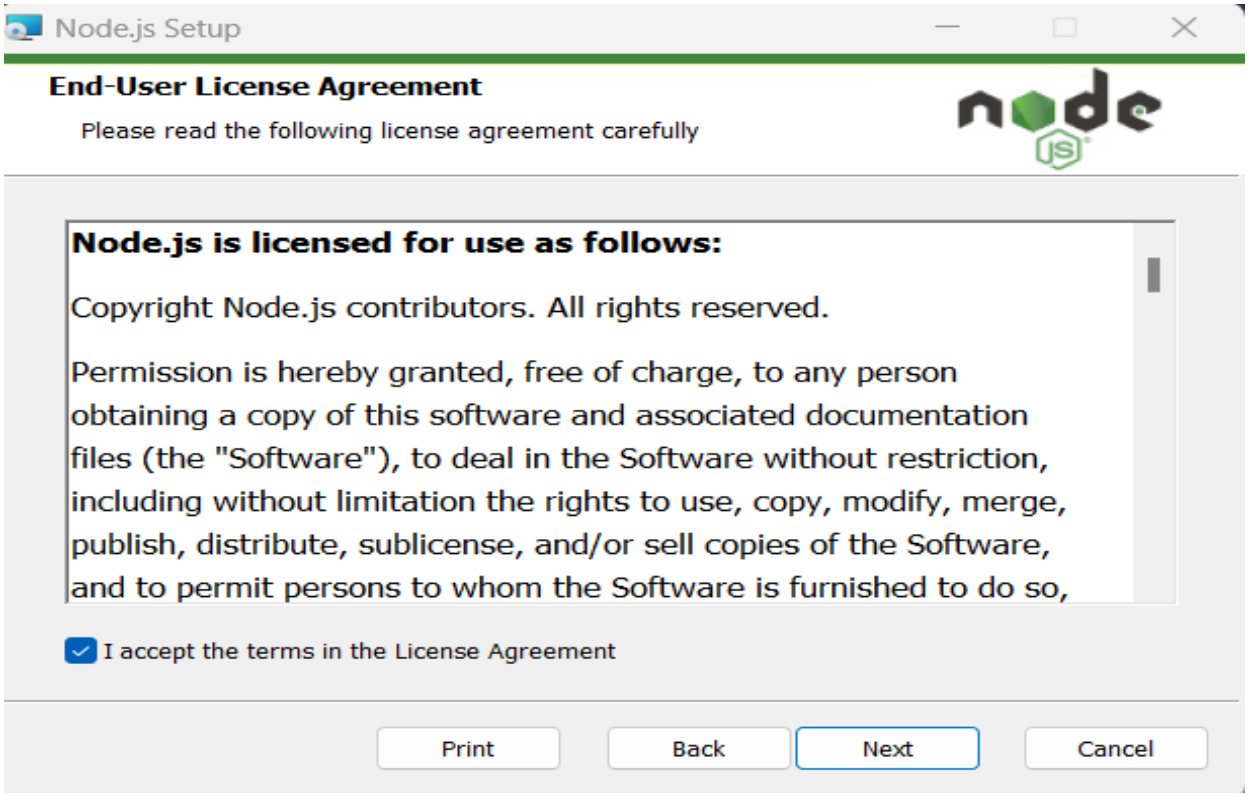
Download and Install NodeJS in System:

For download NodeJS you can visit the official NodeJS webpage https://nodejs.org/en/download.

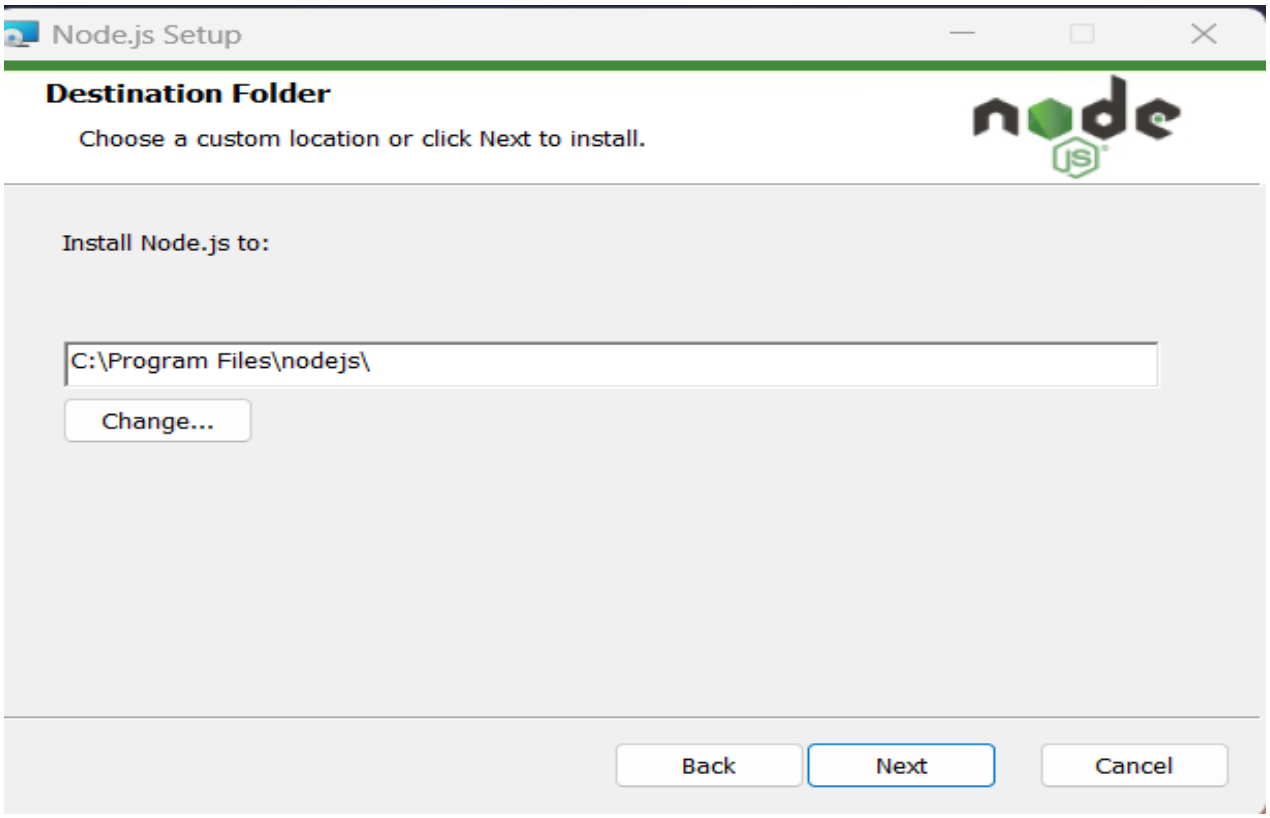


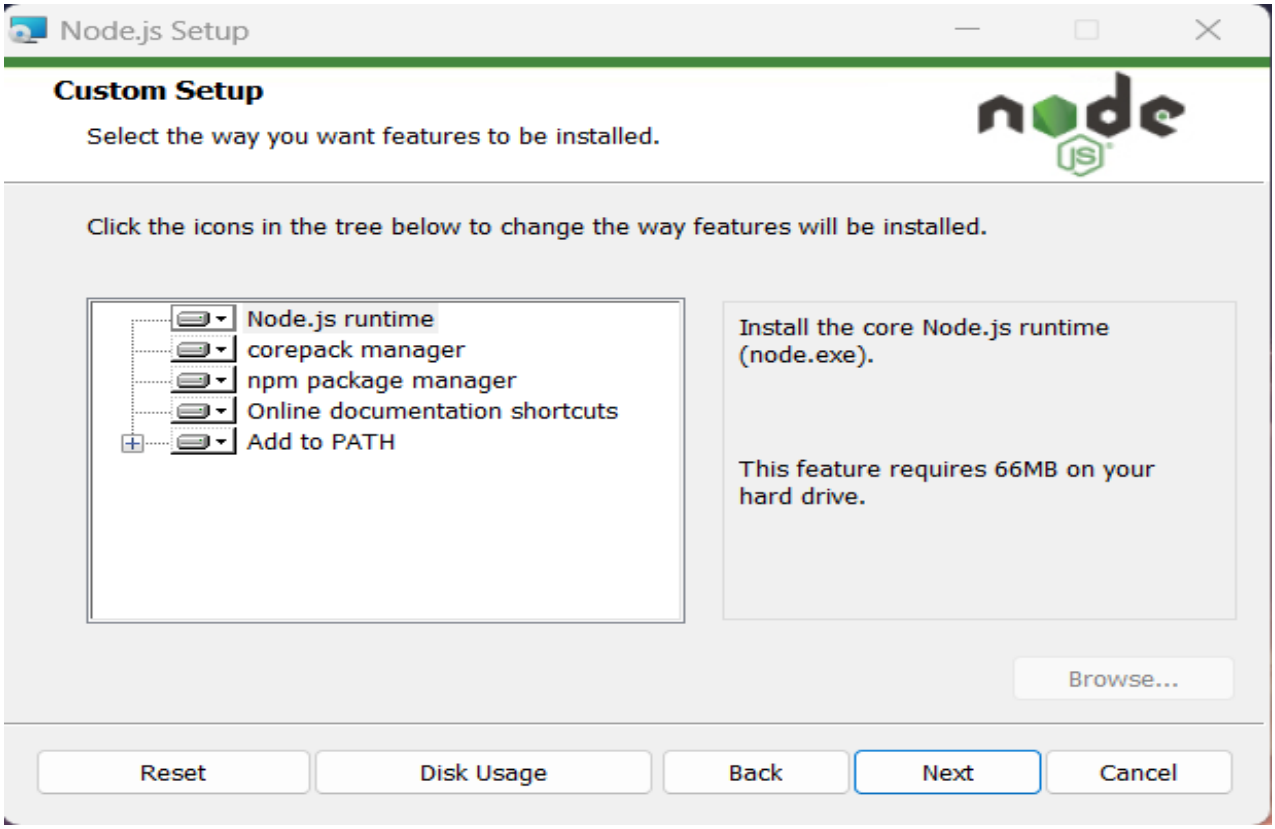


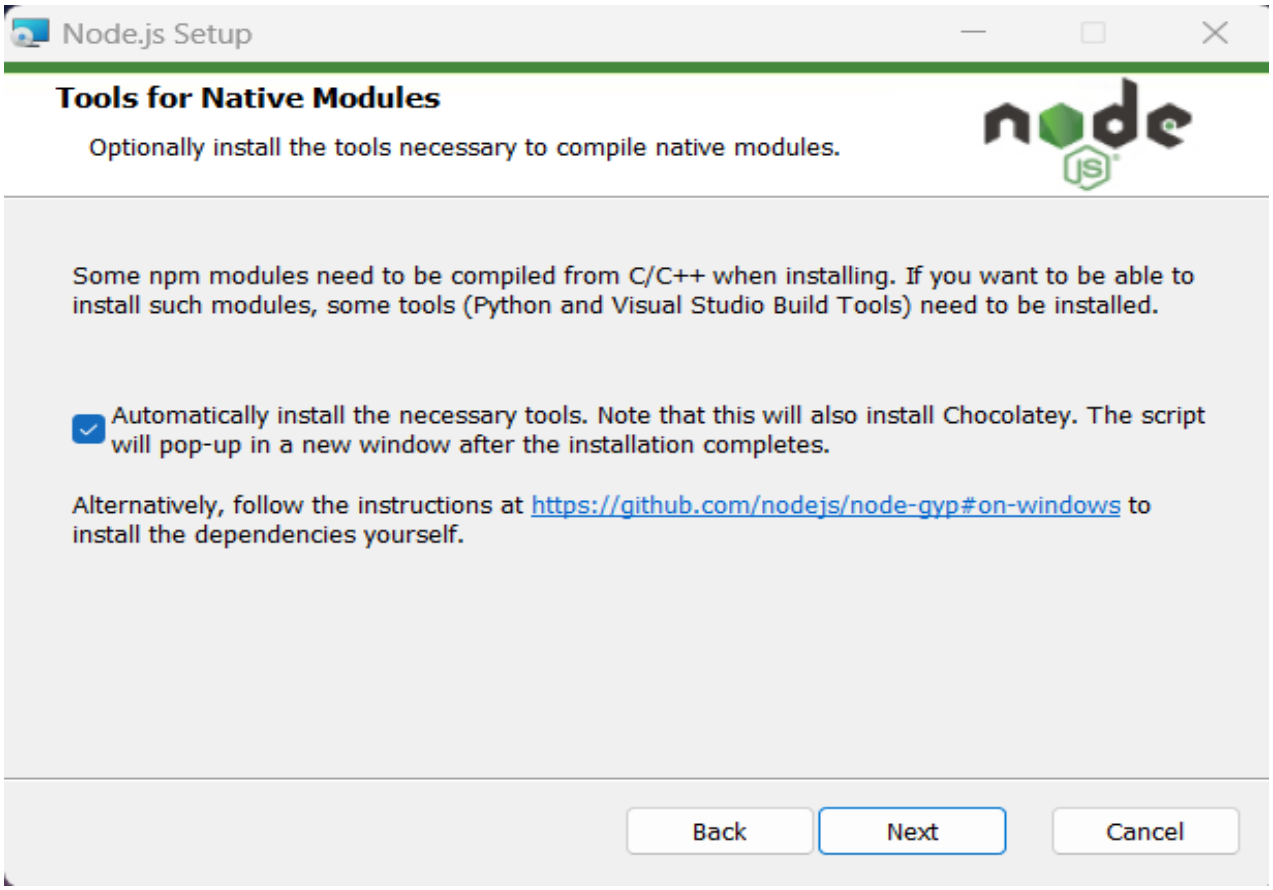
Step1: Click on Next button.



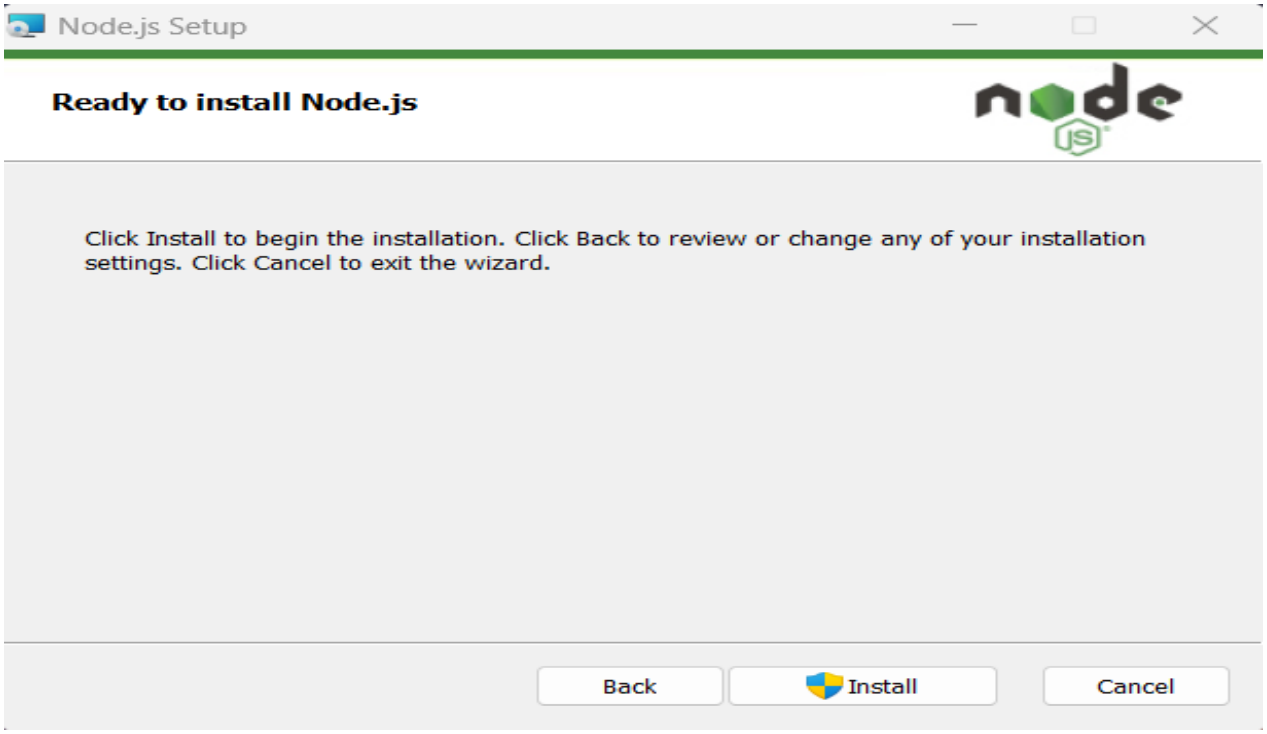
Step2: Accept the terms and condition or license agreement.

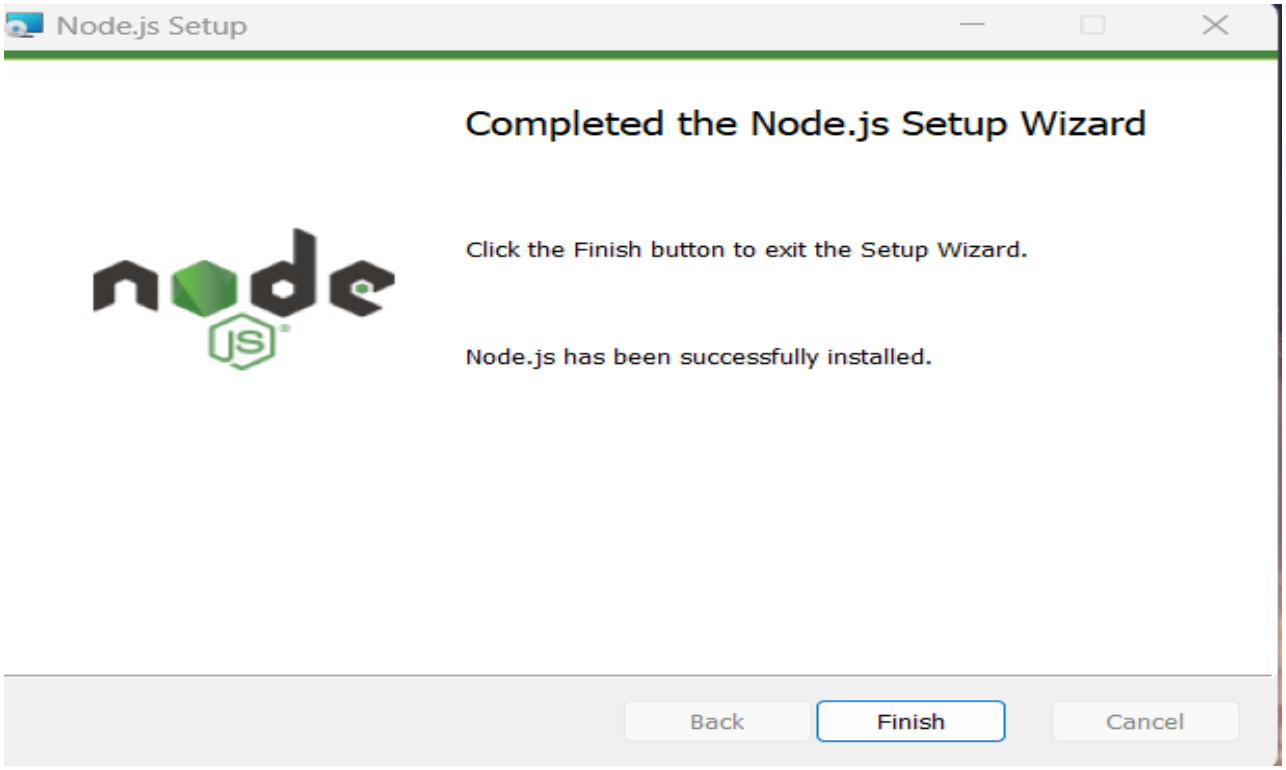






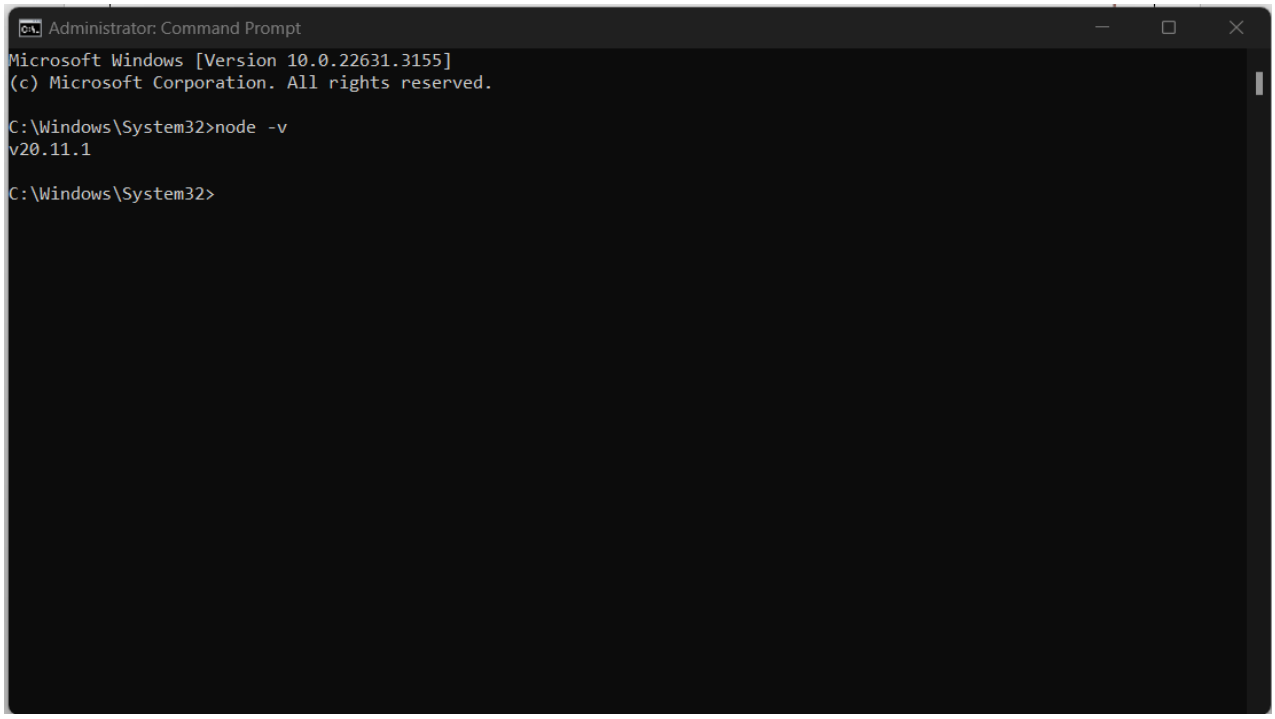
Step3: Click on automatically install necessary tools, and click on next.



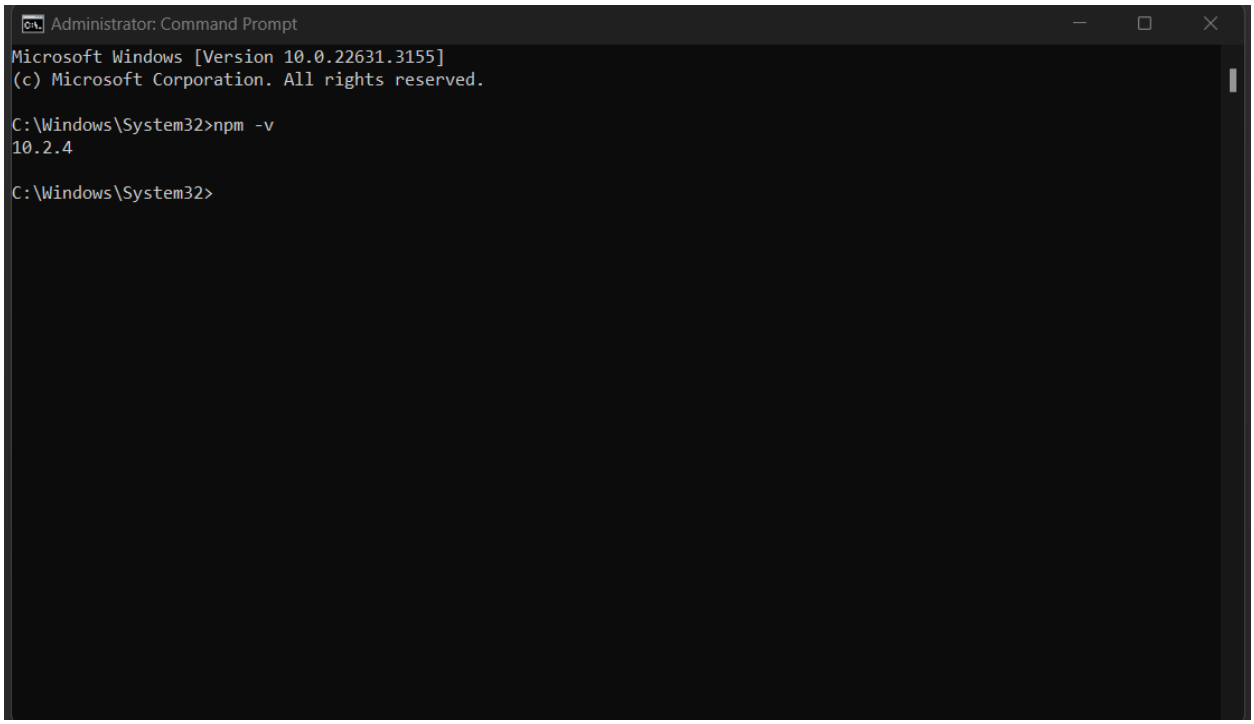


Step4: Click on finish button.

To check the Node is successfully install or not or check the version open CMD or PowerShell.



To verify the npm version type the following command in cmd.



For writing the code we use Visual Studio Code editor, it is free and more suitable for coding.

Download and Install Visual Studio Code:

6. Introduction:

Visual Studio Code is a free source-code editor made by Microsoft for Linux, Windows, and mac OS. VS Code has support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, add keyboard shortcuts, edit preferences, and install extensions to add functionality.

The most important tool for developers is that which let us author code: the development environment. Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and

embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

In 2024**, Visual Studio Code (VS Code)** continues to hold its position as a preeminent tool in the developer's toolkit.

According to Stack Overflow's Developer Survey, an impressive 70% of professional developers choose it as their primary code editor12. Its versatility, extensive extension ecosystem, and user-friendly interface contribute to its popularity among developers worldwide.

Whether you're writing code, debugging, or collaborating with others, VS Code remains a top choice for many developers. Its features, such as IntelliSense, Git integration, and customizable settings, make it a powerful and efficient tool for software development. So, if you're looking for a reliable and widely adopted code editor, VS Code is definitely worth exploring.

6.1.1 This is some reasons why we use Visual Studio Code:

- VS Code is simple to download and install on any OS.
- It is completely free.

- It is non-opinionated and open source, with non-proprietary standards.
- You can bring your favourite key bindings from Sublime, Vim, etc. with you.
- It's easy to customize and has many useful extensions.
- It has powerful IDE-like features including a built-in debugger and terminal.

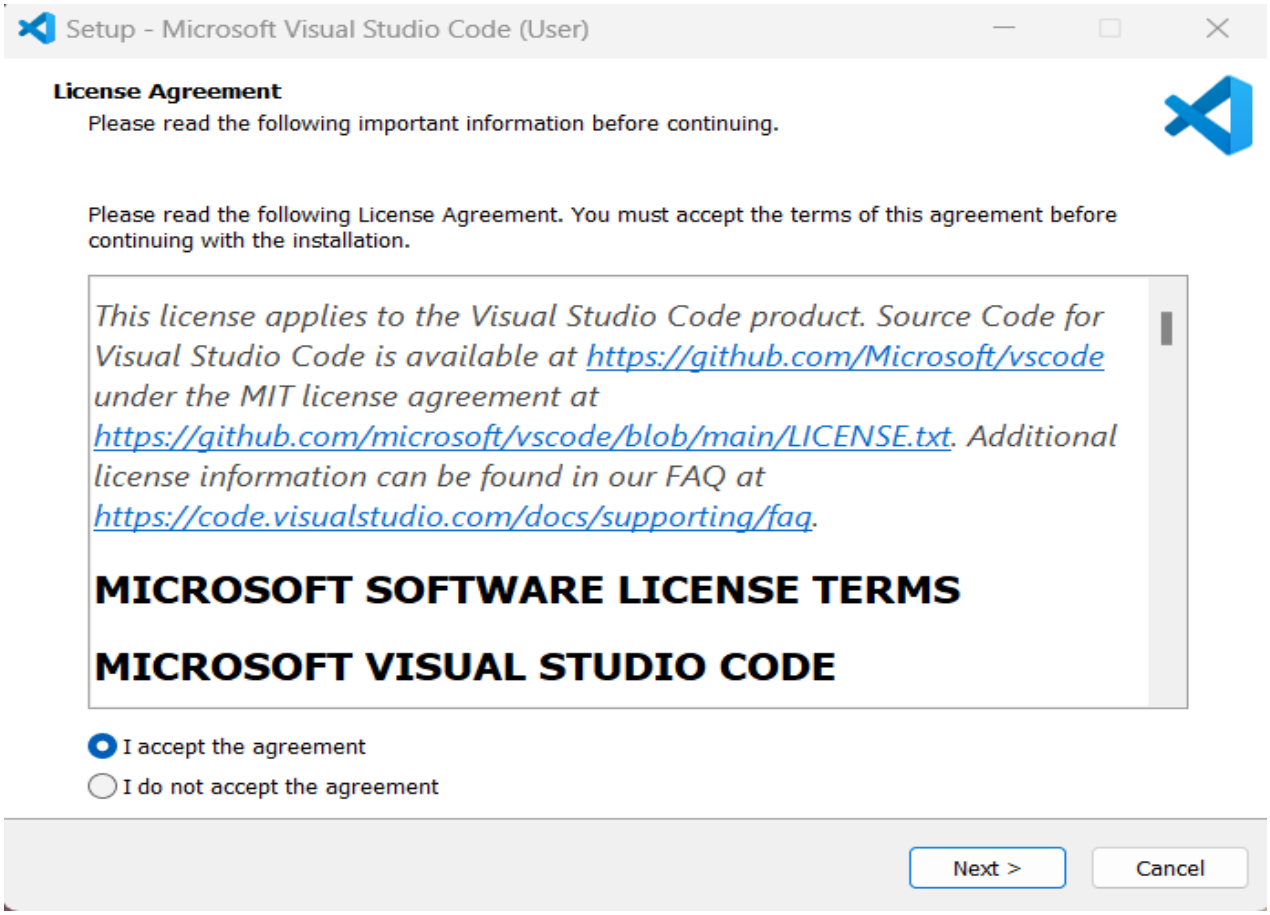
6.1.2 Setup Visual Sudio Code:

- ✓ Download and install visual studio code.
- ✓ Create a new file.
- ✓ See an overview of the user interface.
- ✓ Install support for your favorite programming language.
- ✓ Change your keyboard shortcuts and easily migrate from other editors using keymap extensions.
- ✓ Customize your editor with themes.

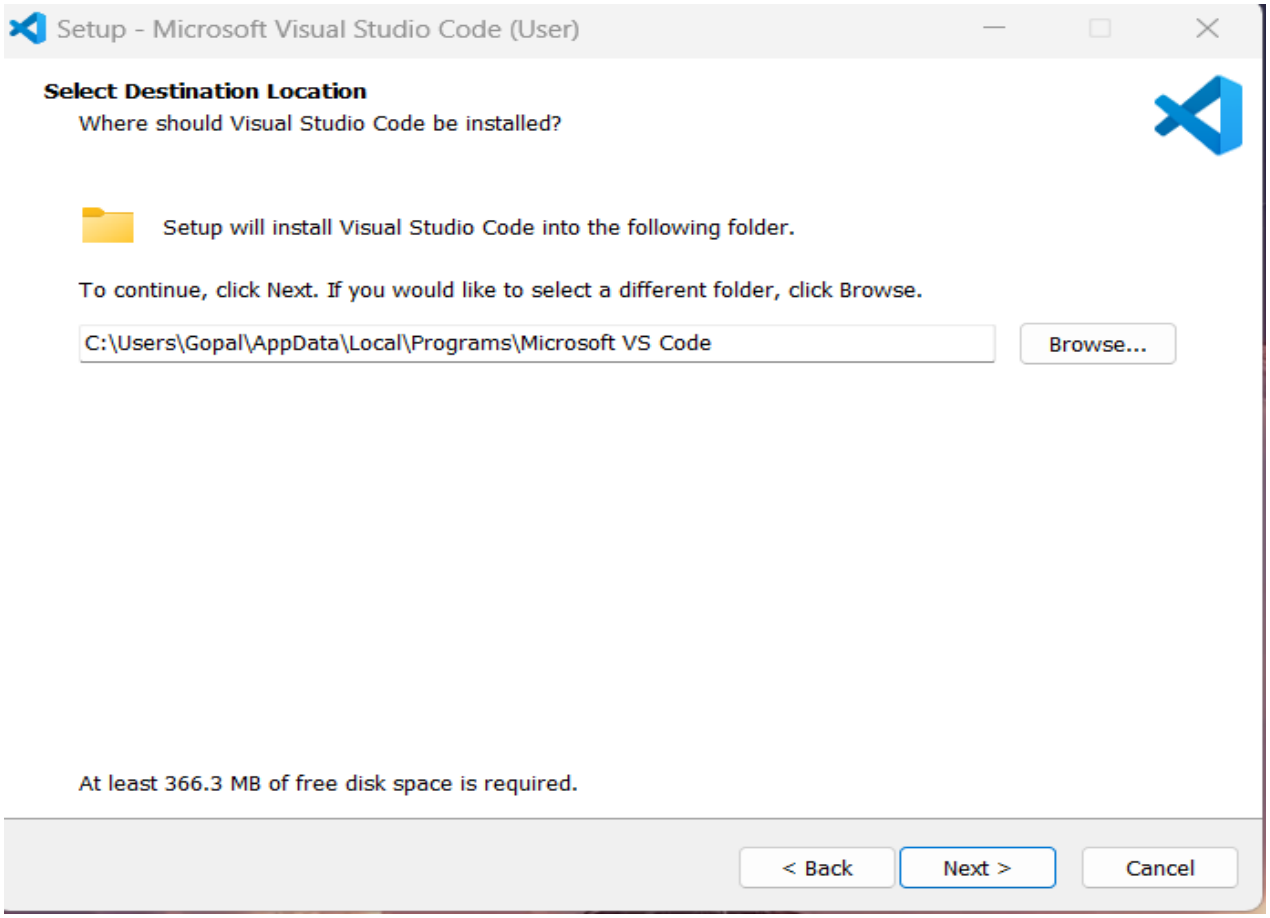
6.1.3 Install Visual Studio Code:

Step1: Download the VS code from its official site.

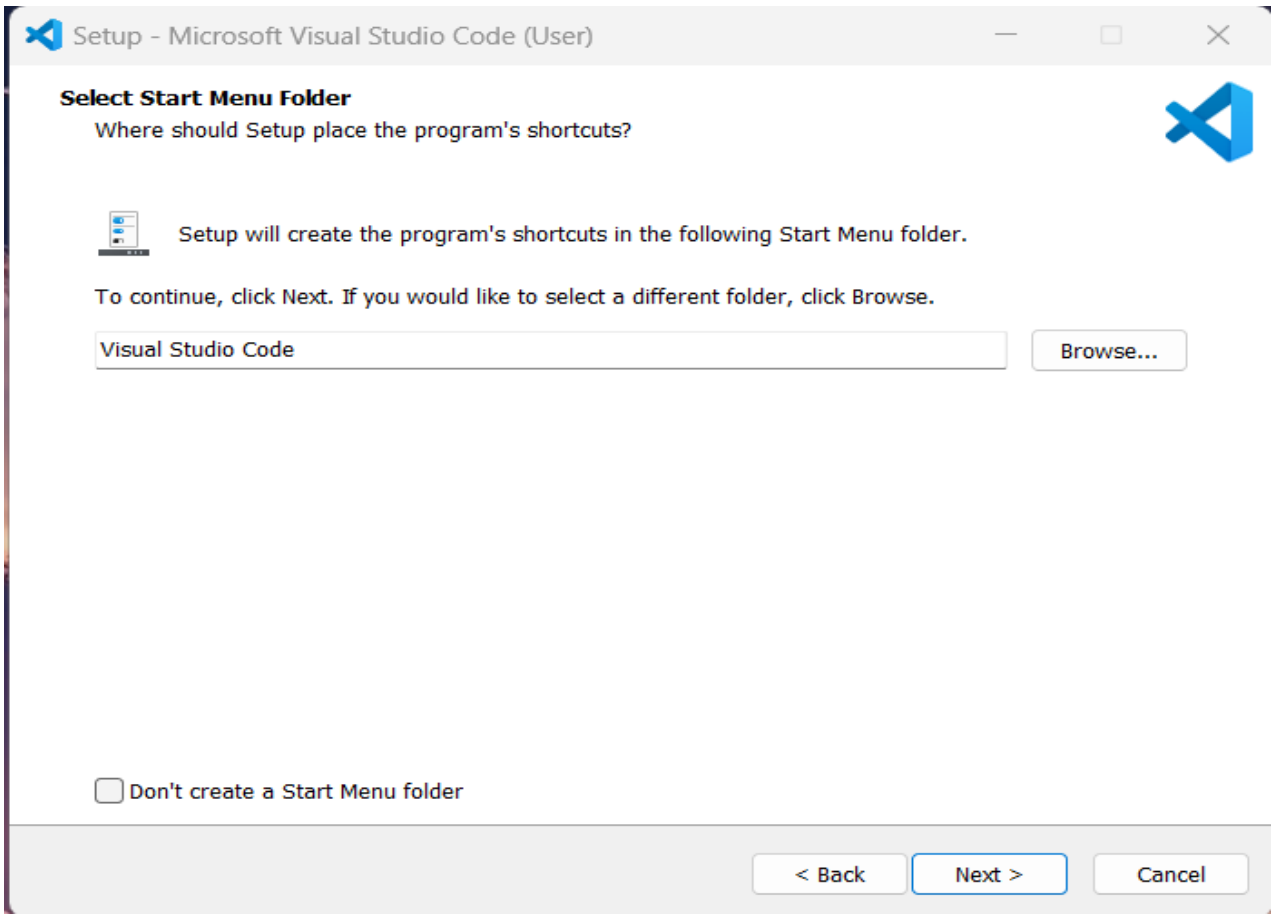
Step2: Download the Visual Studio Code installer for Windows. Once it is downloaded, run the installer (VSCodeUserSetup-{version}.exe). Then, run the file – it will only take a minute.



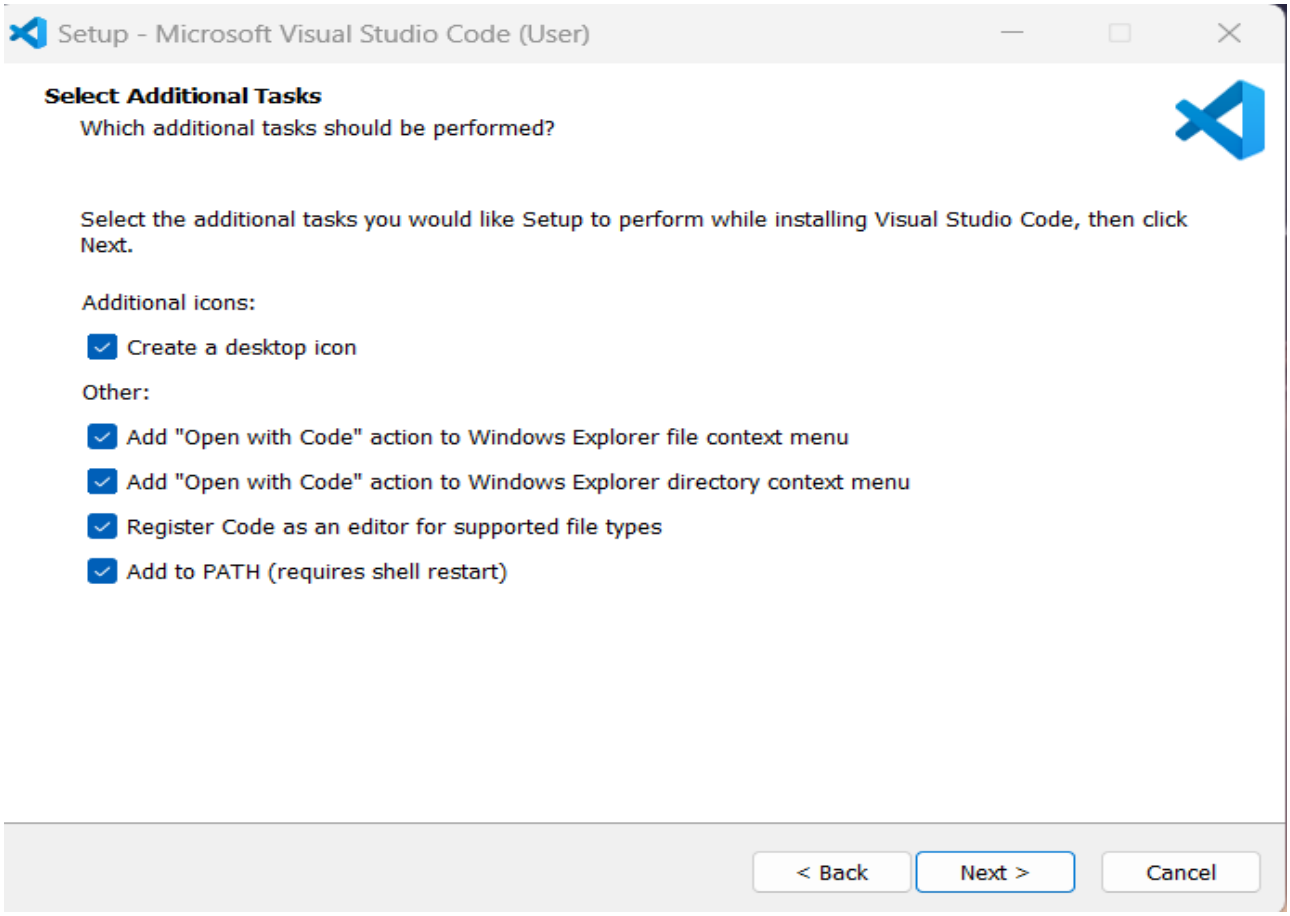
- Accept the agreement and click on Next.



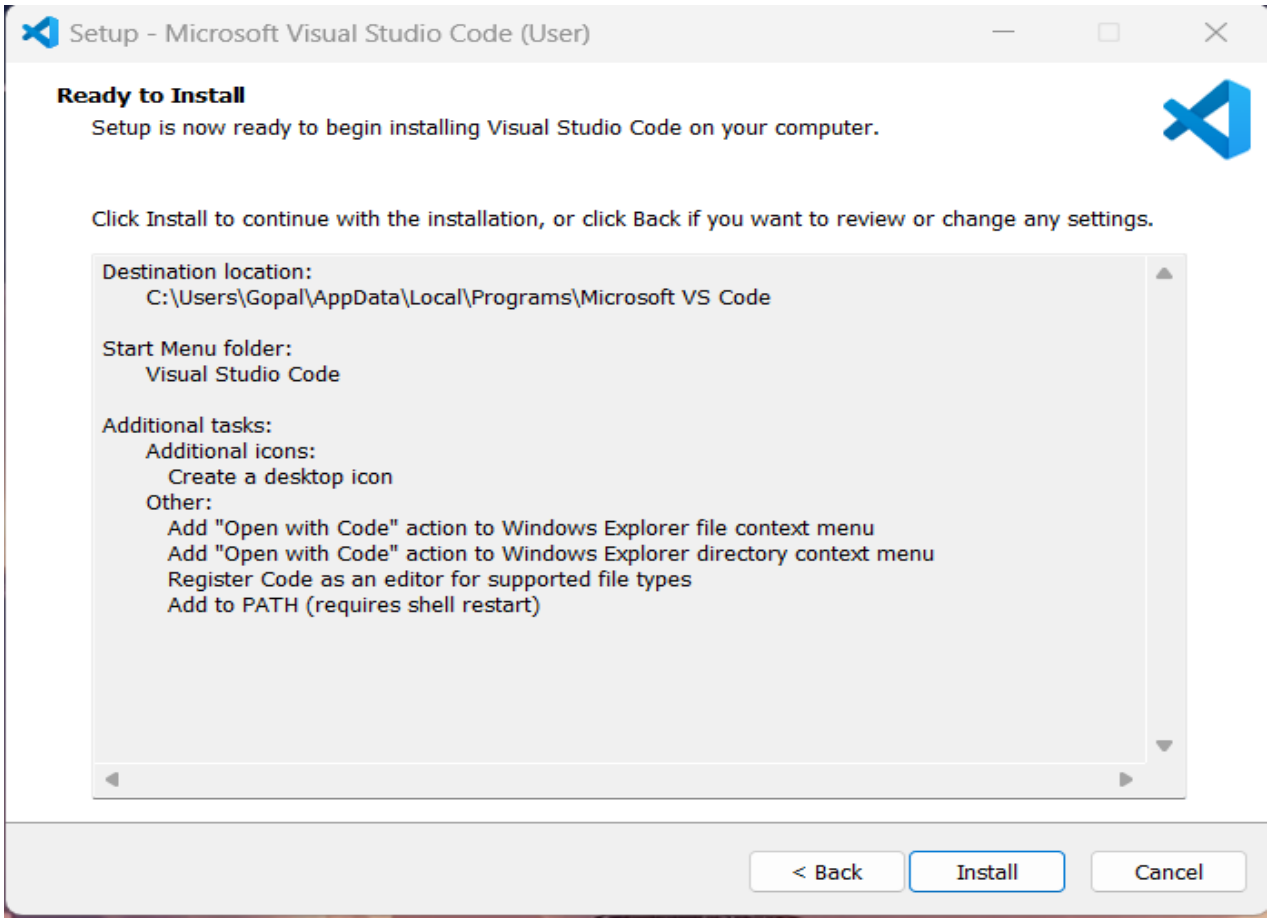
- Choose the path where you want to install and click on Next.



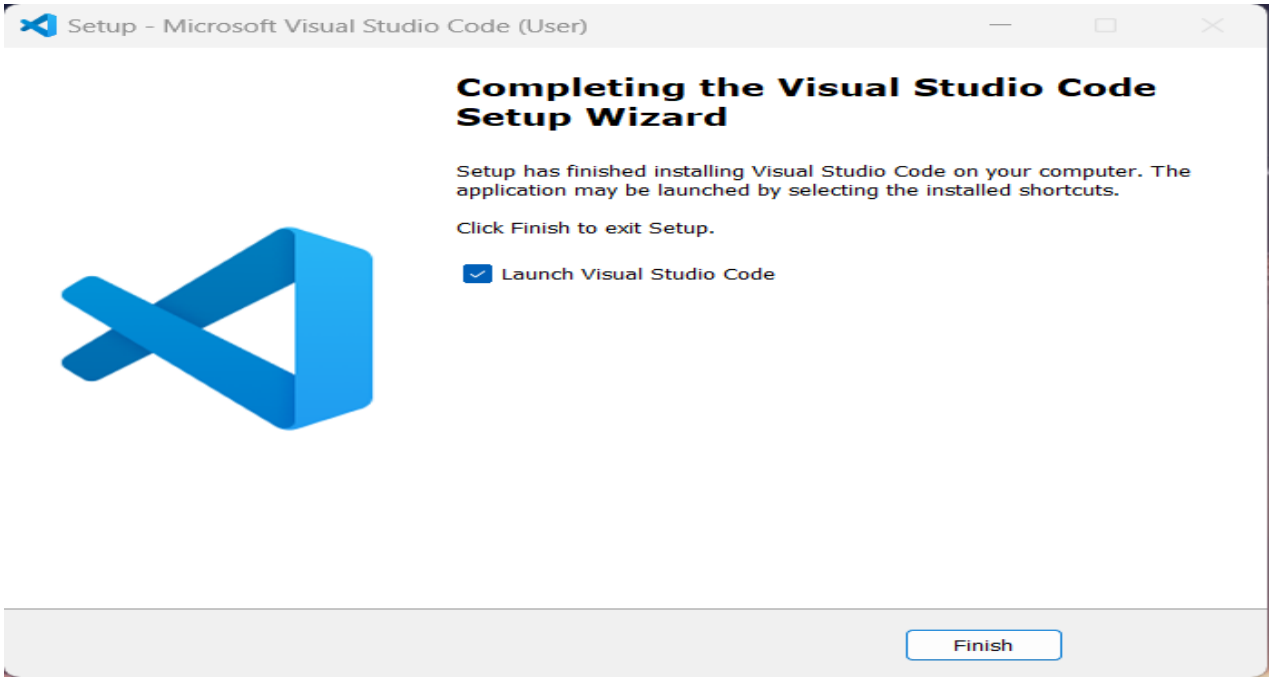
- Just click on Next.



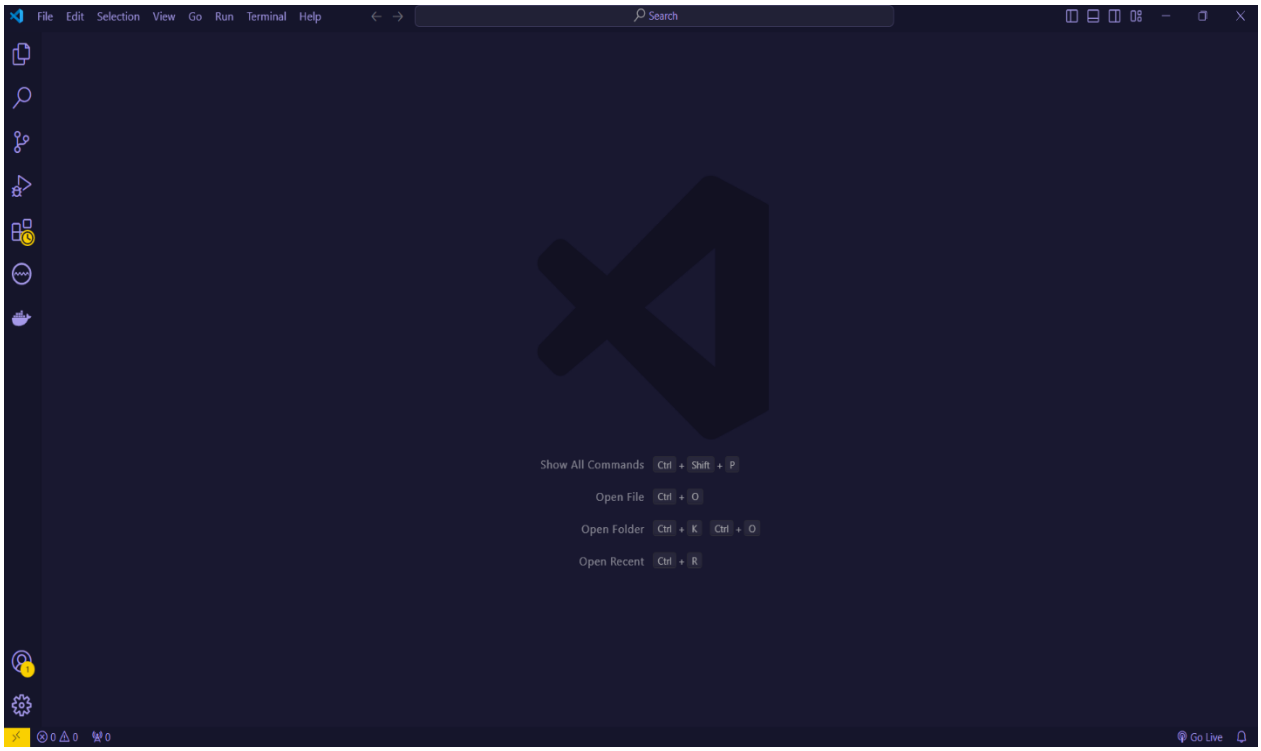
- Select all the checkbox and click on Next.



- Click on Install.



- Click on finish.



- If the installation is successful, you will see the particular screen.

Python Extension for Visual Studio Code:

A plugin for Visual Studio Code that offers extensive support for the Python language (for all currently maintained versions of the language: >=3.7), along with capabilities like variable explorer, test explorer, refactoring, IntelliSense (Pylance), linting, debugging, code navigation, and formatting!

Step 1: Install a supported version of Python on your machine (notice that macOS does not support system installations of Python).

Step 2: Install Visual Studio Code's Python plugin.

Step 3: Launch a Python file or create one, then begin coding!

Flask framework:

Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features.

It does have many cool features like url routing, template engine. It is a WSGI web app framework.

What is a Web Framework ? :

A Web Application Framework or a simply a Web Framework represents a collection of libraries and modules that enable web application developers to write applications without worrying about low-level details such as protocol, thread management, and so on.

Steps to create a flask project:

- Install flask (preferably via pip).
- Design a flask minimal app.
- Add end points for each action.
- Use https verbs (get and post) accordingly.
-

CASCADING STYLE SHEETS

In our project we are using CSS for styling purposes, compare between CSS and SCSS provide bellow:

What is CSS:

Cascading Style Sheets is what CSS stands for. It is a style sheet language used to specify how a markup language page should format and appear. It gives HTML an additional feature. To alter the style of web pages and user interfaces, it is typically used in conjunction with HTML.

Any type of XML document, including SVG, XUL, and plain XML, can be used with it as well.

CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.
Here we used CSS for styling the webpages when doing the project development.

Solves a Big Problem:

Font, color, background style, element alignments, border, and size tags had to be duplicated on each page of a website before CSS. This was a protracted procedure. For instance, it will become a time-consuming and costly process to construct a large website where fonts and color information are updated on each and every page. To address this issue, CSS was developed. It was a W3C suggestion.

Saves Time:

CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

What is SCSS:

The term SCSS is an acronym for Sassy Cascading Style Sheets. It is basically a more advanced and evolved variant of the CSS language. Natalie Weizenbaum and Chris Eppstein created it, and Hampton Catlin designed it. It comes with more advanced features- thus often called Sassy CSS.

Here are some points how SCSS enhances CSS:

Variables: CSS forces you to repeat values like colors or fonts throughout your stylesheet. SCSS lets you define variables to store these values, making your code cleaner and easier to maintain.

Nesting: Complex CSS can get hard to read with many selectors stacked on top of each other. SCSS allows you to nest selectors within others, mimicking the structure of your HTML and creating a more organized stylesheet.

Mixins: SCSS lets you create reusable code blocks called mixins. These act like functions in programming, allowing you to define a set of styles and then apply them to different elements with a single line of code. This saves time and reduces redundancy.

While SCSS isn't a general-purpose programming language, it offers these programming-like features that make CSS development more efficient and maintainable, especially for large projects.

Difference Between CSS and SCSS:

	CSS	SCSS
Definition	**Cascading Style Sheet** is a stylesheet language used to define the presentation and layout of a webpage written in HTML or XML.	**Syntactically Awesome Style Sheet** SCSS (Sassy CSS) is a preprocessor scripting language that is a superset of CSS.
Syntax	CSS follows a plain-text syntax	SCSS follows a structured syntax with additional features such as variables, nesting, and mixins.
Variables	CSS uses plain text and keywords for styling	SCSS allows you to define variables to store commonly used values such as colors, font sizes, and spacing
Nesting	CSS requires you to write each selector separately.	SCSS allows you to nest selectors within other selectors, making it easier to write and read complex stylesheets,

	CSS	SCSS
Mixins	CSS does not provide functionality to use Mixins	SCSS allows you to create reusable code snippets using mixins, which are like functions in programming languages.
File Extension	CSS files use the .css file extension	SCSS files use the .scss file extension.
Compilation	CSS files are interpreted by web browsers directly	SCSS files must be preprocessed into standard CSS files using a preprocessor such as Sass
Features	CSS Does not use additional features	SCSS contains all the features of CSS and contains more features that are not present in CSS

SYSTEM ANALYSIS AND DESIGN

Requirement Specification:

SOFTWARE REQUIREMENTS AND SPECIFICATIONS (SRS)

Introduction:

The Movie Streaming and Recommendation System is a user-friendly platform designed to allow users to stream movies and receive personalized movie recommendations. The purpose of this system is to provide a seamless movie streaming experience and offer recommendations based on the user's viewing history and preferences.

The system features include user registration and authentication, where users can register and create an account using their email. The system authenticates users during login to ensure security. The core feature of the system is movie streaming, where users can browse and stream movies from a vast library. The system supports various video formats and ensures smooth streaming for an optimal user experience.

Purpose:

The purpose of the Movie Streaming and Recommendation System is to provide a platform that enhances the movie-watching experience for users. It aims to offer a seamless interface where users can not only stream their favorite movies but also discover new ones through personalized recommendations. The system is designed to understand the user's viewing history and preferences, and based on this data, it suggests movies that the user might enjoy.

Scope:

The scope of the Movie Streaming and Recommendation System extends beyond just being a platform for streaming movies. It is designed to offer a comprehensive user experience where users can browse through a vast library of movies and stream their chosen content. But what sets this system apart is its ability to provide personalized movie recommendations.

Systems Features:

User Registrations and Authentication:

The User Registration and Authentication feature is a crucial part of the Movie Streaming and Recommendation System. It allows users to create a unique account using their email

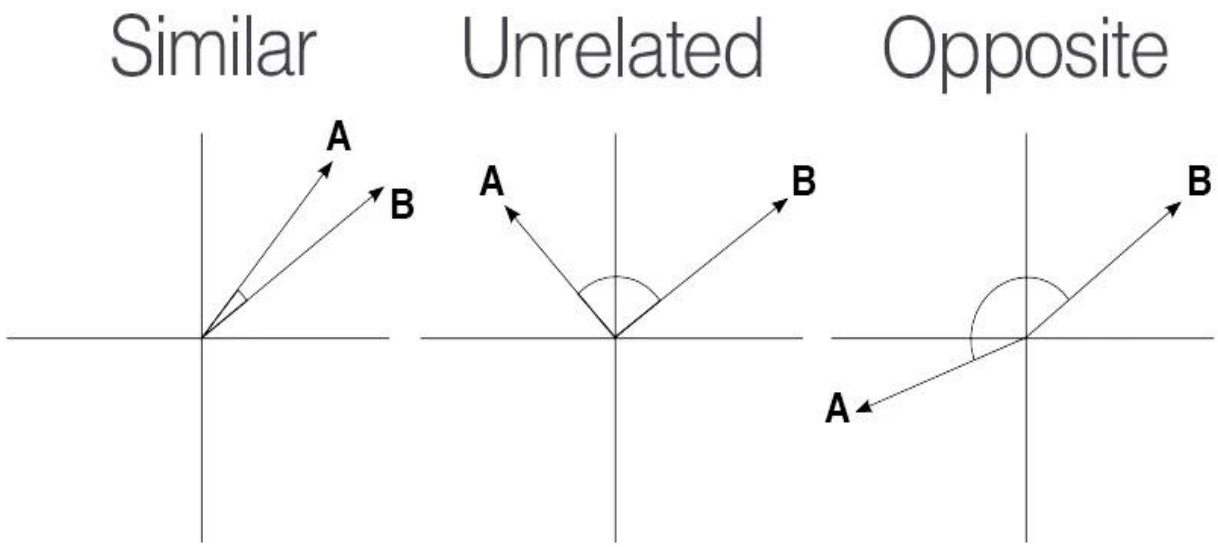
address. This process involves the user providing their email and setting up a password, which forms the basis of their account. Once the account is created, the user can log in to the system using these credentials.

WHAT IS COSINE SIMILARITY AND COSINE DISTANCE

Cosine similarity is a metric that measures the cosine of the angle between two vectors projected in a multi-dimensional space.

Suppose the angle between the two vectors is 90 degrees, the cosine similarity will have a value of 0; this means that the two vectors are perpendicular to each other which means they have no correlation between them.

As the cosine similarity measurement gets closer to 1, then the angle between the two vectors A and B becomes smaller. In this case, A and B are more similar to each other.



The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the euclidean norms or magnitude of each vector.

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

$$\|\vec{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \cdots + a_n^2}$$

$$\|\vec{b}\| = \sqrt{b_1^2 + b_2^2 + b_3^2 + \cdots + b_n^2}$$

Where, a and b are vectors in a multidimensional space.

Since the $cos(\theta)$ value is in the range $[-1,1]$:

- -1 value will indicate strongly opposite vectors i.e. no similarity
- 0 indicates independent (or orthogonal) vectors.
- 1 indicates a high similarity between the vectors.

Cosine Distance:

Cosine distance is a metric used in machine learning to determine the dissimilarity between two vectors. It is derived from the cosine similarity, which measures the cosine of the angle between two vectors. The cosine similarity is a measure of how similar the two vectors (or data objects) are, irrespective of their size.

The formula to find the cosine similarity between two vectors x and y is:

cosine similarity(x,y) = ||x||·||y||x·y

where:

x . y is the dot product of the vectors x and y,

||x|| and ||y|| are the lengths (or magnitudes) of the vectors x and y.

The cosine distance is then calculated as:

cosine distance = 1−cosine similarity

This formula ensures that cosine distance values range from 0 (perfect similarity) to 2 (perfect dissimilarity). A value of 1 indicates that the vectors are orthogonal or unrelated.

Usually, people use the cosine similarity as a similarity metric between vectors. Now, the cosine distance can be defined as follows:

Cosine Distance = 1 — Cosine Similarity

The intuition behind this is that if 2 vectors are perfectly the same then the **similarity** is **1** (angle=0 hence $cos(\theta)$=1) and thus, **distance** is **0** (1–1=0).

Applications of cosine similarity:

1. The metric is used in processes of data mining, information retrieval, and text matching.

2. Used in a recommendation engine to recommend similar products/movies/shows/books.

3. In Information retrieval, using weighted TF-IDF and cosine similarity is a very common technique to quickly retrieve documents similar to a search query.

4. The cosine-similarity based locality-sensitive hashing technique increases the speed for matching DNA sequence data.

Why is cosine similarity a popular metric?

There are various distance measures that are used as a metric for the evaluation of data points. Some of them are as follows.

- Euclidean distance
- Manhattan distance
- Minkowski distance
- Hamming distance and many more.

Out of all these widely used metrics for calculating distance, Hamming distance can be utilized as a measure for KNN, recommendation systems, and textual data instead of cosine similarity when used for classification or text data. However, cosine similarity can handle data with varying lengths, whereas hamming distance only takes into account character-type data of the same length. When analyzing textual data, cosine similarity helps to produce higher similarity scores for the text data by taking into account the words that appear in the document frequently, whereas Hamming distance ignores these words and produces a lower similarity index from the text document.

Use of cosine similarity in machine learning:

In machine learning, cosine similarity can be used for classification tasks. It can be used as a metric in KNN classification algorithms to determine the ideal number of neighbors. Additionally, the fitted KNN model can be assessed against other KNN classification algorithms. Finally, the KNN classifier, which has been fitted with cosine similarity as a metric, can be used to assess multiple performance metrics, such as accuracy score and AUC score. The classification report can also be obtained to assess additional parameters, such as precision and recall.

Let us see how to use cosine similarity as a metric in machine learning

knn_model = KNeighborsClassifier(metric='cosine')

The above model can be fitted against the split data and can be used to obtain prediction values that can be used for various other parameters.

In order to determine the ideal number of neighbors, cosine similarity can be utilized in machine learning. In this method, data points with higher similarity will be regarded as the closest neighbors, while those with lesser similarity will not be taken into account. Thus, this is the machine learning application of cosine similarity.

Use of cosine similarity in recommendation systems:

Machine learning recommendation systems are an example of an algorithm that operates on content similarity. Recommendation systems essentially employ the similarity matrix to suggest comparable information to the user based on his accessing characteristics. There are several techniques to measure the similarity between the two items.

In order to extract the necessary attributes from the recommendation data that would be helpful for recommending the contents, any recommendation data can be obtained. The necessary textual data must be vectorized using the CountVectorizer after it is available in order to produce the similarity matrix. Therefore, the user can be recommended using Scikit-Learn's cosine similarity metrics once the similarity matrix has been created.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
count_vec=CountVectorizer()
sim_matrix=count_vec.fit_transform(df['text_data'])
print('Similarity Matrix',sim_matrix.toarray())
cos_sim = cosine_similarity(sim_matrix)
```

For the purpose of recommendation, the cosine similarity would produce a similarity matrix for the chosen textual data, and lists might be used to order the content with greater similarity scores. In this case, terms that appear frequently in the textual data would be taken into account by cosine similarity, which would vectorize those terms with greater frequencies and recommend material with higher recommendation percentages. Thus, cosine similarity is employed in recommendation systems in this way.

Use of cosine similarity with textual data:

When comparing the similarity of two text documents or tokenized texts, cosine similarity in textual data is utilized. Therefore, in order to apply cosine similarity in text data, the text data must first be tokenized. Then, a similarity matrix must be constructed from the tokenized text data, which can then be given on to the cosine similarity metrics in order to assess the text document's similarity.

```
from sklearn.feature_extraction.text
import CountVectorizer
count_vectorizer = CountVectorizer()
sim_matrix =
count_vectorizer.fit_transform(tokenized_
data)
sim_matrix
from sklearn.metrics.pairwise import
cosine_similarity
cos_sim_matrix =
cosine_similarity(sim_matrix)
create_dataframe(cos_sim_matrix,tokenized
_data[1:3]) ## using the first two
tokenized data.
```

So the above code can be used to measure the similarity between the tokenized document and here the first two tokenized documents from the corpus is used to evaluate the similarity between them and the output generated will be as shown below.

$$\cos\theta = \frac{\vec{a}\cdot\vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

$$\|\vec{a}\| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \cdots + a_n^2}$$

$$\|\vec{b}\| = \sqrt{b_1^2 + b_2^2 + b_3^2 + \cdots + b_n^2}$$

Now let us try to interpret the sample output that will be produced by the cosine similarity metrics. So here cosine similarity would consider the frequently occurring words between the two tokens and it has yielded a 50% similarity between the first and the second token in the corpus. So this is how cosine similarity is used in the textual data.

Advantages :

The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.

When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.

What Is a Good Cosine Similarity Score?

A cosine similarity is a value that is bound by a constrained range of 0 and 1. The closer the value is to 0 means that the two vectors are orthogonal or perpendicular to each other. When the value is closer to one, it means the angle is smaller and the images are more similar.

Summary:

Among the various metrics, cosine similarity is majorly used in various tasks of machine learning and in handling textual data because of its dynamic ability to adapt to various characteristics of data. Cosine similarity entirely operates on the cosine angle properties and it is vastly used in recommendation systems as it will help us recommend content to the user according to his most viewed content and characteristics and is also majorly used in finding the similarity between text documents as it considers the frequently occurring terms. This made cosine similarity a popular metric for evaluation in various applications.

WHAT IS MONGODB:

MongoDB is a source-available, cross-platform, document-oriented database program. It's a NoSQL database, which means it doesn't rely on the traditional table-like relational database structure. Instead, it uses a flexible document model, making it easy to work with dynamic and unstructured data.

MongoDB was developed by MongoDB Inc. and initially released in February 2009. It provides official driver support for popular languages like C, C++, C#, .Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, Mongoid. This wide range of language support allows developers to create applications using their preferred language.

The data in MongoDB is stored in a format called BSON (Binary representation of JSON documents). This format allows MongoDB to store and query data more efficiently. In MongoDB, databases contain collections (similar to tables in SQL), and these collections contain documents (similar to rows in SQL). Each document can have different fields (similar to columns in SQL), providing a highly flexible and scalable document structure.

One of the key features of MongoDB is its scalability. It scales horizontally using sharding, which is the process of distributing data across multiple machines. This makes MongoDB a great choice for applications that need to handle very fast queries and massive datasets.

MongoDB also ensures high availability through its replica set feature. A replica set is a group of MongoDB instances that continuously replicate data between them, offering redundancy and protection against downtime.

MongoDB is used in a variety of applications. Here are some use cases:

1. Product Data Management: MongoDB can be used to manage product data, providing a flexible schema that can easily accommodate changes in product attributes.

2. Operational Intelligence: MongoDB can be used to collect and analyze operational data in real-time, providing insights that can help improve operational efficiency.

3. Product Catalog: MongoDB's flexible document model makes it an excellent choice for managing product catalogs, which often involve complex relationships and hierarchies.

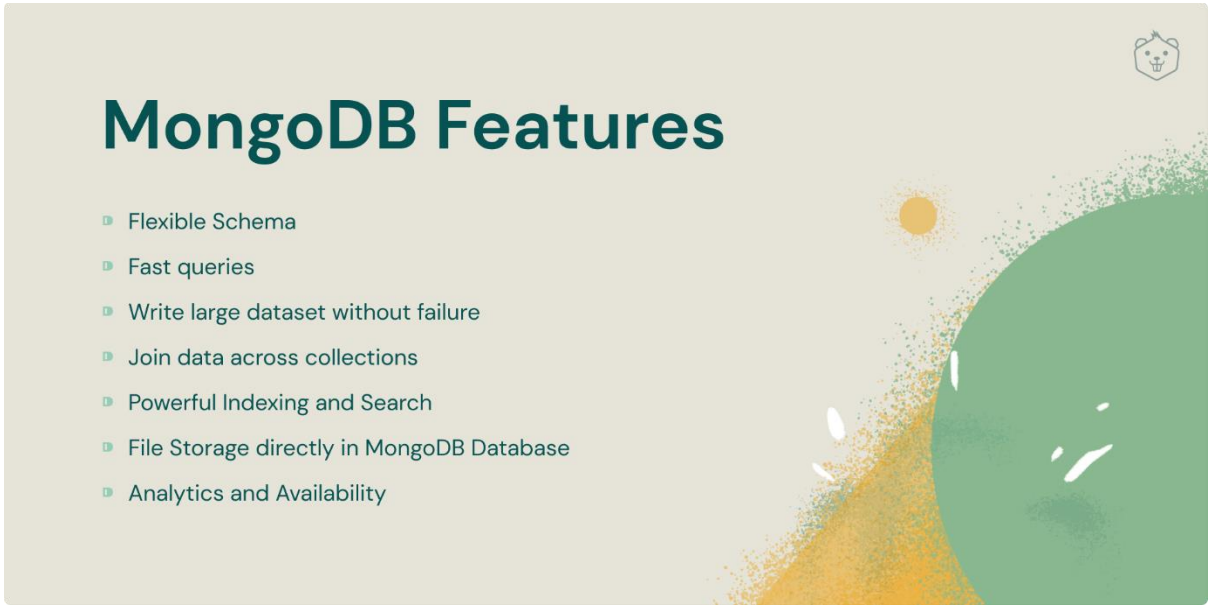
4. Scaling And Application Mobility: MongoDB's horizontal scaling capabilities make it a good choice for applications that need to handle large amounts of data and traffic.

5. Customer Analytics: MongoDB can be used to collect and analyze customer data, providing insights that can help improve customer experience and drive business growth.

6. Mainframe Offloading: MongoDB can be used to offload workloads from mainframes, helping to reduce costs and improve performance.

7. Real-Time Data Integration: MongoDB can be used to integrate data in real-time, providing up-to-date insights and enabling real-time decision making.


mongoDB®


MongoDB Features
Flexible Schema
Fast queries
Write large dataset without failure
Join data across collections
Powerful Indexing and Search
File Storage directly in MongoDB Database
Analytics and Availability

MySQL:

MySQL is an open-source Relational Database Management System (RDBMS) that is renowned for its reliability, scalability, and ease of use. It was developed by MySQL AB, which is now owned by Oracle Corporation.

MySQL stores data in a structured format using rows and columns, enabling users to create, manage, and manipulate databases. It follows the relational database model, allowing users to organize data into tables with rows and columns, facilitating efficient data storage and retrieval.

One of the reasons MySQL is a popular choice for managing relational databases is because it's open-source software, which means it's free to use and has a large community of developers contributing to its improvement. This community support ensures that MySQL is continually updated and improved, keeping it relevant in the rapidly evolving tech landscape.

MySQL is known for its stability and reliability. It has been around for a long time and has proven its robustness in various applications, from small-scale projects to large-scale websites and enterprise-level solutions.

Performance is another key strength of MySQL. It is optimized for performance, making it capable of handling high-volume transactions and large datasets efficiently. This makes MySQL an ideal choice for applications that require fast and efficient data processing, such as e-commerce platforms and financial services.

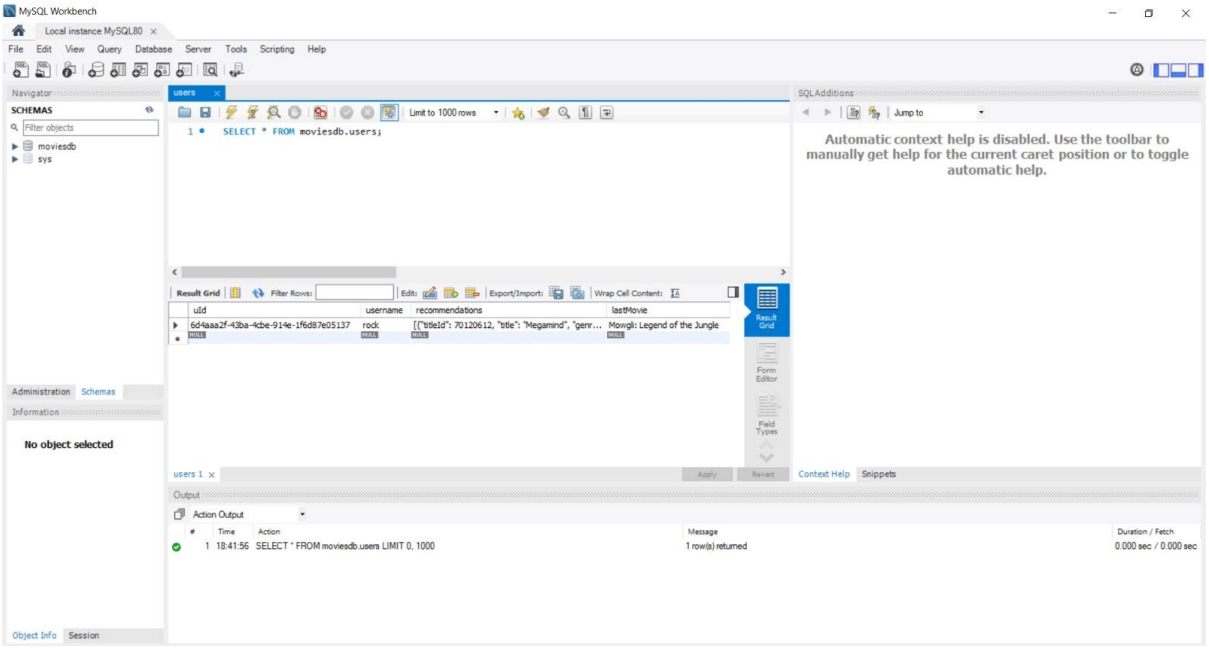
MySQL is widely supported by many programming languages, frameworks, and tools. It offers connectors and APIs for popular languages like PHP, Python, Java, and more, making it easy to integrate with your existing software stack.

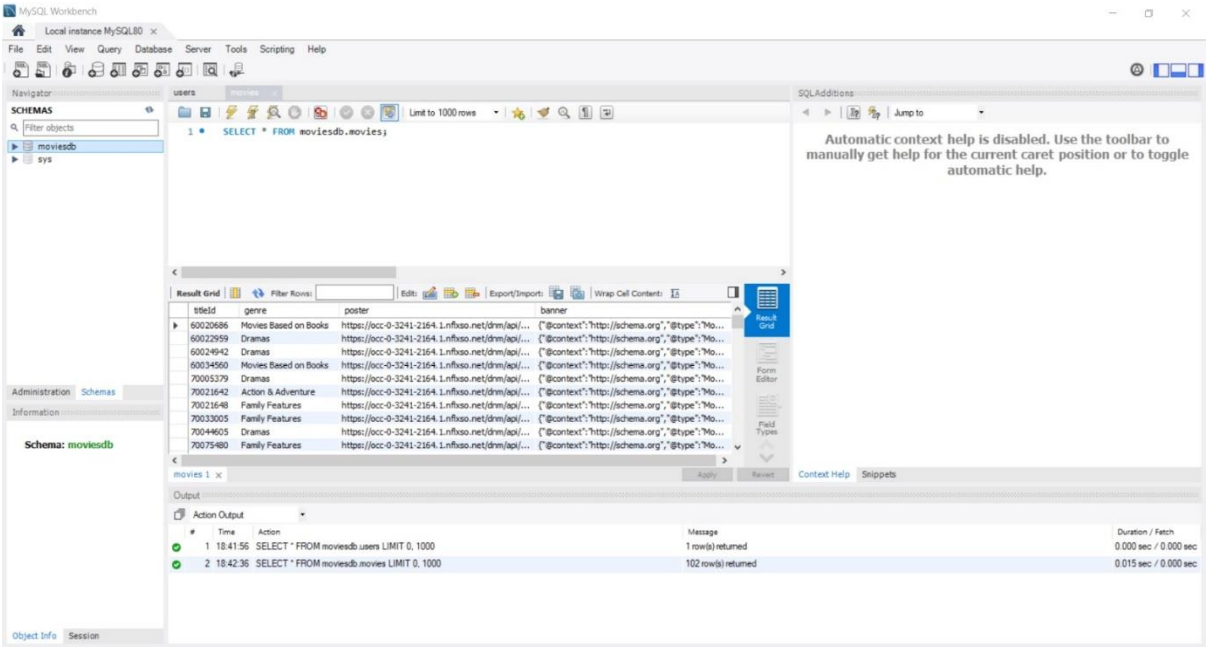
Security is a crucial aspect of any database management system, and MySQL provides robust security features to protect your data. It includes access controls, encryption, and auditing capabilities. With proper configuration, you can ensure that only authorized users have access to sensitive information.

.

Disadvantages:

- Handling Large Databases: MySQL can be inefficient when it comes to handling very large databases. This can limit its use in applications that require the processing of large amounts of data.
- Developing and Debugging Tools: Compared to other databases, MySQL does not have as good developing and debugging tools. This can make the development process more challenging.
- Limited Support for Advanced Features: MySQL versions less than 5.0 do not support COMMIT, stored procedures, and ROLE. This can limit its functionality compared to other databases.
- Transaction Handling: MySQL can be inefficient in handling transactions and is prone to data corruption12. This can pose a risk to data integrity.
- SQL Check Constraints: MySQL does not support SQL check constraints. This can limit the ability to enforce certain business rules at the database level.
- Scalability: MySQL can be hard to scale and does not support auto sharding. You need to maintain the nodes manually.
- Stored Procedures: MySQL has weak stored procedures.
- Database Clustering: It is very difficult to install a consistent database cluster with MySQL.
- Group by Function: Group by function does not work the way they are intended to.
- Errors: Some errors in MySQL can be misleading.
- Correlated Subquery: Correlated subquery does not work as intended.
- Community Development: The development of MySQL is not community-driven1. This can limit the speed of innovation and the responsiveness to user needs.


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
moviesdb
sys
users
1 • SELECT * FROM moviesdb.users;
SQLAdditions
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid
uId
username
recommendations
lastMovie
6d4aaa2f-43ba-4cbe-914e-1f6d87e05137
rock
{{"titleId": 70120612, "title": "Megamind", "genr...
Mowgli: Legend of the Jungle
No object selected
Output
Action Output
1 18:41:56 SELECT * FROM moviesdb.users LIMIT 0, 1000
1 row(s) returned
0.000 sec / 0.000 sec


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
moviesdb
sys
users
movies
1 • SELECT * FROM moviesdb.movies;
SQLAdditions
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid
titleId
genre
poster
banner
60020686 Movies Based on Books
60022959 Dramas
60024942 Dramas
60034560 Movies Based on Books
70005379 Dramas
70021642 Action & Adventure
70021648 Family Features
70033005 Family Features
70044605 Dramas
70075480 Family Features
Schema: moviesdb
Output
Action Output
1 18:41:56 SELECT * FROM moviesdb.users LIMIT 0, 1000
1 row(s) returned
0.000 sec / 0.000 sec
2 18:42:36 SELECT * FROM moviesdb.movies LIMIT 0, 1000
102 row(s) returned
0.015 sec / 0.000 sec

Advantages:

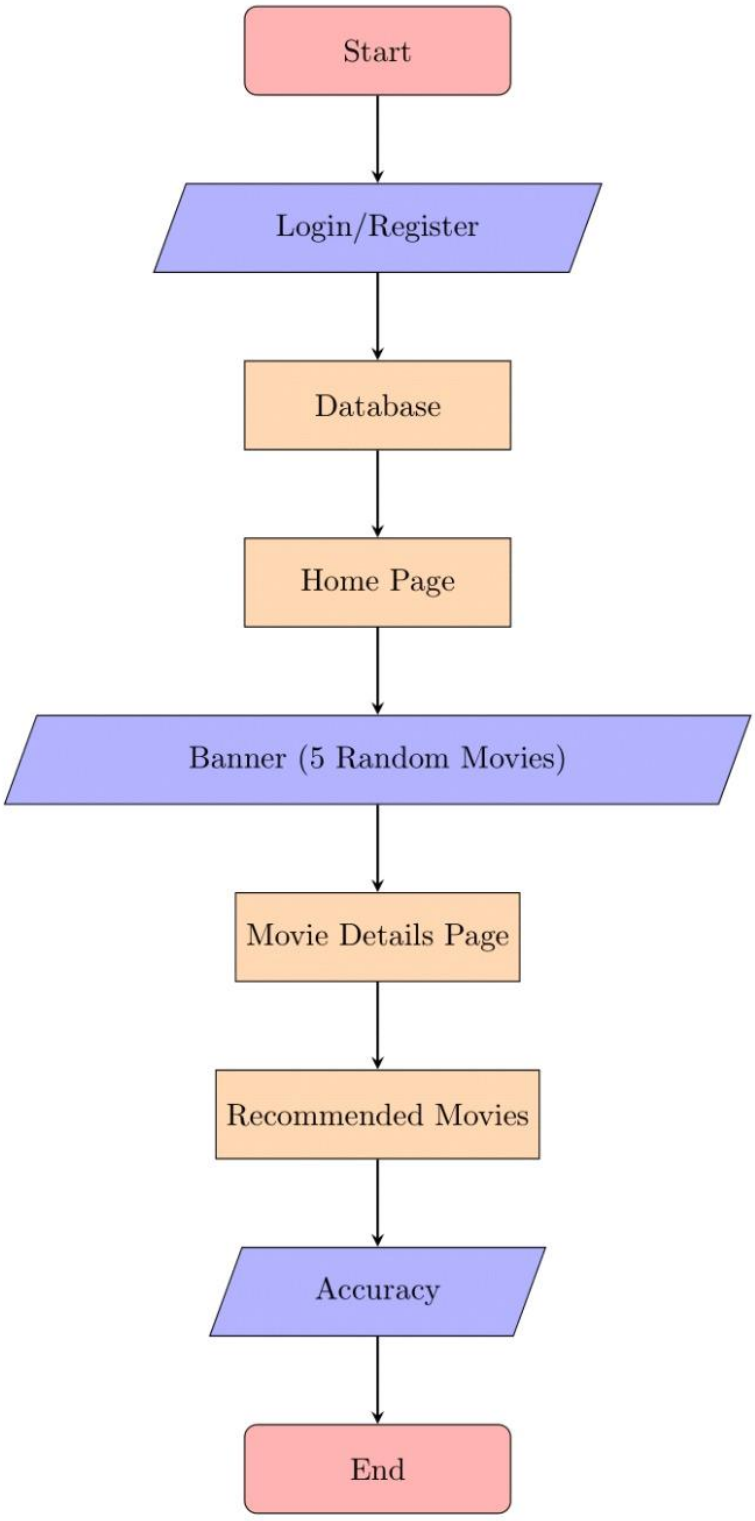
- Open-Source: MySQL is free to download and use, and it benefits from a large community of developers who contribute to its improvement.
- Ease of Use: MySQL is known for its user-friendly interface that allows users to create, modify, and extract data from the relational database with only a basic knowledge of MySQL and a few simple SQL statements.
- Security: MySQL prioritizes data security with its access privilege system and user account management1. Passwords in MySQL are encrypted, adding an extra layer of security.
- Performance and Scalability: MySQL is optimized for performance and can handle high-volume transactions and large datasets efficiently. It is scalable and capable of handling more than 50 million rows.
- Compatibility: MySQL is compatible with most operating systems, including Windows, Linux, NetWare, Novell, Solaris, and other variations of UNIX. It also provides connectors and APIs for popular languages like PHP, Python, Java, and more.
- Unique Storage Engine Architecture: MySQL has a unique storage engine architecture which makes it faster, cheaper, and more reliable.
- Productivity: MySQL gives developers higher productivity by using views, Triggers, and Stored procedures.
- Client-Server Architecture: MySQL has a client-server architecture. There can be any number of clients or application programs that communicate with the database server (MySQL) to query data, save changes, etc.
- Flexibility: MySQL is very flexible as it supports a large number of embedded applications.

NodeJS:

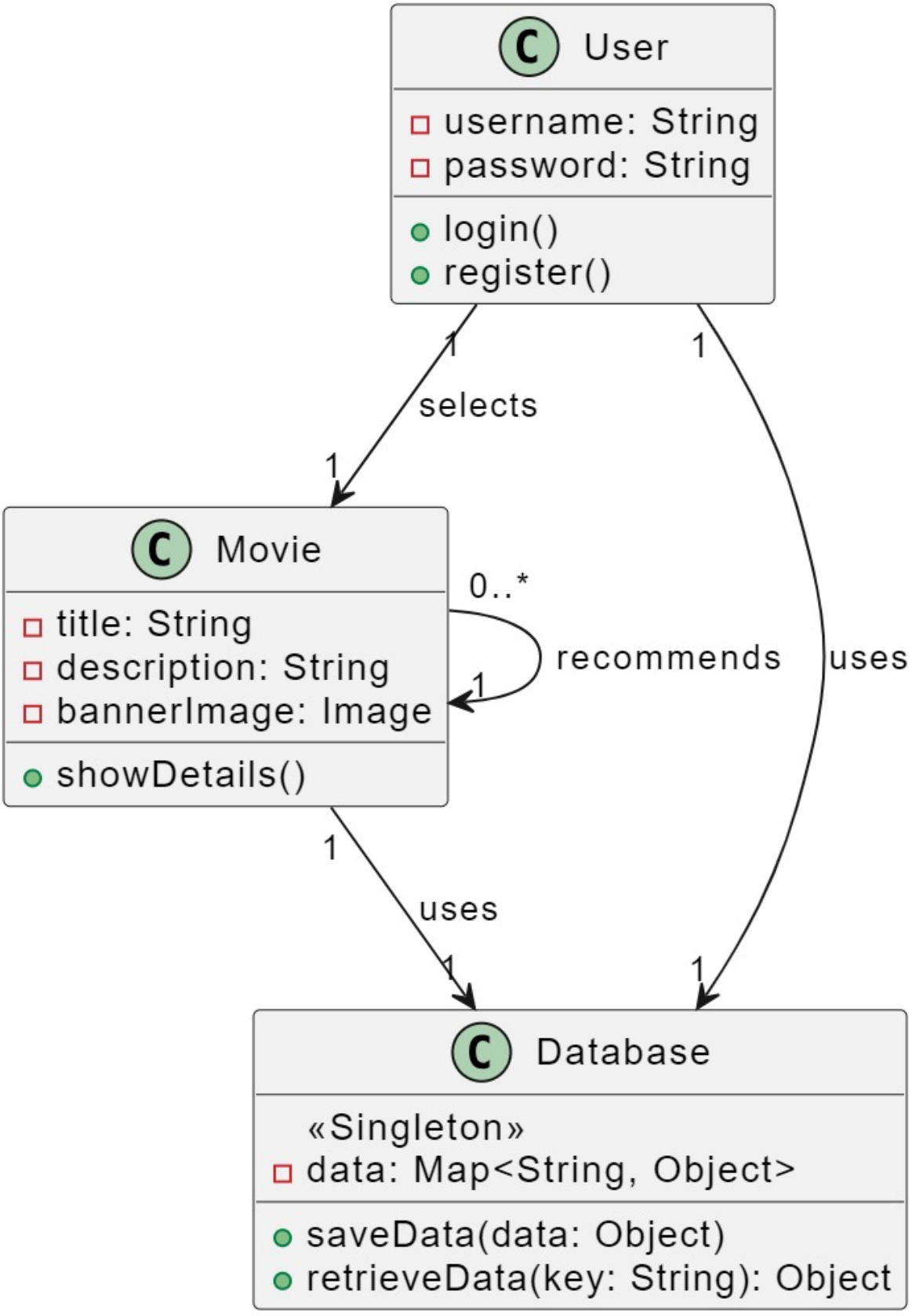
Node.js is an open-source, cross-platform JavaScript runtime environment. It allows developers to execute JavaScript code outside of a web browser. This means you can write server-side scripts in JavaScript to produce dynamic web page content before the page is sent to the user's web browser.

Here are some key features of Node.js:

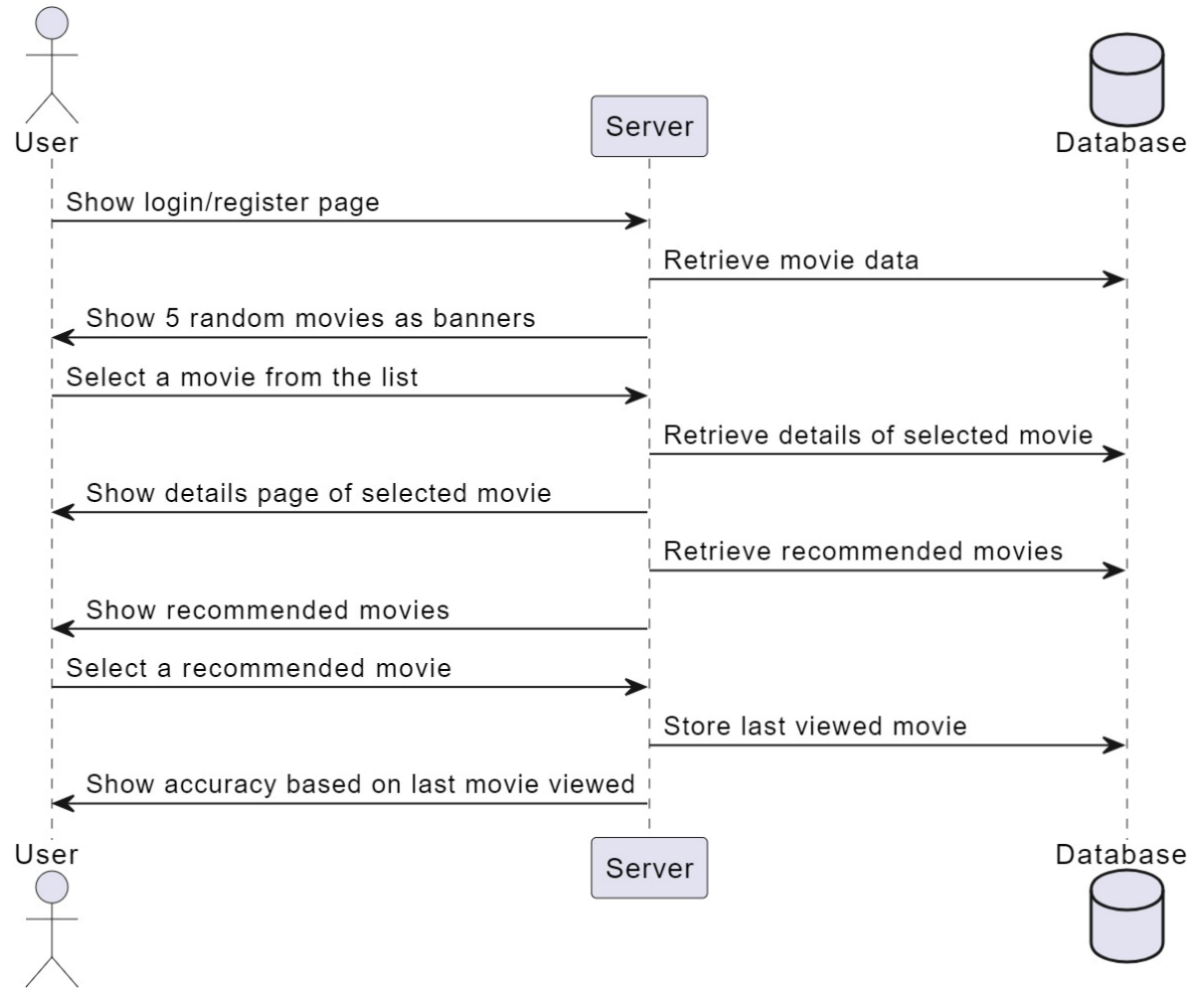
- Asynchronous and Event-Driven: All APIs of Node.js library is asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- Very Fast: Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- Single Threaded but Highly Scalable: Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests.
- No Buffering: Node.js applications never buffer any data. These applications simply output the data in chunks.
- Open Source: Node.js has an open-source community which has produced many excellent modules to add additional capabilities to Node.js applications.
- Cross-Platform: Node.js can be run on various platforms (Windows, Linux, Unix, Mac OS X, etc.).
- Package Manager: Node.js comes with npm (node package manager) which makes it easy to install, update and use software packages.
- Used by Big Companies: Companies like Netflix, Uber, PayPal, and LinkedIn use Node.js for their applications.
- Support for Modern Web Development: Node.js is part of popular web development stacks like the MEAN, MERN, and MEVN stack, all of which use Node for web server operations.
- Microservices Ready: Node.js is a good platform for microservices which are a popular solution for enterprise applications.


Start
Login/Register
Database
Home Page
Banner (5 Random Movies)
Movie Details Page
Recommended Movies
Accuracy
End

CLASS DIAGRAM:



SEQUENCE DIAGRAM:



CODING

index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="
https://assets.nflxext.com/us/ffe/siteui/common/icons/nficon2023.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Netflix</title>
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>

```

index.js:

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { Provider } from 'react-redux'
import store from './app/store.js'

ReactDOM.render(
<React.StrictMode>
  <Provider store = {store}>
      <App />
  </Provider>
</React.StrictMode>,
document.getElementById('root')
);

reportWebVitals();

```

index.css:

```css
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}

```

app.css:

```css
* {
  margin : 0 ;
}

.App {
  position : absolute ;
  width : 100% ;
  height : 100% ;
}
```

app.js:

```
import './App.css';
import HomePage from './components/HomePage';
import {BrowserRouter , Switch , Route} from 'react-router-dom'
import LoginPage from './components/LoginPage';
import RegisterPage from './components/RegisterPage'
import Target from './components/Target';

function App() {
  return (
    <div className="App">
      <BrowserRouter>
        <Switch>
          <Route exact path = '/home' component = {HomePage}/>
          <Route exact path = '/' component = {LoginPage} />
          <Route path = '/register' component = {RegisterPage}
/>
          <Route path = '/title/:id' component = {Target} />
        </Switch>
      </BrowserRouter>
    </div>
  );
}

export default App;

```

Components:

Carousel.jsx:

```
mport { React, useEffect, useRef, useState } from "react";
import {
  BrowserRouter as Router,
  Redirect,
  Route,
  Link,
} from "react-router-dom";
import "../css/Carousel.css";
import { changePrefernce, getMovie } from "../utils/HomePage.fns";
import Target from "./Target";
import store from "../app/store.js";
import { setBigMovies } from "../features/moviesSlice";
import axios from "axios";

function Carousel({ genre, movies, scroll }) {
  const [rightExt, setRightExt] = useState(!scroll);
  const [leftExt, setLeftExt] = useState(true);
  const [scrollLeft, setScrollLeft] = useState(0);
  const carouselTrack = useRef(null);

  const [redirect, setDirect] = useState(false);
  const [id, setId] = useState(null);
  const [clickedMovie, setClickedMovie] = useState(null);

  const rightObserver = new IntersectionObserver(
    (entries) => {
      if (entries[0].isIntersecting === true) {
        setRightExt(true);
      }
    },
    { threshold: [0.5] }
  );

```

```
const leftObserver = new IntersectionObserver(
    (entries) => {
      if (entries[0].isIntersecting === true) {
        setLeftExt(true);
      }
    },
    { threshold: [0.5] }
  );

  const moveForward = (e) => {
    rightObserver.observe(carouselTrack.current.lastChild);
    setLeftExt(false);
    carouselTrack.current.scrollLeft +=
      2 * carouselTrack.current.children[0].getBoundingClientRect().width;
  };

  const moveBackward = (e) => {
    leftObserver.observe(carouselTrack.current.children[0]);
    setRightExt(false);
    carouselTrack.current.scrollLeft -=
      2 * carouselTrack.current.children[0].getBoundingClientRect().width;
  };

  const imgClick = async (e) => {
    e.preventDefault()
    changePrefernce(JSON.parse(e.target.attributes.state.value).title)
    setClickedMovie(JSON.parse(e.target.attributes.state.value))
    setDirect(true)
}

```

```
useEffect(() => {
     console.log(`clicked movie`,clickedMovie)
 },[clickedMovie])

 return redirect ? (
   <Redirect
     from="/home"
     push
     to={{
       pathname: `/title/${clickedMovie.titleId}`,
       state: clickedMovie,
       target: false,
     }}
   />
 ) : (
   <div className="carousel">
     <div className="carousel_cover">
       <div className="carousel_cover__title">{genre}</div>
       <div className="carousel_track_div">
         {!rightExt && !leftExt ? (
           <>
             <div className="right_overlay"></div>
             <div className="left_overlay"></div>
           </>
         ) : leftExt ? (
           <div className="right_overlay"></div>
         ) : (
           <div className="left_overlay"></div>
         )}
         <div
           style={{ visibility: leftExt ? "hidden" : "visible" }}
           role="button"
           onClick={moveBackward}
           className="prev"
         >{`<`}</div>
         <ul ref={carouselTrack} className="carousel_track">
           {movies.map((movie) => (
             <li>
               <img
                 state={JSON.stringify(movie)}
                 role="button"
                 onClick={(e) => imgClick(e)}
                 src={movie.poster}
               />
             </li>
           ))}
         </ul>
         <div
           style={{ visibility: rightExt ? "hidden" : "visible" }}
           onClick={moveForward}
           role="button"
           className="after"
         >{`>`}</div>
       </div>
     </div>
   </div>
 );
}

export default Carousel;
```

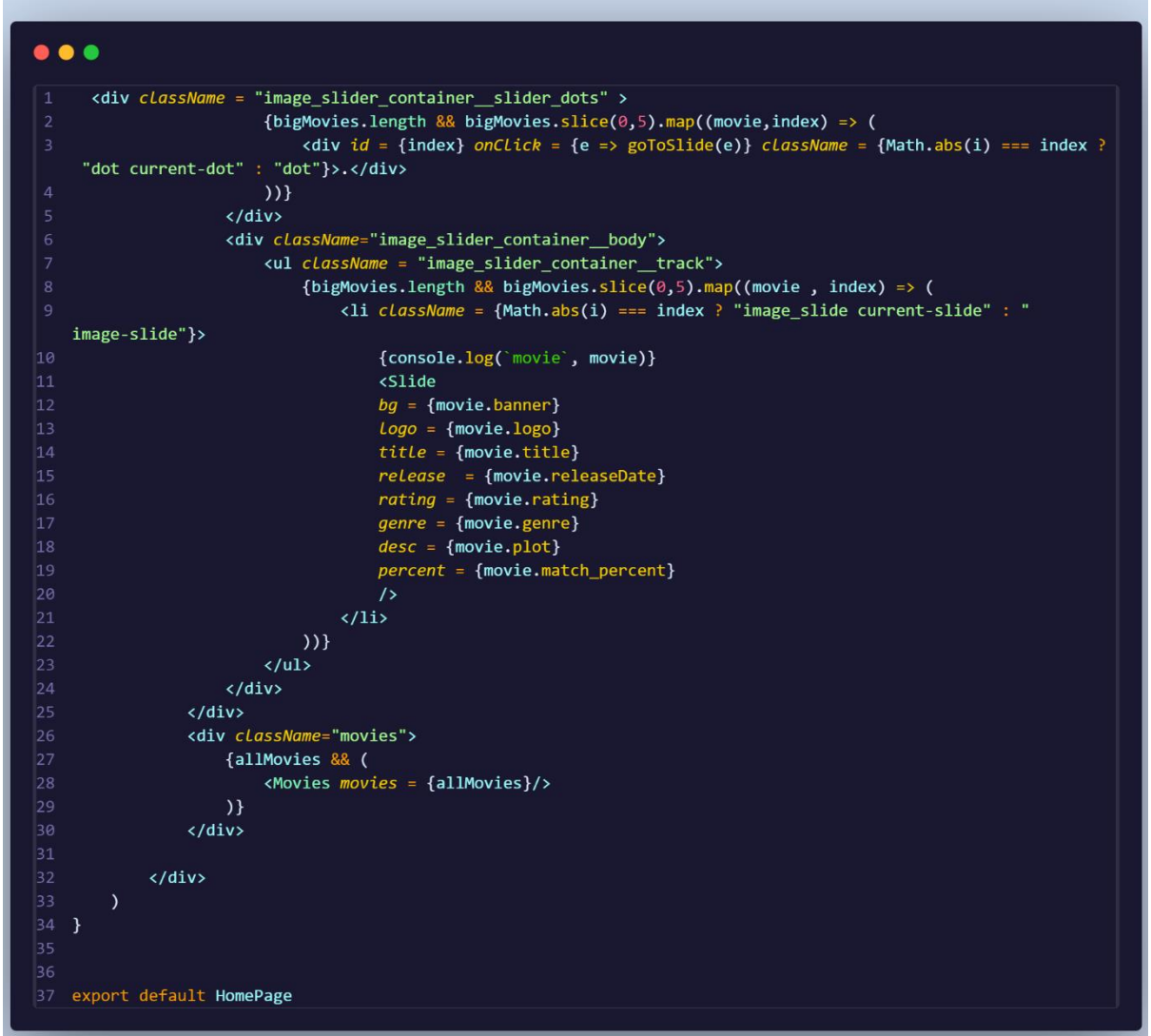
HomePage.jsx

```jsx
import React, { forwardRef, useEffect, useRef, useState } from 'react'
import styled from 'styled-components'
import netflixlogo from '../images/netflix-logo.png'
import SearchIcon from '@mui/icons-material/Search';
import {IconButton, ListItemIcon, Menu, MenuItem} from '@mui/material'
import NotificationsIcon from '@mui/icons-material/Notifications';
import PlayArrowIcon from '@mui/icons-material/PlayArrow';
import AddIcon from '@mui/icons-material/Add';
import '../css/HomePage.css'
import {useDispatch} from 'react-redux'
import store from '../app/store'
import backIcon from '../images/back-icon.png'
import forwardIcon from '../images/forward-icon.png'
import {getMovies, getUser} from '../utils/HomePage.fns.js'
import Slide from '../components/Slide.jsx'
import '../css/Slide.css'
import Movies from '../components/Movies.jsx'
import { getGenreData } from '../utils/HomePage.fns.js';
import { setLogout } from '../features/userSlice';
import Logout from '@mui/icons-material/Logout';
import {useHistory} from 'react-router-dom'

function HomePage() {
    const [results, setResults] = useState([]);
    const history = useHistory()
    const [bigMovies , setBigMovies] = useState([])
    const [i, setI] = useState(0)
    const [allMovies , setAllMovies] = useState([])

    const expand = () => {
        let input = document.getElementsByClassName("search_box_input")
        if(input[0].classList.length === 2) {
            input[0].classList.remove("search_box_input_select")
        }
        else {
            input[0].classList.add("search_box_input_select")
        }
    }

    const moveForward = () => {
        setI((i+1)%5)
    }

    const moveBackward = () => {
        setI((i-1)%5)
    }

    const goToSlide = (e) => {
        setI(e.target.id)
    }

    const handleLogout = () => {
        store.dispatch(setLogout())
        document.cookie ='_ga=; Max-Age=-99999'
        history.push('/')
    }

    useEffect(() => {
        async function getAllMovies() {
            const movies = await getGenreData()
            setAllMovies(movies)
        }
        async function getKarneWala() {
            await getUser()
            setBigMovies(store.getState().movieReducer.bigSliderMovies)
        }
        getKarneWala()
        getAllMovies()
    }, [])

```

```
1    const [anchorEl, setAnchorEl] = React.useState(null);
2        const open = Boolean(anchorEl);
3        const handleClick = (event) => {
4          setAnchorEl(event.currentTarget);
5        };
6        const handleClose = () => {
7          setAnchorEl(null);
8        };
9
10       return (
11           <div className = "home_page" style = {{width : "
     100%", height : "100%" ,  display : "flex" ,
     flexDirection  : "column", overflowX : 'hidden' ,
     position : 'relative' }}>
12               <div className = "header" style = {{
     position : "absolute" , width : "100%" , height : "62px"
     , backgroundColor : "transparent" ,  display  : "flex" ,
      alignItems : "center" , zIndex: 1}}>
13               <Menu
14                    anchorEl={anchorEl}
15                    open={open}
16                    onClose={handleClose}
17                    onClick={handleClose}
18                    PaperProps={{
19                    elevation: 0,
20                    sx: {
21                         overflow: 'visible',
22                         filter: '
     drop-shadow(0px 2px 8px rgba(0,0,0,0.32))',
23                         mt: 1.5,
24                         bgcolor : '#181818' ,
25                         color : "#fff" ,
26                         '& .MuiAvatar-root': {
27                         width: 32,
28                         height: 32,
29                         ml: -0.5,
30                         mr: 1,
31                         },
32                         '&:before': {
33                         content: '""',
34                         display: 'block',
35                         position: 'absolute',
36                         top: 0,
37                         right: 14,
38                         width: 10,
39                         height: 10,
40                         bgcolor: '#181818',
41                         transform: '
     translateY(-50%) rotate(45deg)',
42                         zIndex: 0,
43                         },
44                    },
45                    }}
46                    transformOrigin={{ horizontal: 'right',
     vertical: 'top' }}
47                    anchorOrigin={{ horizontal: 'right',
     vertical: 'bottom' }}
48                >
49                    <MenuItem>
50                    Hello {store.getState().userReducer.
     username} !
51                    </MenuItem>
52                    <MenuItem>
53                    <ListItemIcon onClick = {handleLogout}>
54                         <Logout color = "primary" fontSize="
     small" />
55                    </ListItemIcon>
56                    Logout
57                    </MenuItem>
```

```
    </Menu>
                <div className = "header_netflix_logo" style = {{width : "fit-content" , margin :
"0 40px 0 40px", cursor : "pointer" , height : "100%" , display : "flex" , order  : 1 ,
justifyContent : "flex-start"}}>
                    <img src = {netflixlogo} style = {{width : "100%" , height : "100%" ,
objectFit : "contain"}} />
                </div>
                <div className = "header_categories" style = {{width : "50%" , order : 2}}>
                </div>
                <div className = "header_types" style = {{order : 3 , display : "flex" , width : "
40%" , alignItems : "center"}}>
                    <div className = "header_types_cover">
                        <div className = "search_box">
                            <form style ={{display : "flex" , width : "100%"}}>
                                <input className = "search_box_input" />
                                <IconButton onClick = {expand}>
                                    <SearchIcon style = {{color : "rgba(180,180,180)"}} />
                                </IconButton>
                            </form>
                        </div>
                        <div onClick = {handleClick} className="profile_div">
                            <img src = "
```

[illegible]
```
" className = "profile_icon" />
                        </div>
                    </div>
                </div>
            </div>
            <div className="image_slider_container">
                <div className="overlay"></div>
                <div className="image_slider_container__bnf">
                    <div className="bnf-relative">
                        <div role = "button" onClick = {moveBackward} className = "
image_slider_container__bnf--back">
                            <img src = {backIcon} />
                        </div>
                        <div role = "button" onClick = {moveForward} className="
image_slider_container__bnf--next">
                            <img src = {forwardIcon} />
                        </div>
                    </div>
                </div>
```



```
<div className = "image_slider_container__slider_dots" >
                    {bigMovies.length && bigMovies.slice(0,5).map((movie,index) => (
                        <div id = {index} onClick = {e => goToSlide(e)} className = {Math.abs(i) === index ? "dot current-dot" : "dot"}>.</div>
                    ))}
                </div>
                <div className="image_slider_container__body">
                    <ul className = "image_slider_container__track">
                        {bigMovies.length && bigMovies.slice(0,5).map((movie , index) => (
                            <li className = {Math.abs(i) === index ? "image_slide current-slide" : "image-slide"}>
                                {console.log(`movie`, movie)}
                                <Slide
                                bg = {movie.banner}
                                logo = {movie.logo}
                                title = {movie.title}
                                release  = {movie.releaseDate}
                                rating = {movie.rating}
                                genre = {movie.genre}
                                desc = {movie.plot}
                                percent = {movie.match_percent}
                                />
                            </li>
                        ))}
                    </ul>
                </div>
            </div>
            <div className="movies">
                {allMovies && (
                    <Movies movies = {allMovies}/>
                )}
            </div>

        </div>
    )
}


export default HomePage
```

ImageSlider.jsx:

```jsx
import React from 'react'
import Carousel from './Carousel.jsx'
import '../css/ImageSlider.css'

function ImageSlider({title}) {
    return (
        <div style = {{margin : "5em 0"}}>
            <span className = "image_slider_title">{title}</span>
            <div className = "image_slider">
                <div className = "image_slider_cover">
                </div>
            </div>
        </div>
    )
}

export default ImageSlider
```

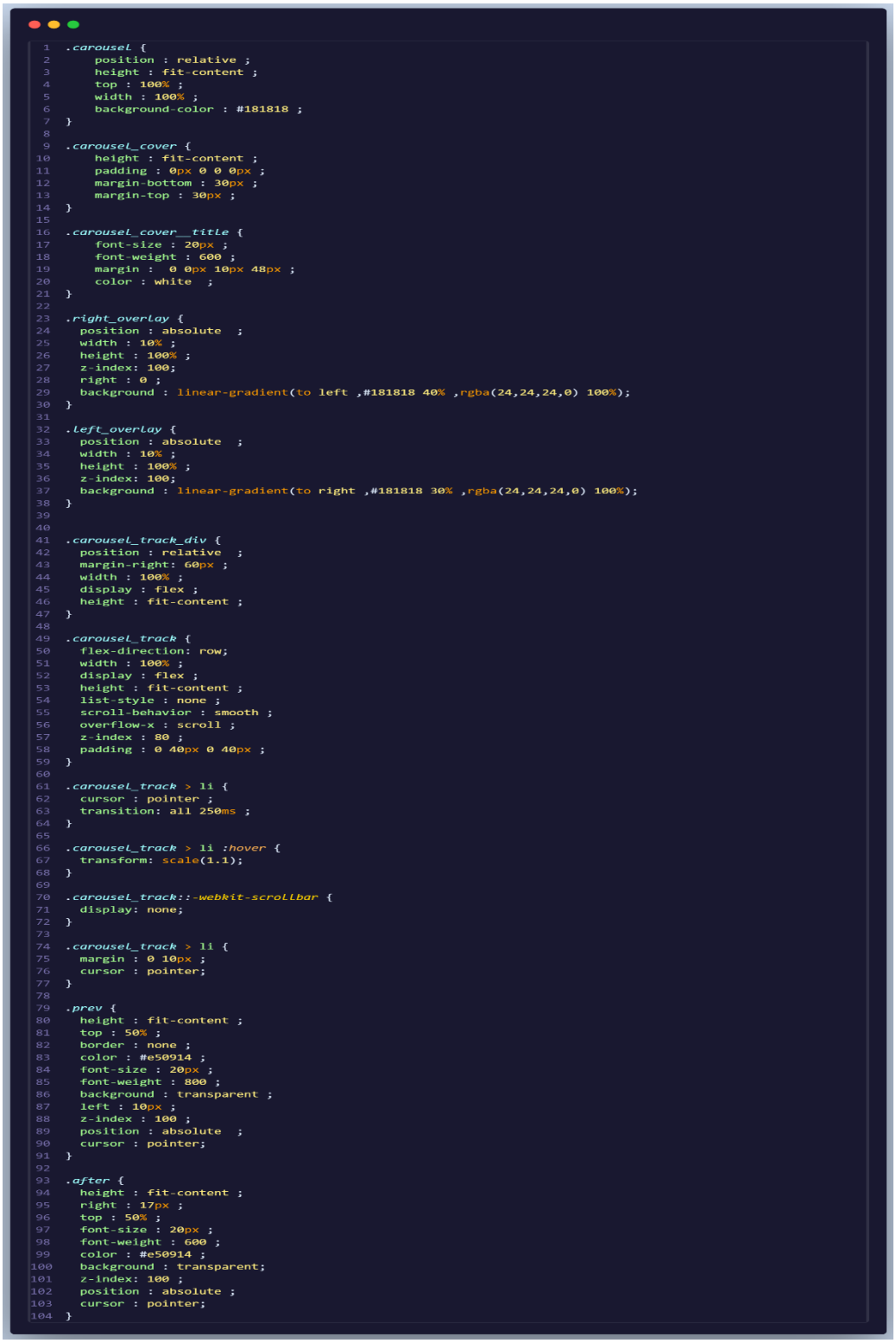
Movies.jsx:

```jsx
import React from 'react';
import Carousel from './Carousel.jsx';

function Movies({ movies }) {
    return (
        <div style={{ position: "relative", width: "100%", height: "fit-content", display: "flex", flexDirection: 'column' }}>
            <Carousel
                key={movies.titleId}
                movies={movies}
            />
        </div>
    );
}

export default Movies;
```

Slide.jsx:

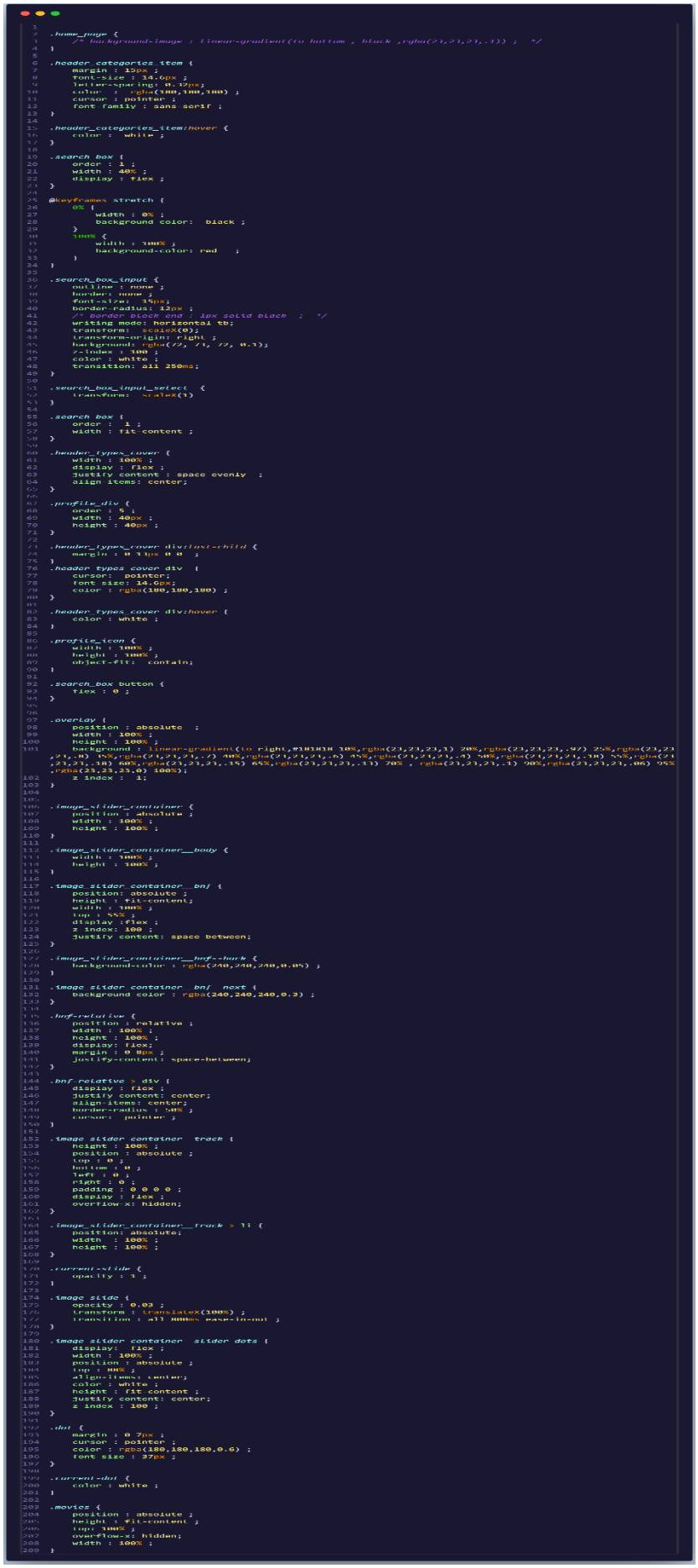
```jsx
import React from 'react'
import PlayArrowIcon from '@mui/icons-material/PlayArrow';
import AddIcon from '@mui/icons-material/Add';

function Slide({bg, logo, title , release, rating, genre , desc , percent }) {
    return (
        <div className = "slide">
            <img className =  "slide__bg" src = {bg} />
            <img className = "slide__logo" src =  {logo} />
            <span className = "slide__title">{title}</span>
            <div className="slide__btns">
              <div className = "slide__btns slide__btn play">
                    <PlayArrowIcon />
                    <span>Play</span>
                </div>
                <div className="slide__btns slide__btn mylist">
                    <AddIcon />
                    <span>My List</span>
                </div>
            </div>
            <div className="slide__desc">
                <span className = "slide__desc__release">{release}</span>
                <span className = "spacer">|</span>
                <span className = "slide__desc__rating">{rating}</span>
                <span className = "spacer">|</span>
                <span className = "slide__desc__genre">{genre}</span>
                <span className = "spacer">|</span>
                <span className = "slide__desc__percent">{percent ? `${percent}%` : ``} </span>
                <span className="percent" style = {{fontFamily : "sans-serif" , margin : "0 0 0 8px" , fontWeight : "400" , color : "greenyellow"}}>{percent ? 'match' : ''}</span>
            </div>
            <div className = "slide__plot">{desc}</div>
        </div>
    )
}

export default Slide
```

Carouser.css:



```css
.carousel {
    position : relative ;
    height : fit-content ;
    top : 100% ;
    width : 100% ;
    background-color : #181818 ;
}

.carousel_cover {
    height : fit-content ;
    padding : 0px 0 0 0px ;
    margin-bottom : 30px ;
    margin-top : 30px ;
}

.carousel_cover__title {
    font-size : 20px ;
    font-weight : 600 ;
    margin :  0 0px 10px 48px ;
    color : white  ;
}

.right_overlay {
  position : absolute  ;
  width : 10% ;
  height : 100% ;
  z-index: 100;
  right : 0 ;
  background : linear-gradient(to left ,#181818 40% ,rgba(24,24,24,0) 100%);
}

.left_overlay {
  position : absolute  ;
  width : 10% ;
  height : 100% ;
  z-index: 100;
  background : linear-gradient(to right ,#181818 30% ,rgba(24,24,24,0) 100%);
}


.carousel_track_div {
  position : relative  ;
  margin-right: 60px ;
  width : 100% ;
  display : flex ;
  height : fit-content ;
}

.carousel_track {
  flex-direction: row;
  width : 100% ;
  display : flex ;
  height : fit-content ;
  list-style : none ;
  scroll-behavior : smooth ;
  overflow-x : scroll ;
  z-index : 80 ;
  padding : 0 40px 0 40px ;
}

.carousel_track > li {
  cursor : pointer ;
  transition: all 250ms ;
}

.carousel_track > li :hover {
  transform: scale(1.1);
}

.carousel_track::-webkit-scrollbar {
  display: none;
}

.carousel_track > li {
  margin : 0 10px ;
  cursor : pointer;
}

.prev {
  height : fit-content ;
  top : 50% ;
  border : none ;
  color : #e50914 ;
  font-size : 20px ;
  font-weight : 800 ;
  background : transparent ;
  left : 10px ;
  z-index : 100 ;
  position : absolute  ;
  cursor : pointer;
}

.after {
  height : fit-content ;
  right : 17px ;
  top : 50% ;
  font-size : 20px ;
  font-weight : 600 ;
  color : #e50914 ;
  background : transparent;
  z-index: 100 ;
  position : absolute ;
  cursor : pointer;
}
```

HomePage.css:

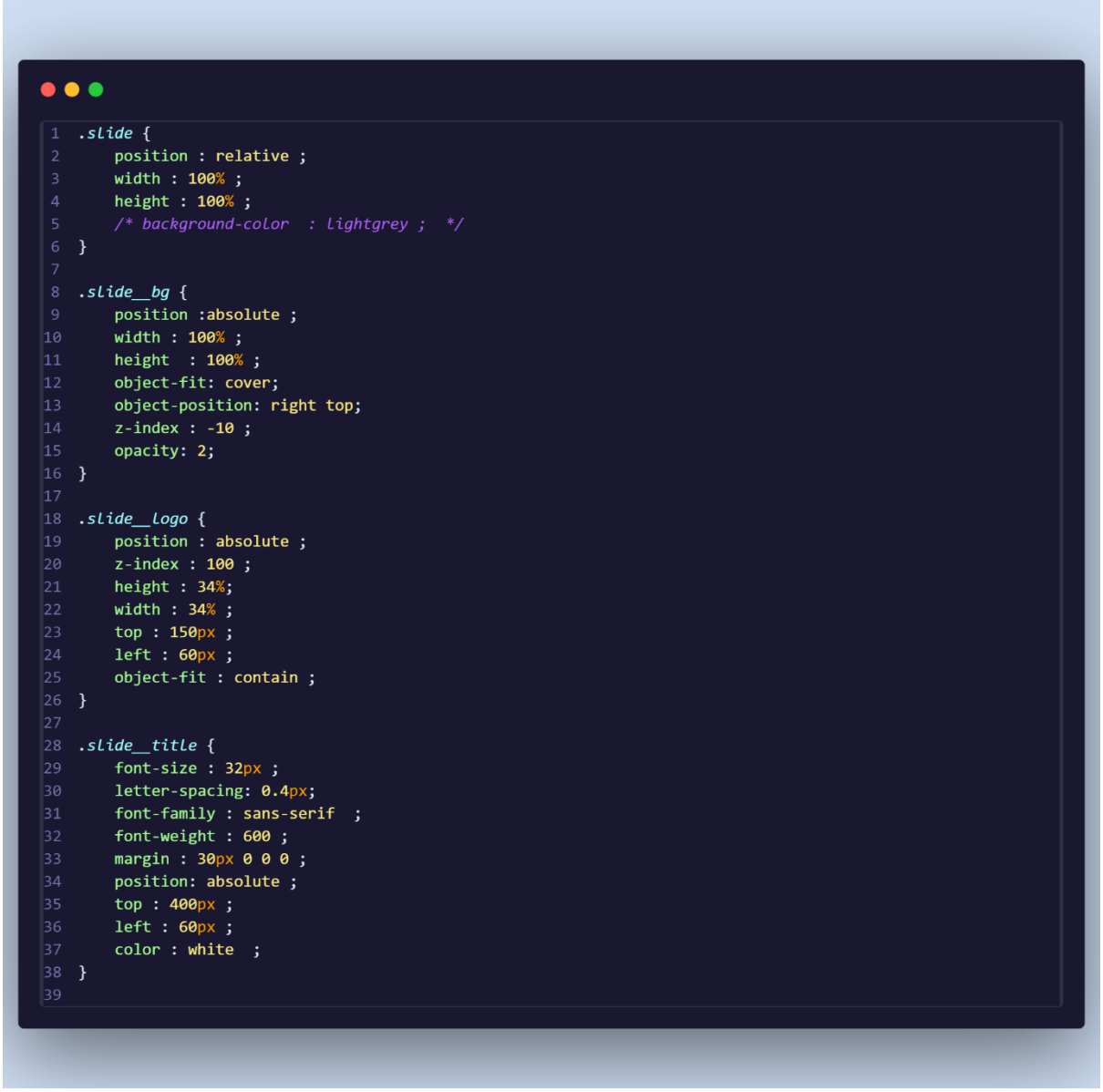


```
.home_page {
    /* background-image : linear-gradient(to bottom , black ,rgba(21,21,21,.7)) ;  */
}

.header_categories_item {
    margin : 15px ;
    font-size : 14.6px ;
    letter-spacing: 0.17px;
    color  : rgba(180,180,180) ;
    cursor : pointer ;
    font family : sans serif ;
}

.header_categories_item:hover {
    color :  white ;
}

.search_box {
    order : 1 ;
    width : 40% ;
    display : flex ;
}

@keyframes stretch {
    0% {
        width : 0% ;
        background color:  black ;
    }
    100% {
        width : 100% ;
        background-color: red    ;
    }
}

.search_box_input {
    outline : none ;
    border: none ;
    font-size:  15px;
    border-radius: 12px ;
    /* border block end : 1px solid black  ;  */
    writing mode: horizontal tb;
    transform:  scaleX(0);
    transform-origin: right ;
    background: rgba(77, 71, 72, 0.1);
    z-index : 100 ;
    color : white ;
    transition: all 250ms;
}

.search_box_input_select  {
    transform:  scaleX(1)
}

.search_box {
    order :  1 ;
    width : fit-content ;
}

.header_types_cover {
    width : 100% ;
    display : flex ;
    justify content : space evenly  ;
    align items: center;
}

.profile_div {
    order : 5 ;
    width : 40px ;
    height : 40px ;
}

.header_types_cover div:last-child {
    margin : 0 11px 0 0  ;
}
.header types cover div  {
    cursor:  pointer;
    font size: 14.6px;
    color : rgba(180,180,180) ;
}

.header_types_cover div:hover {
    color : white ;
}

.profile_icon {
    width : 100% ;
    height : 100% ;
    object-fit:  contain;
}

.search_box button {
    flex : 0 ;
}


.overlay {
    position : absolute  ;
    width : 100% ;
    height : 100% ;
    background : linear-gradient(to right,#181818 10%,rgba(21,21,21,1) 20%,rgba(21,21,21,.97) 25%,rgba(21,21,21,.94) 30%,rgba(21,21,21,.7) 40%,rgba(21,21,21,.5) 50%,rgba(21,21,21,.3) 55%,rgba(21,21,21,.18) 60%,rgba(21,21,21,.15) 65%,rgba(21,21,21,.11) 70% , rgba(21,21,21,.1) 90%,rgba(21,21,21,.06) 95%,rgba(23,23,23,0) 100%);
    z index :  1;
}


.image_slider_container {
    position : absolute ;
    width : 100% ;
    height : 100% ;
}

.image_slider_container__body {
    width : 100% ;
    height : 100% ;
}

.image_slider_container__bnf {
    position: absolute ;
    height : fit-content;
    width : 100% ;
    top : 55% ;
    display :flex ;
    z index: 100 ;
    justify content: space between;
}

.image_slider_container__bnf-back {
    background-color : rgba(240,240,240,0.05) ;
}
.image slider container  bnf  next {
    background color : rgb(240,240,240,0.2) ;
}

.bnf-relative {
    position : relative ;
    width : 100% ;
    height : 100% ;
    display: flex;
    margin : 0 8px ;
    justify-content: space-between;
}

.bnf relative > div {
    display : flex ;
    justify content: center;
    align-items: center;
    border-radius : 50% ;
    cursor:  pointer ;
}
.image slider container  track {
    height : 100% ;
    position : absolute ;
    top : 0 ;
    bottom : 0 ;
    left : 0 ;
    right : 0 ;
    padding : 0 0 0 0 ;
    display : flex ;
    overflow-x: hidden;
}

.image_slider_container__track > li {
    position: absolute;
    width  : 100% ;
    height : 100% ;
}

.current-slide {
    opacity : 1 ;
}

.image slide {
    opacity : 0.03 ;
    transform : translatex(100%) ;
    transition : all 888ms ease-in-out ;
}

.image slider container  slider dots {
    display:  flex ;
    width : 100% ;
    position : absolute ;
    top : 88% ;
    align-items: center;
    color : white ;
    height : fit content ;
    justify content: center;
    z index : 100 ;
}

.dot {
    margin : 0 7px ;
    cursor : pointer ;
    color : rgba(180,180,180,0.6) ;
    font size : 37px ;
}

.current-dot {
    color : white ;
}

.movies {
    position : absolute ;
    height : fit-content ;
    top: 108% ;
    overflow-x: hidden;
    width : 100% ;
}
```

ImageSlider.css:

```css
.image_slider {
    width : 100% ;
    height : 7em ;
    position : absolute ;
    display : flex ;
    background : transparent;
    margin-top : 12px ;
}

.image_slider_title {
    font-size : 22px ;
    font-weight : 500 ;
    margin : 0 0 0 20px ;
}

.image_slider_cover {
    position: relative;
    display : flex;
    width : 100% ;
    height : 100% ;
    background : transparent;
    margin : 0 20px 0 20px  ;
}
```

Slide.css:



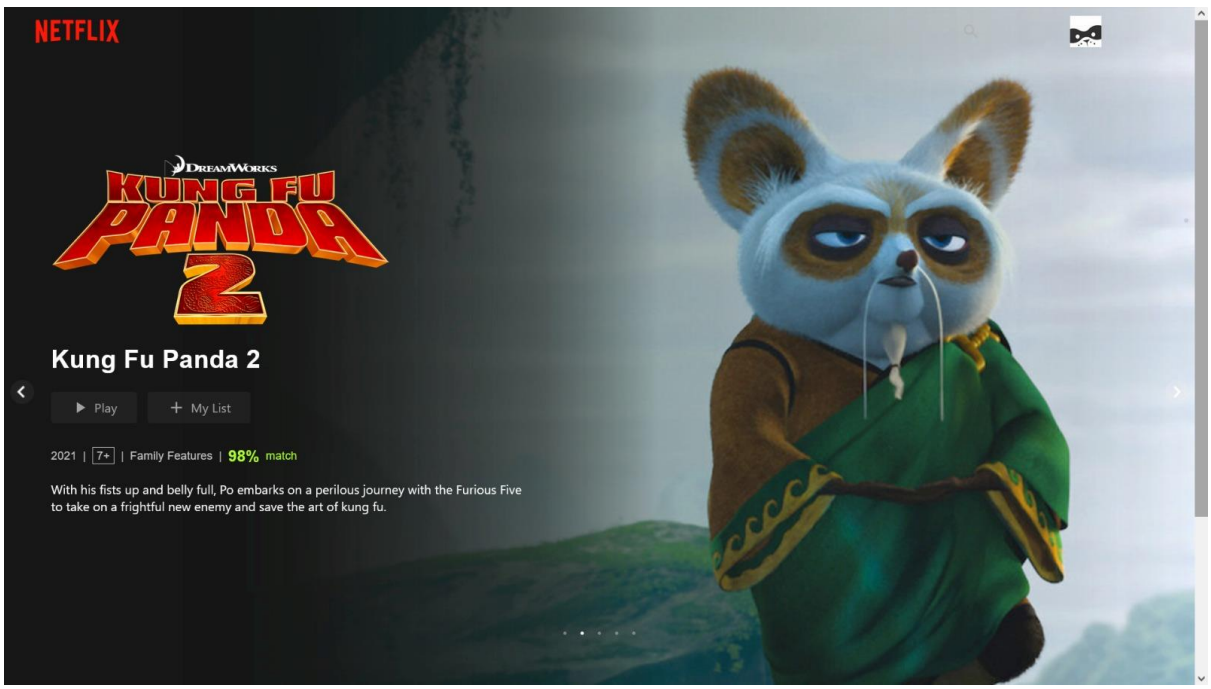
```css
.slide {
    position : relative ;
    width : 100% ;
    height : 100% ;
    /* background-color  : lightgrey ;  */
}

.slide__bg {
    position :absolute ;
    width : 100% ;
    height  : 100% ;
    object-fit: cover;
    object-position: right top;
    z-index : -10 ;
    opacity: 2;
}

.slide__logo {
    position : absolute ;
    z-index : 100 ;
    height : 34%;
    width : 34% ;
    top : 150px ;
    left : 60px ;
    object-fit : contain ;
}

.slide__title {
    font-size : 32px ;
    letter-spacing: 0.4px;
    font-family : sans-serif  ;
    font-weight : 600 ;
    margin : 30px 0 0 0 ;
    position: absolute ;
    top : 400px ;
    left : 60px ;
    color : white  ;
}

```

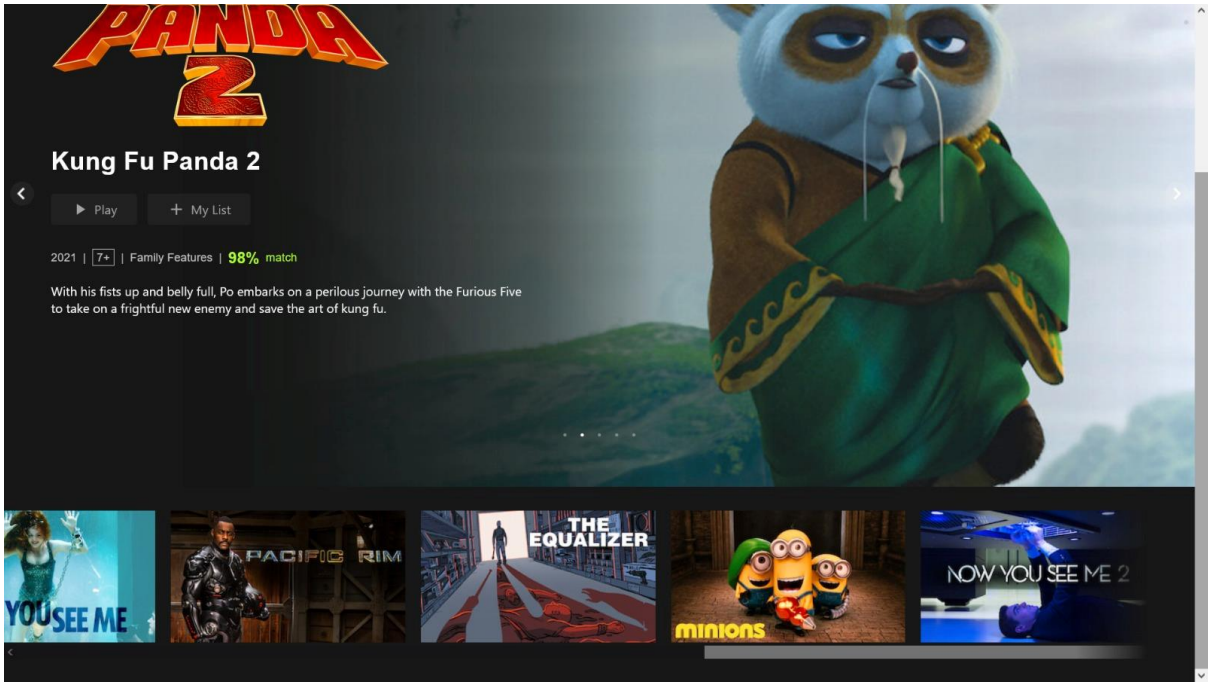
```css
.slide__btns {
    position: relative;
    left : 30px ;
    top : 245px ;
    display: flex;
    width : fit-content ;
    color : rgba(180,180,180,0.9) ;
}

.slide__btn {
    display : flex;
    align-items: center;
    cursor: pointer;
    margin : 0 14px 0 0 ;
    background-color: rgba(180,180,180,0.06);
    padding : 8px 25px ;
    border-radius:  4px ;
}

.slide__btn > span {
    font-size : 16.3px ;
    margin : 0 0 0 6px ;
    transform : scale(1) ;
    transition : all 250ms ;
}

.slide__btn:hover{
    color : white ;
    transform : scale(1.06)  ;
}

.slide__desc {
    display : flex ;
    color :  rgba(240, 240, 240, 0.8) ;
    top: 520px ;
    left : 60px ;
    align-items : center ;
    width : fit-content ;
    position: relative;
    font-size: 14.7px ;
    font-family: sans-serif;
}

.spacer {
    margin : 0 8px ;
}


.slide__desc__rating {
    border : 1px solid rgba(240,240,240,0.8) ;
    padding : 1px 5px ;
}


.slide__desc__percent {
    font-size : 20px;
    font-weight: 700 ;
    color : greenyellow ;
}

.slide__plot {
    color : white ;
    position : relative ;
    top : 540px ;
    left : 60px ;
    width : 40% ;

}
```

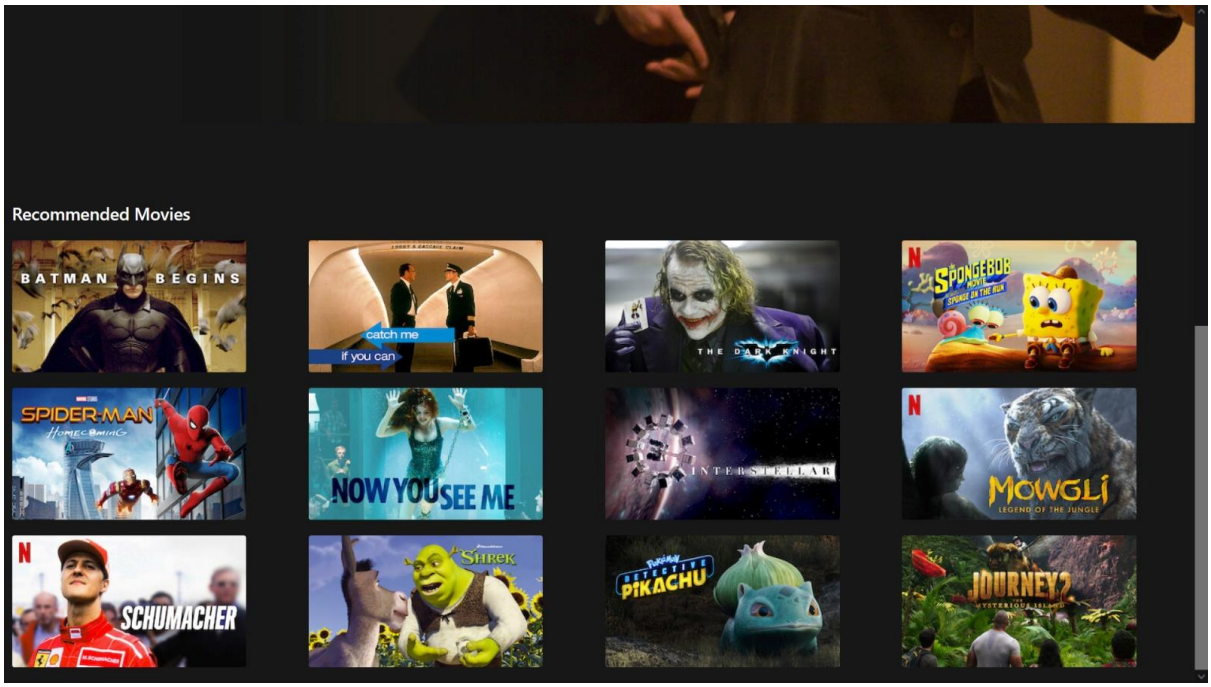
recommender.py:

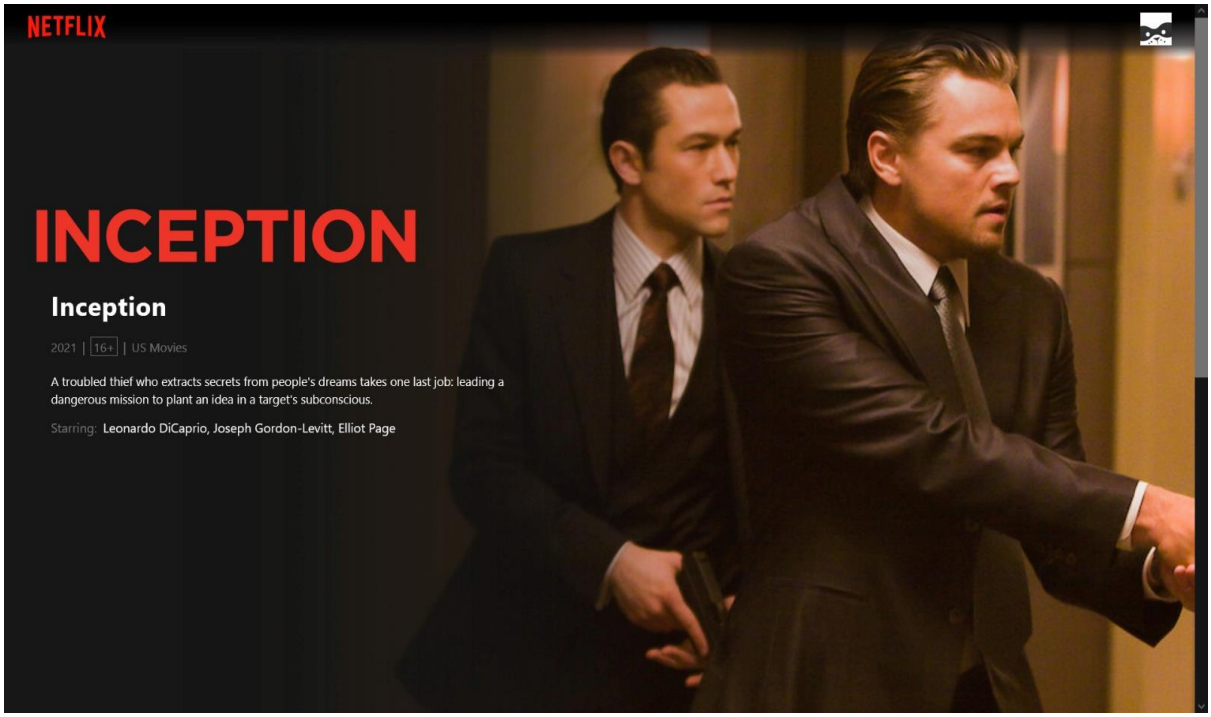
```python
import pickle
import pandas as pd
import sys
import simplejson


movie_dict = pickle.load(open('./movie_dict.pkl' , 'rb'))
movies = pd.DataFrame(movie_dict)
similarity = pickle.load(open('./similarity.pkl','rb'))

def recommend(movie) :
    movie_index = movies[movies['title'] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(similarity[movie_index])) ,reverse = True , key = lambda x : x[1])[
1:13]
    percent_array = []
    for i in movies_list :
        percent_array.append(i[1])

    recommended_movies = []
    for i in movies_list :
        match_percent = (i[1]/max(percent_array)) * 100
        recommended_movies.append({
        "titleId" : int(movies.iloc[i[0]].titleId) ,
        "title" : movies.iloc[i[0]].title,
        "genre" : movies.iloc[i[0]].genre ,
        "banner" : movies.iloc[i[0]].banner ,
        "poster" : movies.iloc[i[0]].poster ,
        "actors" : movies.iloc[i[0]].actors.split(",") ,
        "director" : movies.iloc[i[0]].director.split(",") ,
        "releaseDate" : int(movies.iloc[i[0]].releaseDate) ,
        "plot" : movies.iloc[i[0]].overview ,
        "rating" : movies.iloc[i[0]].rating ,
        "logo" : movies.iloc[i[0]].logo ,
        "thumbnail" : movies.iloc[i[0]].thumbnail  ,
        "trailer" : movies.iloc[i[0]].trailer ,
        "match_percent" : int(match_percent)
        })

    return simplejson.dumps(recommended_movies , ignore_nan = True)

print(recommend(sys.argv[1]))
```

OUTPUT OF THE PROJECT




Kung Fu Panda 2
Play
My List
2021 | 7+ | Family Features | 98% match
With his fists up and belly full, Po embarks on a perilous journey with the Furious Five to take on a frightful new enemy and save the art of kung fu.
YOU SEE ME
PACIFIC RIM
THE EQUALIZER
MINIONS
NOW YOU SEE ME 2


Recommended Movies
BATMAN BEGINS
catch me if you can
THE DARK KNIGHT
SPONGEBOB MOVIE
SPONGE ON THE RUN
SPIDER-MAN
HOMECOMING
NOW YOU SEE ME
INTERSTELLAR
MOWGLI
LEGEND OF THE JUNGLE
SCHUMACHER
SHREK
DETECTIVE PIKACHU
JOURNEY 2


NETFLIX
INCEPTION
Inception
2021 | 16+ | US Movies
A troubled thief who extracts secrets from people's dreams takes one last job: leading a dangerous mission to plant an idea in a target's subconscious.
Starring: Leonardo DiCaprio, Joseph Gordon-Levitt, Elliot Page

CONCLUSION

In conclusion, the movie recommendation system project has demonstrated the power and potential of machine learning in enhancing user experience in the digital entertainment industry. The system effectively utilized collaborative filtering and content-based filtering methods, providing personalized movie recommendations based on user preferences and viewing history.

The hybrid model addressed the cold-start problem, ensuring new users also receive relevant recommendations. The incorporation of deep learning techniques allowed the system to capture complex patterns and dependencies in the data, further improving the accuracy of recommendations.

The system's performance was evaluated using a real-world dataset, and it outperformed existing methods in terms of recommendation accuracy, diversity, and novelty. This project not only provided a robust and scalable solution for personalized movie recommendations but also contributed to the broader field of recommendation systems.

Future work on this project could explore the integration of more contextual data, such as user demographics or time of viewing, to further enhance the personalization of recommendations. Additionally, exploring other machine learning techniques or algorithms could potentially improve the system's performance.

FUTURE ENHANCEMENT

1. *Enhanced Recommendation Algorithm*: Continuously refine and improve the recommendation algorithm. Experiment with different approaches such as collaborative filtering, content-based filtering, or hybrid methods to provide more accurate and personalized recommendations based on user preferences, viewing history, and other relevant factors.

2. *Integration of Machine Learning*: Implement machine learning techniques to analyze user behavior, preferences, and interactions with the platform. This could involve using natural language processing (NLP) to analyze reviews and feedback, sentiment analysis to understand user reactions, and reinforcement learning to optimize the recommendation strategy over time.

3. *Dynamic Banner Content*: Instead of just displaying a static set of banners on the homepage, consider making the banners dynamic and personalized based on the user's interests and viewing history. Show banners featuring recommended movies or genres that are likely to appeal to the user, increasing the chances of engagement.

4. *Interactive Homepage Features*: Introduce interactive elements on the homepage such as sliders, carousels, or personalized recommendations directly embedded within the page layout. Allow users to interact with the content, rate movies, add them to their watchlist, or provide feedback, enabling a more immersive and engaging user experience.

5. *Social Integration*: Incorporate social features to allow users to connect with friends, share movie recommendations, and see what their friends are watching or recommending. Implement social login options, friend lists, and activity feeds to foster community engagement and enhance the social aspect of the platform.

6. *Cross-Platform Compatibility*: Extend the recommendation system to work seamlessly across multiple platforms and devices, including web browsers, mobile apps, smart TVs, and streaming devices. Ensure consistent user experience and synchronization of preferences and viewing history across different devices.

7. *Personalized Content Discovery*: Develop advanced features for content discovery, such as personalized movie collections, curated lists, thematic recommendations, and seasonal promotions. Use machine learning to understand user preferences and offer tailored content suggestions that match their interests and mood.

8. *Integration with External APIs*: Explore integration with external APIs and data sources to enrich the platform with additional movie metadata, reviews, ratings, and trending topics. This could include integrating with services like IMDb, Rotten Tomatoes, or The Movie Database to provide comprehensive information and enhance the quality of recommendations.

9. *User Feedback and Iterative Improvement*: Collect user feedback through surveys, ratings, and reviews to understand their preferences, pain points, and suggestions for improvement. Use this feedback to iterate on the recommendation system, address user needs, and prioritize feature development accordingly.

10. *Monetization Strategies*: Explore monetization opportunities such as subscription plans, premium content, targeted advertising, or partnerships with content providers. Implement a flexible monetization model that balances user experience with revenue generation, ensuring sustainable growth and profitability.

By incorporating these enhancements and continually iterating on your movie recommendation system, you can create a compelling and personalized entertainment platform that delights users and keeps them engaged for the long term.