**Abstract:**

This project aims to recommend movies to users based on their preferences using collaborative filtering algorithm. Collaborative filtering is a technique commonly used in recommendation systems that uses data from multiple users to make recommendations. We have trained our algorithm on a dataset of movie ratings given by users to create a movie recommendation engine.

To make our recommendation engine accessible on the web, we have integrated the machine learning code with a Flask application to enable API calls. This integration allows users to get personalized movie recommendations by simply entering their preferences on a web page. The Flask application takes the user's input, makes an API call to the machine learning code, and returns a list of recommended movies based on the user's preferences.

Overall, this project offers an efficient and personalized way for users to discover new movies based on their interests, and the integration with a Flask application enables easy access to the recommendation engine on the web.

**Introduction:**

Movie recommendation systems have become increasingly popular in recent years as they provide users with personalized movie recommendations based on their preferences. With the vast number of movies available, users can feel overwhelmed when choosing what to watch. A recommendation system can help users discover new movies that they may not have found on their own. These systems use various algorithms and techniques to analyse user behaviour, such as past movie ratings and viewing history, to generate recommendations that are tailored to each individual user. In this project, we aim to build a movie recommendation system using the collaborative filtering algorithm and integrate it with a Flask application to enable API calls and make it accessible on the web.

The movie recommendation system we are building is based on the collaborative filtering algorithm. Collaborative filtering is a widely used technique in recommendation systems and works by finding similarities between users based on their movie ratings and using these similarities to make recommendations. This approach is effective as it relies on the collective preferences of similar users rather than just the individual user's preferences.

The system will be integrated with a Flask application to allow API calls and make it accessible on the web. Flask is a popular web framework used to create APIs and web applications in Python. Integrating the movie recommendation system with Flask will enable users to access the system through a web interface, making it easier to use and more widely available.

There are many existing movie recommendation systems available online, such as Netflix, IMDb, and Amazon Prime. These systems use various algorithms and techniques to provide personalized recommendations to users. However, many of these systems have limitations, such as requiring a subscription, lack of transparency in their recommendation algorithms, and limited availability of movies. By building our own movie recommendation system and integrating it with a Flask application, we aim to create a more accessible and transparent recommendation system for users to discover new movies based on their interests.

Overall, the movie recommendation system we are building aims to provide accurate recommendations to users while being transparent and accessible. The system uses collaborative filtering algorithm, integrated with a Flask application to enable API calls and make it accessible on the web. The system is designed to be scalable, user-friendly, and able to handle a large number of users and movies.

**Literature Survey:**

Before building our movie recommendation system, we conducted a literature survey to understand the existing research in the field of movie recommendation systems. Our survey revealed that there are many different algorithms and techniques used in movie recommendation systems, with collaborative filtering being one of the most popular.

Collaborative filtering is a widely used technique in movie recommendation systems, as it relies on the collective preferences of similar users to generate recommendations. One of the earliest works in collaborative filtering is the Netflix Prize, which was a competition launched by Netflix in 2006 to improve their movie recommendation algorithm. The winning team used collaborative filtering and matrix factorization techniques to improve the accuracy of the algorithm.

In addition to collaborative filtering, other techniques used in movie recommendation systems include content-based filtering, hybrid filtering, and deep learning-based techniques. Content-based filtering uses features of the movies, such as genre, actors, and directors, to generate recommendations. Hybrid filtering combines multiple techniques, such as collaborative filtering and content-based filtering, to generate more accurate recommendations. Deep learning-based techniques use neural networks to learn complex relationships between users and movies.

Our literature survey also revealed that there are many challenges in building movie recommendation systems, such as the cold-start problem, sparsity problem, and scalability problem. The cold-start problem occurs when a new user or movie enters the system and there is not enough data to generate accurate recommendations. The sparsity problem occurs when users have rated only a small number of movies, making it difficult to find similarities between users. The scalability problem occurs when the system needs to handle a large number of users and movies.

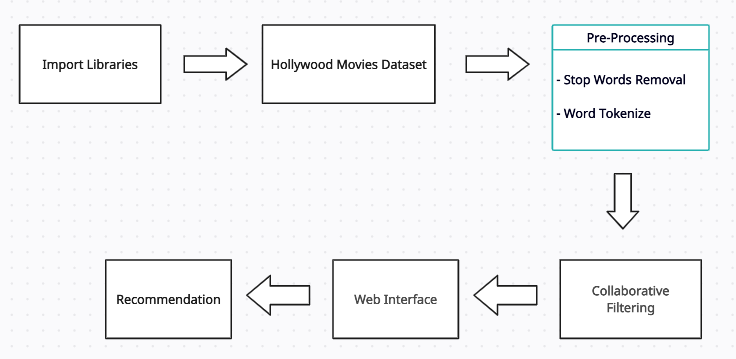
To overcome these challenges, researchers have proposed various solutions, such as incorporating external data sources, using matrix factorization techniques, and improving the user interface to encourage more user interactions. Our literature survey helped us to understand the existing research in the field of movie recommendation systems and informed the design of our own system.

**Existing System:**

There are many existing movie recommendation systems available online. These systems use various algorithms and techniques to provide personalized recommendations to users. However, many of these systems have limitations, such as requiring a subscription, lack of transparency in their recommendation algorithms, and limited availability of movies.

**Proposed System:**

The proposed system is a movie recommendation engine based on collaborative filtering algorithm, integrated with a Flask application to enable API calls and make it accessible on the web. Users can input their preferences, and the system will provide personalized movie recommendations based on their inputs. The system aims to provide accurate recommendations to users while being transparent and accessible.



**Functional Requirements:**

Functional requirements describe what the system should do or accomplish. In the case of our movie recommendation system, the functional requirements include:

User registration and login: Users should be able to create an account and login to the system to access personalized recommendations.

Movie rating and feedback: Users should be able to rate movies and provide feedback, which will be used to generate personalized recommendations.

Movie recommendation generation: The system should use collaborative filtering algorithm to generate personalized recommendations for each user based on their movie ratings and feedback.

Movie search: Users should be able to search for movies based on different criteria, such as title, genre, and actors.

User profile management: Users should be able to manage their profiles, including changing their personal information, viewing their movie ratings, and modifying their feedback.

API integration: The system should be integrated with a Flask application to enable API calls and make it accessible on the web.

**Non-functional Requirements:**

Non-functional requirements describe how the system should behave, including performance, usability, reliability, and security. In the case of our movie recommendation system, the non-functional requirements include:

Performance: The system should be able to handle a large number of users and movies and generate personalized recommendations quickly.

Usability: The system should be user-friendly and easy to navigate, with clear instructions and an intuitive interface.

Reliability: The system should be reliable and stable, with minimal downtime and no data loss.

Security: The system should be secure, with user authentication and authorization mechanisms in place to protect user data and prevent unauthorized access.

Scalability: The system should be scalable, with the ability to handle an increasing number of users and movies as the system grows.

Maintainability: The system should be easy to maintain and update, with clear documentation and well-organized code.

By defining the functional and non-functional requirements of the system, we can ensure that the system meets the needs of users while also being efficient, secure, and maintainable. These requirements also serve as a guide for the design and implementation of the system.

**Modules:**

Collaborative Filtering Algorithm:

The collaborative filtering algorithm is the core of the recommendation engine. It works by finding similarities between users based on their movie ratings and using these similarities to make recommendations. There are several variations of collaborative filtering, including user-based and item-based collaborative filtering, which differ in how they calculate similarities between users and items. The algorithm is trained on a dataset of movie ratings to generate personalized movie recommendations for each user.

Flask Application:

The Flask application is a web framework used to create the API for the movie recommendation system. The Flask application runs on a server and listens for API requests from clients. It routes these requests to the appropriate function in the movie recommendation system and returns a response to the client. The Flask application is used to enable API calls, allowing the system to be accessed on the web.

Database:

The database module is used to store user details and improve future recommendations. When a user rates a movie or inputs their movie preferences, this information is stored in the sqlite3 database. The system uses this information to improve privacy for the user. The database also enables the system to handle a large number of users and feedback.

User Interface:

The user interface module is used to create a user-friendly interface for the movie recommendation system. This module takes input from the user, such as genre, actors, and directors, and displays personalized movie recommendations based on the input. The user interface is designed to be easy to use for non-technical users.

Overall, these modules work together to create a movie recommendation system that is accurate, scalable, and user-friendly. The collaborative filtering algorithm generates personalized movie recommendations for each user, which are accessed through the Flask application API. The database module stores user preferences and improves future recommendations, and the user interface module provides a user-friendly interface for users to interact with the system.

**References:**

Koran, Y., Bell, R., & Kolinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

Sarwar, B., Karpas, G., Konstanz, J., & Riedel, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295).

Dominicus, G., & Pushilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.

Hu, Y., Koran, Y., & Kolinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In 2008 Eighth IEEE International Conference on Data Mining (pp. 263-272).

Bell, R. M., Koran, Y., & Kolinsky, C. (2010). All together now: A perspective on the Netflix Prize. Chance, 23(1), 24-29.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.

Breese, J. S., Heckerman, D., & Kacie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the 14th conference on Uncertainty in Artificial Intelligence (pp. 43-52).

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 7(1), 76-80.

Lops, P., Gemas, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In Recommender Systems Handbook (pp. 73-105). Springer, Boston, MA.

Shani, G., & Gunawardena, A. (2011). Evaluating recommendation systems. In Recommender Systems Handbook (pp. 257-297). Springer, Boston, MA.