**Final Project Report**

**Movie Recommendation System**



**Project Supervisor**

N/A

**Submitted By**

N/A

**Software Projects & Research Section,**

**Department of Computer Sciences,**

**Virtual University of Pakistan**

|  |
| --- |



**CERTIFICATE**

This is to certify that N/A, have worked on and completed their Software Project at Software & Research Projects Section, Department of Computer Sciences, Virtual University of Pakistan in partial fulfillment of the requirement for the degree of BS in Computer Sciences under my guidance and supervision.

In our opinion, it is satisfactory and up to the mark and therefore fulfills the requirements of BS in Computer Sciences.

**Supervisor / Internal Examiner**

<<Project Supervisor Name>>

Supervisor,

Software Projects & Research Section,

Department of Computer Sciences

Virtual University of Pakistan

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature)

**External Examiner/Subject Specialist**

<<External Supervisor Name>>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Signature)

**Accepted By:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_**

(For office use)

**EXORDIUM**

**In the name of Allah, the Compassionate, the Merciful.**

**Praise be to Allah, Lord of Creation,**

**The Compassionate, the Merciful,**

**King of Judgment-day!**

**You alone we worship, and to You alone we pray for help,**

**Guide us to the straight path**

**The path of those who You have favored,**

**Not of those who have incurred Your wrath,**

**Nor of those who have gone astray.**

**DEDICATION**

* I dedicate this project to all the movie enthusiasts and technology aficionados who believe in the power of innovation and entertainment. Your passion for films and your curiosity in the world of algorithms and data science have been a constant source of inspiration throughout the journey of creating this Movie Recommendation System.
* I am immensely grateful to my family and friends for their unwavering support and encouragement. Your belief in me and my abilities has fueled my determination to bring this project to fruition.
* A special thanks goes to my project advisor, [N/A], for their invaluable guidance and insights. Your expertise and mentorship have been instrumental in shaping the direction of this work.
* Last but not least, I extend my heartfelt appreciation to the open-source community for providing the tools and resources that made this project possible. Your collective efforts have paved the way for aspiring developers like me to explore and innovate.
* This project is a testament to the convergence of creativity, technology, and the love for cinema. May it serve as a small contribution to the world of personalized movie experiences.

**ACKNOWLEDGEMENT**

* I would like to express my heartfelt gratitude to all those who have contributed to the successful completion of my final project on the Movie Recommendation System.
* First and foremost, I am immensely thankful to my project supervisor [N/A], for their, insightful feedback in Prototype Phase Viva. Their expertise and encouragement have been invaluable in shaping the direction of this work.
* I am also indebted to [VU WhatsApp groups] for providing me with the necessary resources and environment to carry out this research. The data and references needed for the project.
* I extend my sincere appreciation to the OpenAI team for developing the GPT-3.5 language model, which played a pivotal role in enhancing the intelligence and capabilities of my recommendation system. The insights and assistance provided by ChatGPT were instrumental in shaping the technical aspects of the system.
* I would like to thank my friends for their moral support, brainstorming sessions, and valuable discussions that helped me refine my ideas and overcome challenges.
* Last but not least, I am grateful to my family for their unwavering encouragement, patience, and understanding throughout this journey. Their constant belief in me motivated me to give my best.
* In conclusion, this project would not have been possible without the collective efforts, guidance, and inspiration from all those mentioned above. I am truly grateful for their contributions.

**PREFACE**

* In the modern digital landscape, the explosion of data and the ever-increasing availability of content have given rise to the need for personalized experiences. From streaming services to e-commerce platforms, the ability to provide users with tailored recommendations is a driving force behind user engagement and satisfaction. Collaborative filtering, a technique widely used in recommendation systems, empowers platforms to deliver precisely that.
* This document delves into the world of collaborative filtering, focusing specifically on the item-based approach and employing the powerful K-Nearest Neighbors (KNN) algorithm. Our aim is to unveil the art of constructing a recommendation system that suggests movies to users based on their preferences and the similarities between movies. This user-centric approach not only enhances the user experience but also aids in discovery, enabling users to stumble upon hidden gems that align with their tastes.
* Throughout this guide, we will navigate through the intricacies of building an item-based collaborative filtering recommendation system using KNN. Starting with the fundamental concepts of collaborative filtering, we will gradually lead you through data pre-processing, feature extraction, and the mechanics of KNN. We'll explore the concept of similarity metrics and unveil how they play a pivotal role in finding neighbors in the vast ocean of movies.
* Our journey will encompass transforming raw data into a structured format, calculating similarities between movies, and establishing a foundation for personalized recommendations. Moreover, we'll discuss the significance of model evaluation and ways to optimize performance.
* This guide is intended to be informative for both newcomers to recommendation systems and seasoned data scientists. By the end, you'll not only have a working understanding of building an item-based collaborative filtering movie recommendation system but also the tools to embark on more complex recommendation challenges. So, let's embark on this journey of crafting meaningful and personalized movie recommendations, tailored to the preferences of each unique user.

**TABLE OF CONTENTS**

|  |
| --- |

**CHAPTER NO. 1**

**GATHERING & ANALYZING INFO 10**

1.1 Introduction ……………………………………………………11 to 12

1.2 purpose ……………………………………………………13 to 14

1.3 scope ……………………………………………………15 to 16

1.4 definitions, acronyms and abbreviations …………………17 to 18

1.5 Project requirements ……………………………………….19 to 22

1.5.1 Functional Requirements

1.5.2 Non-Functional Requirements

1.6 use cases and usage scenarios ………………………….23 to 34

1.6.1 Use Case Diagrams

1.6.2 Usage Scenarios

1.7 Development Methodology …………………………………….34 to 36

1.7.1 Chosen Methodology

1.7.2 Reasons for Chosen Methodology

1.7.2 Work Plan (Gantt Chart)

1.7.2 Project Schedule (Submission Calendar)

**CHAPTER NO. 2**

**DESIGNING THE PROJECT 37**

2.1 Introduction ……………………………………………………38 to 41

2.2 purpose ……………………………………………………42 to 43

2.3 scope ……………………………………………………44 to 46

2.4 definitions, acronyms and abbreviations …………………47 to 48

2.5 Architectural Representation (Architecture Diagram) ……49

2.6 Dynamic Model: Sequence Diagrams ………50 to 56

2.7 Object Model/Logical Model: Class Diagram …………57

2.8 Database Model (Database Diagram) ………58

2.9 Graphical User Interfaces ………59

**CHAPTER NO.3**

**DEVELOPMENT 60**

3.1 Development plan (Architecture Diagram) ……………..61

**CHAPTER 1**

Gathering & Analyzing Info

# **1.1 Introduction:**

* In the world of data-driven personalization, recommendation systems have gained significant prominence. These systems aim to provide users with personalized suggestions, enhancing user experiences across various platforms. One powerful approach to building such systems is "item-based collaborative filtering" using the K-Nearest Neighbors (KNN) algorithm. This technique offers a robust way to recommend movies to users based on their similarity to items they have shown interest in.
* Item-Based Collaborative Filtering:
* Item-based collaborative filtering is a type of recommendation system that leverages the similarities between items themselves to make recommendations. Instead of focusing solely on user preferences, this method exploits item-item relationships within the user-item interaction matrix. By identifying items that are frequently interacted with together by users, the system can suggest items that are similar to those the user has already shown interest in.
* K-Nearest Neighbors (KNN):
* The K-Nearest Neighbors algorithm is a machine learning technique used for classification and regression tasks. In the context of recommendation systems, KNN is applied to identify the nearest neighbors of a target item based on a defined similarity metric. This involves finding items that are most similar to the target item, and subsequently recommending items that these "neighbors" have been positively rated by other users.
* Benefits of Item-Based Collaborative Filtering with KNN:
* No User Profiling Required: Unlike user-based collaborative filtering, which requires user profiles and their preferences, item-based methods can work without user-specific information.
* Transparency: Recommendations are made based on item similarities, which can be more intuitive and interpretable than complex user-based models.
* Cold Start Problem Mitigation: Item-based approaches tend to handle new or less popular items better, as they rely on item-to-item relationships.
* Conclusion:
* The item-based collaborative filtering recommendation system with K-Nearest Neighbors offers a powerful way to provide relevant and personalized movie recommendations to users. By analyzing the interactions between items and leveraging the similarity between them, this approach enhances user engagement and satisfaction by suggesting movies that align with the user's interests.

# **1.2 Purpose:**

* The purpose of an item-based collaborative filtering movie recommendation system with K-Nearest Neighbors (KNN) is to provide personalized movie recommendations to users based on the similarity between movies. This approach leverages the historical preferences of users to identify similar movies and make recommendations. Here's an explanation of the purpose and benefits of using item-based collaborative filtering with KNN:
* Personalized Recommendations: The system aims to offer movie recommendations tailored to individual users' preferences. By analyzing users' historical movie ratings and comparing them to the ratings of other movies, the system identifies movies that are similar in terms of user preferences.
* Item Similarity: Item-based collaborative filtering focuses on identifying similarities between movies themselves. This is achieved by calculating a similarity score between each pair of movies based on user ratings. Movies that have similar ratings by users are considered to be related or similar.
* No User Profiles Required: Unlike user-based collaborative filtering, which requires creating user profiles, item-based collaborative filtering relies solely on the movie ratings matrix. This can be advantageous in scenarios where user data is sparse or when new users join the system.
* Effective Handling of Cold Start Problem: The item-based approach is particularly effective in addressing the "cold start" problem, where there is limited or no user history available. Since the focus is on similarities between items, new movies can still be recommended based on their attributes and the preferences of users who have rated them.
* Robustness to Sparsity: Item-based collaborative filtering tends to perform well even when the user-item interaction matrix is sparse (many missing entries). It can leverage similarities between movies to provide meaningful recommendations.
* Scalability: The item-based collaborative filtering with KNN is relatively computationally efficient and can handle large datasets. It involves calculating similarities between items and selecting the top-k neighbors, making it scalable for systems with a substantial number of movies.
* Transparency: The recommendations generated by item-based collaborative filtering are often transparent and interpretable. The system can provide explanations for recommendations by showing users the similar movies that led to a specific recommendation.
* Diversity in Recommendations: Since the system identifies similar movies, it can introduce diversity in recommendations by suggesting movies that are similar but not identical to the ones the user has already seen.
* In summary, an item-based collaborative filtering movie recommendation system with KNN aims to provide accurate and personalized movie recommendations by leveraging the similarities between movies based on user ratings. This approach is well-suited for addressing the cold start problem and can work effectively with sparse data, making it a valuable tool in personalized recommendation systems.

# **1.3 Scope:**

* The scope for an item-based collaborative filtering movie recommendation system using K-Nearest Neighbors (KNN) is significant, as it addresses a common challenge in the field of recommendation systems: providing personalized and relevant content to users. Here are some key aspects that highlight the scope and benefits of such a system:
* Personalized Recommendations: Item-based collaborative filtering with KNN allows you to recommend movies to users based on their preferences and the preferences of similar users. This personalization can significantly enhance the user experience by suggesting movies that align with individual tastes.
* Utilizing Item Similarities: In an item-based approach, movies with similar user ratings and characteristics are grouped together. KNN leverages these item-item similarities to recommend movies that are liked by users who enjoyed similar movies, leading to more accurate and relevant recommendations.
* Cold-Start Problem: Item-based collaborative filtering is robust in handling the "cold-start" problem, which occurs when new movies are added to the system and don't have sufficient ratings. The system can still recommend these new movies based on their similarities to existing movies.
* Scalability: Item-based collaborative filtering tends to be more scalable than user-based approaches. Calculating item-item similarities and maintaining the similarity matrix is often more feasible than dealing with user-user similarities.
* Sparse Data: Recommender systems often work with sparse data, where not all users have rated all movies. Item-based collaborative filtering handles sparsity well because it focuses on similarities between items rather than relying solely on user preferences.
* Easy Interpretation: The recommendations generated by an item-based collaborative filtering system are often easier to interpret compared to other methods. Users can understand why certain movies are recommended based on the movies they have liked.
* Long Tail Recommendations: Item-based systems can efficiently provide recommendations for niche or less popular movies, also known as the "long tail" of content. This encourages users to explore a wider range of movies.
* Adaptability: The system can adapt to changes in user preferences over time. If a user's tastes evolve, the system will reflect those changes in its recommendations.
* User Engagement: A well-implemented recommendation system can increase user engagement and satisfaction by suggesting movies that resonate with users' interests, leading to longer platform usage and higher user retention.
* Customization and Enhancement: You can enhance the recommendation system with additional features, such as incorporating movie metadata (genres, actors, directors) for richer recommendations.
* In summary, building an item-based collaborative filtering movie recommendation system using KNN offers the potential to provide personalized and relevant movie recommendations, tackle the cold-start problem, handle sparse data, and contribute to user engagement and satisfaction. This scope makes it a valuable tool for platforms aiming to enhance user experiences and promote content discovery.

# **1.4 Definitions, Acronyms and Abbreviations:**

* Definitions:
* Item-Based Collaborative Filtering: A recommendation system technique that recommends items (movies in this case) to users based on the similarity between items. It uses the ratings or preferences of users to find items that are similar to the ones a user has already liked or interacted with.
* K-Nearest Neighbors (KNN): A machine learning algorithm used for both classification and regression tasks. In the context of recommendation systems, KNN is used to find the nearest neighbors (similar items or users) to a given item or user based on certain features or attributes.
* Recommendation System: A system that suggests items to users, typically based on their preferences, historical interactions, or similarities between items or users.
* Acronyms and Abbreviations:
* CF: Collaborative Filtering
* RS: Recommendation System
* IBCF: Item-Based Collaborative Filtering
* KNN: K-Nearest Neighbors
* SIM: Similarity
* MUR: Movies Users Rating
* CORR\_MATRIX: A correlation matrix is a table showing the correlation coefficients between multiple variables (take values between -1 and +1, where -1 represents a perfect negative correlation, +1 represents a perfect positive correlation, and 0 represents no correlation.)
* NN: Nearest Neighbors (a general term referring to the concept of finding nearby items or users)

# **1.5 Project requirements**

1.5.1 Functional Requirements

* + Data Collection and Preprocessing:
    - Gather movie data, including information such as titles, genres, and ratings.
    - Collect user rating data, which should include user IDs, movie IDs, and corresponding ratings.
    - Preprocess the data to handle missing values, duplicates, and inconsistencies.
    - Ensure data is in a structured format suitable for analysis.
  + Exploratory Data Analysis (EDA):
    - Analyze and visualize the distribution of movie ratings.
    - Explore the distribution of user ratings and the number of ratings per movie.
  + Item-Based Collaborative Filtering:
    - Understand the concept of item-based collaborative filtering and its advantages.
    - Choose a similarity metric (e.g., cosine similarity) to measure the similarity between movie items.
  + Feature Extraction:
    - Create a user-movie rating matrix with movies as rows and users as columns.
  + K-Nearest Neighbors (KNN):
    - Understand how KNN works and its relevance to collaborative filtering.
    - Choose the value of 'k' (number of neighbors) based on experimentation or cross-validation.
  + Building the Recommendation System:
    - Implement the KNN algorithm to find similar movies based on item-item similarity.
    - Calculate weighted average ratings for user-specific recommendations.
    - Sort and recommend movies with the highest predicted ratings.
  + Evaluation and Testing:
    - Split the dataset into training and testing sets.
    - Evaluate the recommendation system's performance using metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE).
    - Fine-tune the model parameters to achieve better results.
  + User Interface:
    - Develop a user-friendly interface where users can input a movie and receive recommendations.
    - Display recommended movies along with their details such as titles, genres, and predicted ratings.
  + Documentation and Reporting:
    - Document the project's architecture, algorithms, and methodologies.
    - Describe the dataset, preprocessing steps, and feature extraction techniques used.
    - Explain how the KNN-based recommendation system works and its limitations.
    - Present evaluation results and discuss the system's performance.
  + Presentation and Communication:
    - Prepare a presentation to communicate the project's objectives, methods, results, and insights.
    - Demonstrate the recommendation system in action using real examples.
  + Deployment:
    - Deploy the recommendation system as a web application or API for users to access and interact with.

# **1.5.2 Non-Functional Requirements**

* + Non-functional requirements define the quality attributes and constraints of a system rather than its specific functionalities here are some non-functional requirements to consider:
  + Performance:
    - Response Time: The recommendation system should provide recommendations within an acceptable time frame, ensuring a quick response to user requests.
    - Scalability: The system should be able to handle an increasing number of users, movies, and ratings without significantly compromising performance.
  + Accuracy:
    - Prediction Accuracy: The recommendations provided by the system should be accurate and relevant to user preferences. The system should aim to minimize false positives (incorrect recommendations) and false negatives (missing relevant recommendations).
  + Robustness:
    - Handling Sparse Data: The system should be capable of providing recommendations even when the user-item interaction matrix is sparse (many missing values).
    - Graceful Degradation: The system should gracefully handle errors, unexpected inputs, or system failures, ensuring that it doesn't crash or provide incorrect results.
  + Privacy and Security:
    - User Privacy: The system should adhere to privacy regulations and protect user data. Personal user information should be anonymized and safeguarded.
    - Secure Communication: Any communication between the system and users should be encrypted to prevent data breaches.
  + Usability:
    - User Interface: The user interface should be intuitive and easy to navigate, allowing users to provide feedback, refine preferences, or override recommendations.
    - Personalization: The system should provide users with options to customize their preferences and refine recommendations according to their interests.
    - Maintainability:
    - Modularity: The system's components should be well-organized and modular, allowing for easier maintenance and updates.
    - Documentation: The system's code and documentation should be well-documented to assist future development, troubleshooting, and enhancement.
  + Adaptability:
    - Parameter Tuning: The system should allow administrators to fine-tune KNN parameters, such as the number of neighbors, to optimize performance and accuracy.
    - Integration: The recommendation system should be designed to integrate with other systems, databases, or APIs as needed.
  + Resource Utilization:
    - Memory Usage: The system should use memory efficiently to store and process user-item interaction data, especially for larger datasets.
    - Processing Power: The system should utilize available processing power efficiently to generate recommendations in a timely manner.
  + Feedback and Monitoring:
    - Feedback Mechanism: The system should provide users with mechanisms to rate or provide feedback on the recommendations, helping to improve accuracy over time.
    - Monitoring: The system should have monitoring tools to track performance, identify issues, and make continuous improvements.
  + Cultural Sensitivity:
    - Diversity: The recommendation system should ensure diversity in recommendations, considering different genres, cultures, and preferences to avoid unintentional bias.

# **1.6 Use Cases and Usage Scenarios:**

# **1.6.1 Use Case Diagrams**

# 

# **1.6.2 Usage Scenarios**

| **Use Case Title** | **Enter Movie Name** |
| --- | --- |
| **Use Case ID** | UC-EMN-00 |
| **Actions** | User visits the recommendation system's webpage.  User is presented with a search bar labeled "Enter Movie Name".  User types the name of a movie they are interested in.  User clicks the "Search" button. |
| **Description** | The user interacts with the system to input the name of a movie they want recommendations for. |
| **Alternative Paths** | If the user does not provide a valid movie name or provides an incorrect name, the system displays an error message and prompts the user to enter a valid movie name. |
| **Pre-Conditions** | The user has accessed the recommendation system's webpage. |
| **Post-Conditions** | The user's input is recorded and ready to be processed by the system. |
| **Author** | System User |
| **Exceptions** | If the user's input is not valid or the movie name is not recognized, an error message is displayed. |

| **Use Case Title** | **View/Access Recommendations** |
| --- | --- |
| **Use Case ID** | UC-VR-01 |
| **Actions** | The system displays a list of movie recommendations |
| **Description** | The user can view the recommended movies based on his previous interests and the similarities with other users. The recommendations are displayed in a list format and can be sorted by different criteria such as genre, release date, etc. |
| **Alternative Paths** | N/A |
| **Pre-Conditions** | The system must have generated recommendations for the user. |
| **Post-Conditions** | The user can view the recommended movies. |
| **Author** | System User |
| **Exceptions** | If there is a technical error in the system, the user may not be able to view the recommendations.  If the user's account is deactivated, they will not be able to view the recommendations. |

| **Use Case Title** | **Define and Select the Dataset(s)** |
| --- | --- |
| **Use Case ID** | UC-DSD-02 |
| **Actions** | Identify the problem of providing movie recommendations to users based on their previous interests.  Research and evaluate different datasets of movies that can be used to train the model.  Select the appropriate/given dataset(s) of movies for the project. |
| **Description** | The purpose of this use case is to define the problem that the project will address and to select the dataset(s) of movies that will be used to train the model. The system will identify the problem as providing movie recommendations to users based on their previous interests. The system will then research and evaluate different datasets of movies that can be used to train the model. Once the appropriate/given dataset(s) of movies has been selected, the project can proceed to the next phase of development. |
| **Alternative Paths** | If there is not a suitable dataset available, the system may have to create a new dataset by scraping movie data from various sources.  If multiple datasets are found to be appropriate, the system may choose to combine them for better results. |
| **Pre-Conditions** | The system must have access to internet to research and evaluate different datasets of movies. |
| **Post-Conditions** | The problem of providing movie recommendations to users based on their previous interests is defined.  The appropriate/given dataset(s) of movies is selected for the project. |
| **Author** | BC190200432 |
| **Exceptions** | If the system encounters an error while researching and evaluating different datasets of movies, an exception will be raised.  If the system is unable to find a suitable dataset, then alternative options will be discussed. |

| **Use Case Title** | **Data Analysis & Pre-processing** |
| --- | --- |
| **Use Case ID** | UC-DAP-03 |
| **Actions** | Collect and import movie ratings and user preferences data from various sources.  Clean and preprocess the data to remove any missing or irrelevant information.  Perform statistical analysis on the data to identify patterns and trends.  Use data visualization techniques to represent the findings in a clear and concise manner. |
| **Description** | The system performs data analysis on the movie ratings and user preferences to find patterns and trends. This information is then used to develop a user model and generate recommendations. |
| **Alternative Paths** | If there is a lack of data or the data is incomplete, the system may not be able to perform accurate analysis. In this case, additional data may need to be collected or the scope of the analysis may need to be limited.  If the data is too large or complex to be processed by the system, it may be necessary to use advanced techniques such as machine learning or big data processing tools. |
| **Pre-Conditions** | The system has access to a dataset of movie ratings and user preferences. |
| **Post-Conditions** | The system has analyzed the data and identified patterns and trends, which can be used to develop a user model and generate recommendations. |
| **Author** | BC190200432 |
| **Exceptions** | If the dataset is not formatted properly or is missing critical information, the system may not be able to perform the analysis.  If there are technical issues with the system, it may not be able to process the data and the analysis may not be completed. |

| **Use Case Title** | Data Science (EDA) |
| --- | --- |
| **Use Case ID** | UC-EDA-03 |
| **Actions** | Load the dataset into a data analysis tool (e.g., Jupyter Notebook, Python script).  Examine the structure of the dataset using functions like .head(), .info(), and .describe().  Visualize data distributions with histograms, box plots, and scatter plots.  Identify missing values and handle them through imputation or removal.  Calculate summary statistics to understand central tendencies and variabilities.  Perform data cleaning by addressing outliers, inconsistencies, and errors.  Create correlation matrices or heatmaps to identify relationships between variables.  Plot graphs to explore trends, patterns, and anomalies.  Generate insights about data characteristics and initial hypotheses. |
| **Description** | Exploratory Data Analysis (EDA) involves visually and statistically exploring datasets to understand their structure, patterns, and relationships. It helps data scientists gain insights, identify data quality issues, and formulate initial hypotheses for further analysis. |
| **Alternative Paths** | If missing values are extensive, impute with median instead of mean.  Use different visualization types based on data characteristics (e.g., bar charts, line plots). |
| **Pre-Conditions** | Obtain the dataset for analysis.  Install necessary libraries (e.g., pandas, matplotlib, seaborn) for data manipulation and visualization. |
| **Post-Conditions** | Gain a deeper understanding of data distribution, relationships, and potential issues.  Formulate initial insights and hypotheses for further analysis. |
| **Author** | BC190200432 |
| **Exceptions** | If the dataset is too large, sampling may be necessary for efficient analysis.  Incomplete data may lead to biased insights.  Inaccurate data cleaning may distort analysis results. |

| **Use Case Title** | **Feature Extraction** |
| --- | --- |
| **Use Case ID** | UC-FE-04 |
| **Actions** | Identify the features to be extracted from the movie dataset  Extract the selected features from the dataset  Clean and pre-process the extracted features  Store the extracted and pre-processed features in a format suitable for further analysis |
| **Description** | Feature extraction is the process of identifying and extracting relevant information from the movie dataset that will be used to develop the user model and make recommendations. This includes identifying which features are important, such as movie genre, release year, and rating, and then extracting and cleaning those features so that they can be used for analysis. |
| **Alternative Paths** | N/A |
| **Pre-Conditions** | The movie dataset must be available and accessible to the system |
| **Post-Conditions** | The extracted and pre-processed features are stored in a format suitable for further analysis |
| **Author** | BC190200432 |
| **Exceptions** | N/A |

| **Use Case Title** | **Develop User Model** |
| --- | --- |
| **Use Case ID** | UC-DUM-05 |
| **Actions** | Collect and analyze user data such as age, location, and movie ratings  Create a user model using machine learning algorithms  Validate the user model using a sample dataset  Improve the user model by incorporating feedback and additional data |
| **Description** | The system uses the collected user data to develop a user model that represents the users in the dataset. The user model is created using machine learning algorithms such as collaborative filtering or neural networks. The model is then validated using a sample dataset and improved by incorporating feedback and additional data. |
| **Alternative Paths** | If the user data is insufficient, the system may need to collect additional data to improve the accuracy of the user model.  If the user model is not accurate, the system may need to try different machine learning algorithms or incorporate additional data to improve the model. |
| **Pre-Conditions** | User data and machine learning algorithms are available |
| **Post-Conditions** | A user model is created and validated |
| **Author** | BC190200432 |
| **Exceptions** | If the system is unable to collect sufficient user data, the user model may not be accurate.  If the system is unable to develop a user model using the available machine learning algorithms, the system may need to try different algorithms or incorporate additional data. |

| **Use Case Title** | **Predict Recommendations** |
| --- | --- |
| **Use Case ID** | UC-PR-07 |
| **Actions** | Analyzing the user model and similarities with other users  Identifying movies that the user has not yet rated but have been positively rated by similar users  Generating a list of recommendations for the user based on this analysis |
| **Description** | Given a user's preferences and ratings, and a dataset of similar users and their ratings, generate a list of movie recommendations for the user based on the preferences and ratings of similar users. |
| **Alternative Paths** | if the user has not provided sufficient ratings or information, the system will not be able to generate accurate recommendations  if there are no similar users in the dataset, the system will not be able to generate recommendations |
| **Pre-Conditions** | User model and similarities with other users have been developed  User has provided some ratings or preferences  A dataset of similar users and their ratings is available |
| **Post-Conditions** | A list of movie recommendations for the user is generated |
| **Author** | BC190200432 |
| **Exceptions** | Insufficient user data or no similar users in the dataset, causing the system to not be able to generate recommendations  Technical errors or bugs in the system that prevent it from generating recommendations. |

| **Use Case Title** | **Build System** |
| --- | --- |
| **Use Case ID** | UC-BS-08 |
| **Actions** | Gather all necessary code, libraries, and dependencies for the movie recommendation system.  Assemble the code and libraries to create the functional system.  Test the system for any bugs or issues.  Deploy the system for use. |
| **Description** | The Build System use case is responsible for creating a functional and deployable version of the movie recommendation system. This includes gathering all necessary components, assembling them, and testing the system for any bugs or issues. Once the system is deemed stable, it can be deployed for use. |
| **Alternative Paths** | If bugs or issues are found during testing, they will need to be addressed before the system can be deployed.  If there are missing dependencies or libraries, they will need to be acquired before the system can be built. |
| **Pre-Conditions** | All necessary code and libraries must be available.  The system must pass all tests without errors. |
| **Post-Conditions** | The movie recommendation system is deployed for use.  Any bugs or issues are addressed. |
| **Author** | BC190200432 |
| **Exceptions** | If there are missing dependencies or libraries, the system cannot be built until they are acquired.  If the system fails testing, it cannot be deployed until the issues are resolved. |

| **Use Case Title** | **Access Movie Recommendations Webpage** |
| --- | --- |
| **Use Case ID** | UC-BS-08.1 |
| **Actions** | User opens a web browser.  User enters the URL of the deployed movie recommendation webpage.  The webpage loads, displaying the homepage. |
| **Description** | Users can access the movie recommendation webpage by entering the URL in their web browser. The webpage serves as the user interface for the recommendation system. |
| **Alternative Paths** | If bugs or issues are found during testing, they will need to be addressed before the system can be deployed.  If there are missing dependencies or libraries, they will need to be acquired before the system can be built. |
| **Pre-Conditions** | If the entered URL is invalid or the webpage is down, an error message is displayed.  If the user's internet connection is not available, the webpage will not load. |
| **Post-Conditions** | The movie recommendation system has been deployed using Django/Flask.  The user has a device with a web browser and an internet connection. |
| **Author** | BC190200432 |
| **Exceptions** | Invalid URL entered: The webpage displays an error message.  Webpage is down: The user cannot access the webpage.  No internet connection: The webpage cannot load. |

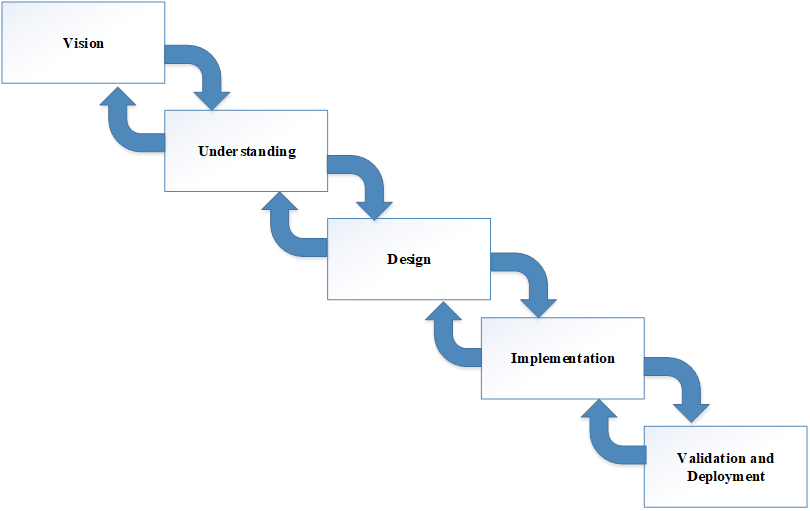
| **Use Case Title** | **Test System** |
| --- | --- |
| **Use Case ID** | UC-TS-09 |
| **Actions** | The system is prepared for testing by setting up test data and environments.  The system is run through a series of tests to check for bugs, errors, and issues.  Any issues or bugs found during testing are documented and reported to the development team.  The system is retested after any necessary fixes or updates have been made. |
| **Description** | The Test System use case is used to ensure that the movie recommendation system is functioning properly and meets the requirements before it is deployed. This includes testing for bugs, errors, and other issues that may affect the performance or usability of the system. |
| **Alternative Paths** | If no issues are found during testing, the system is ready for deployment.  If issues are found, we makes necessary changes and retests the system. |
| **Pre-Conditions** | The system has been built and is ready for testing. |
| **Post-Conditions** | The system has been tested and any issues have been addressed before deployment. |
| **Author** | BC190200432 |
| **Exceptions** | If the test data or environments are not set up properly, the test results may not be accurate.  If a critical issue is found during testing that cannot be resolved, the system may not be able to be deployed. |

| **Use Case Title** | **Tune System** |
| --- | --- |
| **Use Case ID** | UC-TS-010 |
| **Actions** | Analyze system performance metrics  Identify areas for improvement  Implement and test system optimizations  Repeat steps 1-3 until desired performance is achieved |
| **Description** | The Tune System use case is used to improve the performance of the movie recommendation system. This includes analyzing system metrics such as accuracy, speed, and scalability, identifying areas of improvement, and implementing and testing optimizations until the desired level of performance is achieved. |
| **Alternative Paths** | If there are no areas for improvement identified, the use case will end.  If the implemented optimizations do not improve the system's performance, alternative optimization methods will be considered. |
| **Pre-Conditions** | The movie recommendation system has been built and deployed.  The system has been tested and any bugs have been fixed. |
| **Post-Conditions** | The movie recommendation system's performance has been improved.  The system's performance metrics are within the desired range. |
| **Author** | BC190200432 |
| **Exceptions** | If the system's performance metrics are already within the desired range, the use case will not be executed.  If there are technical limitations that prevent the system from being optimized, the use case will not be executed. |

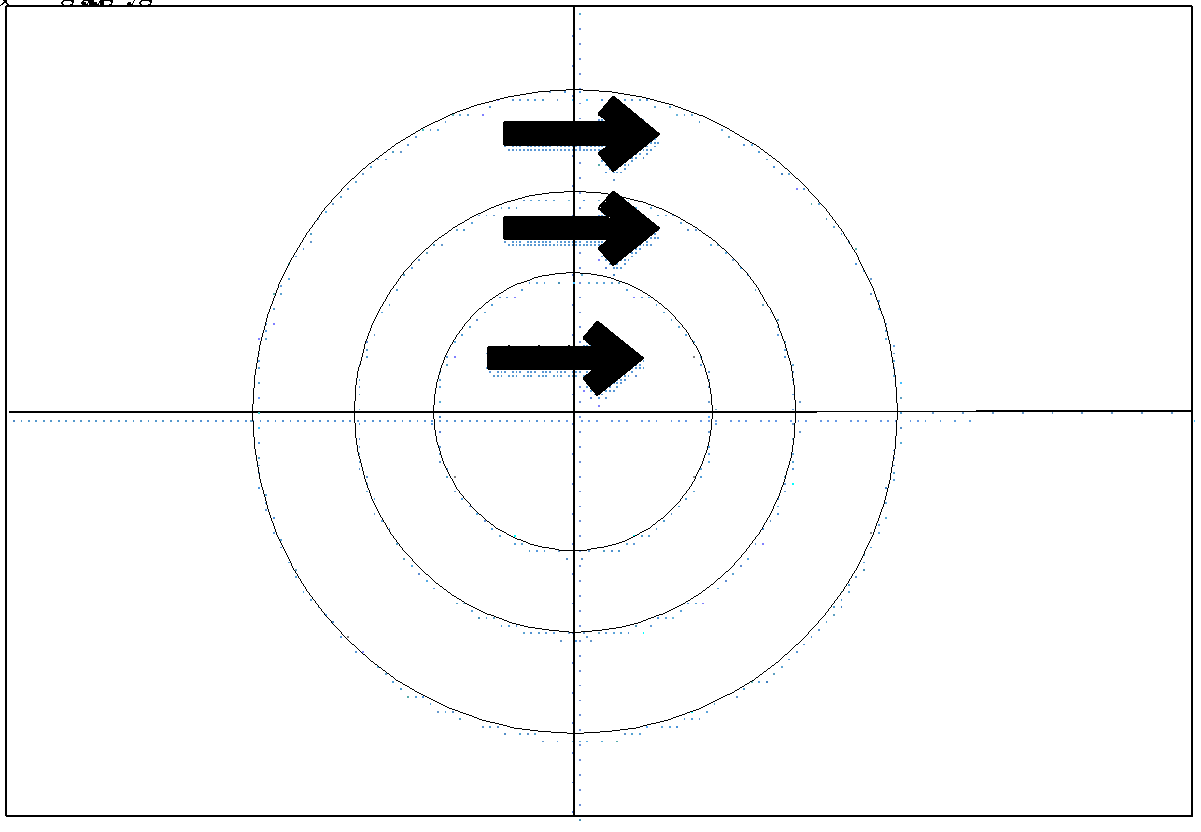
# **1.7 Development Methodology**

# 

# **1.7.1 Chosen Methodology**



# 



* The VU Process Model is an iterative process, meaning that it loops back to the understanding phase after each step. This allows for flexibility and adaptability in case new requirements are identified or problems arise during the development process.
* It is important to note that this methodology is not a one size fit all, it depends on the project, the team, the stakeholders, and the complexity of the problem you are solving. It's always a good idea to evaluate different methodologies and choose the one that best fits the project's needs.

# **1.7.2 Reasons for Chosen Methodology**

* We must choose VU Process Model (specifically asked) which is a combination of waterfall and spiral models. More or combining two or more methodologies is explained in SE2 (CS605) handouts.

# **1.7.2 Work Plan (Gantt Chart)**

# 

# **1.7.2 Project Schedule (Submission Calendar)**



**CHAPTER 2**

Designing the Project

# **2.1 Introduction:**

* Design Document is a comprehensive document that outlines the design of a software system. It describes the architecture, components, interfaces, and data for the system, and provides a blueprint for the development, testing, and maintenance of the software.
* Designing a project involves planning, organizing, and outlining the various aspects and components of a project before its actual implementation. A well-designed project lays the foundation for successful execution, efficient resource utilization, and achieving desired outcomes. In the context of data science or any technical project, the design phase is essential to ensure clarity, feasibility, and alignment with project goals. Here's an introduction to designing a project:

1. Project Overview:

* + - Begin by providing a concise overview of the project. Describe the problem statement, the objectives you aim to achieve, and the significance of the project in the broader context.

2. Project Scope:

* + - Clearly define the boundaries and limitations of your project. Outline what will be included in the project and what will be excluded. This helps manage expectations and prevent scope creep.

3. Project Goals and Objectives:

* + - Specify the specific goals and objectives of the project. What do you intend to accomplish through this project? Set clear, measurable, and achievable targets.

4. Project Deliverables:

* + - List the tangible outputs or outcomes that will be produced by the project. This could include reports, analyses, models, visualizations, software, or any other artifacts.

5. Stakeholders and Roles:

* + - Identify the key stakeholders involved in the project. Define their roles, responsibilities, and expectations. This ensures clear communication and accountability.

6. Project Timeline:

* + - Create a timeline that outlines the project phases, milestones, and deadlines. A timeline helps in tracking progress and managing time effectively.

7. Data Collection and Preparation:

* + - Detail the sources of data you will use for the project. Describe the methods you'll use to collect, clean, preprocess, and transform the data for analysis.

8. Exploratory Data Analysis (EDA):

* + - Mention how you'll explore the data to gain insights, identify patterns, and understand the characteristics of the data.

9. Methodology and Techniques:

* + - Explain the methodologies, algorithms, or techniques you plan to use to address the problem. Justify why these approaches are suitable for solving the problem.

10. Feature Selection/Extraction:

* + - Discuss how you'll select or extract relevant features from the data for analysis or modeling. Explain the rationale behind your choices.

11. Model Building:

* + - Describe how you'll build, train, and validate models if applicable. Detail the metrics you'll use to evaluate model performance.

12. Results and Interpretation:

* + - Outline how you'll present and interpret the results obtained from your analysis or models. Explain how the results address the project objectives.

13. Visualization and Reporting:

* + - Explain the visualizations and reports you'll create to communicate your findings effectively to non-technical stakeholders.

14. Resources and Constraints:

* + - Identify the resources required for the project, including data, tools, software, and human resources. Mention any potential constraints that could affect the project.

15. Risk Assessment and Mitigation:

* + - Highlight potential risks that could impact the project's success. Discuss strategies or contingency plans to mitigate these risks.

16. Ethical Considerations:

* + - Address any ethical considerations related to data privacy, bias, fairness, or any other relevant factors.

17. Conclusion:

* + - Summarize the key points of your project design and reiterate its importance in addressing the problem statement.
* A well-designed project plan acts as a roadmap for successful execution. It helps you stay organized, manage resources efficiently, and achieve the desired outcomes while aligning with the project's goals and objectives.
* **The benefits of this phase in a software development project include:**
  + - It helps to clearly define the requirements for the software and to ensure that all stakeholders have a common understanding of the system.
    - It provides a clear and detailed design for the development team to follow, which helps to reduce the risk of errors and misunderstandings.
    - It helps to identify potential issues and constraints early on in the development process, allowing for adjustments to be made before they become costly or time-consuming to fix.
    - It helps to ensure that the final product meets the needs of the users and stakeholders, and that it is maintainable and extensible in the future.
    - The purpose of this phase in a software development project is to provide a detailed and comprehensive design of the software system, which serves as a blueprint for the development, testing, and maintenance of the software. The Design Document serves as a guide for the development team and helps to ensure that the final product meets the needs of the users and stakeholders.

# **2.2 Purpose:**

* A Design Document, often referred to as a Technical Design Document (TDD) or System Design Document (SDD), is a comprehensive written document that outlines the technical specifications, architecture, and design considerations for a software project or system. The purpose of a Design Document is to provide a clear and detailed blueprint for how a software solution will be built, including its structure, components, interactions, and functionalities. Here are some key purposes of a Design Document:
* Guidance and Direction: A Design Document provides a roadmap for the development team, helping them understand how to implement the software solution effectively. It outlines the technical choices, approaches, and methodologies to be followed during the development process.
* Communication: The document serves as a means of communication between various stakeholders, including developers, architects, project managers, and clients. It helps ensure that all parties have a common understanding of the software's architecture and design.
* Documentation: A Design Document serves as a permanent record of the design decisions made during the software development process. It provides documentation that can be referred to in the future for maintenance, updates, and enhancements.
* Risk Mitigation: By detailing the architecture and design early in the project lifecycle, potential design flaws, scalability issues, and technical challenges can be identified and addressed before development begins. This helps mitigate risks associated with inadequate design.
* Resource Allocation: The document helps allocate resources efficiently by providing a clear plan for how different components of the software will interact and how resources such as memory, processing power, and storage will be utilized.
* Collaboration: Design Documents encourage collaboration among team members. Developers, architects, and other stakeholders can contribute their expertise and insights during the design phase, resulting in a more robust and well-rounded solution.
* Standardization: The document enforces standardization of coding practices, naming conventions, data structures, and other technical aspects. This consistency aids in maintaining code quality and making the software more maintainable.
* Scalability and Extensibility: A well-designed system can be more easily scaled and extended as requirements evolve. The Design Document outlines how the software's architecture supports these aspects.
* Evaluation and Review: The Design Document can be reviewed by technical experts and stakeholders to provide feedback, ensure alignment with business goals, and validate that the proposed solution meets the requirements.
* Training: The document can be used for training purposes, helping new team members understand the software's architecture, design choices, and technical implementation.
* In summary, a Design Document plays a crucial role in guiding the development process, ensuring technical coherence, and facilitating communication and collaboration among team members and stakeholders. It helps transform conceptual ideas into a well-structured and implementable software solution.

# **2.3 Scope:**

* A Design Document (also known as a Technical Design Document or System Design Document) is a comprehensive blueprint that outlines the architecture, components, and technical details of a software project or system. It provides a clear and structured plan for how the software will be developed, implemented, and tested. The scope of a Design Document encompasses several important aspects:

System Architecture:

* Describe the overall architecture of the software system. This includes the high-level structure, components, and their interactions. Discuss how different components will communicate and work together to achieve the system's goals.

Module/Component Design:

* Detail the design of individual modules or components that make up the system. Explain the functionality, data flow, input-output mechanisms, and interactions within each module. Define interfaces and relationships between modules.

Database Design:

* If the software involves a database, outline the database schema, tables, relationships, and data storage methods. Describe how data will be accessed, retrieved, updated, and maintained.

User Interface (UI) Design:

* Provide a description of the user interface components, layout, navigation, and interactions. Include wireframes, mockups, or diagrams to visually represent the user interface design.

Algorithms and Logic:

* If the software involves complex algorithms or logic, explain their design and functioning. Provide pseudocode or flowcharts to illustrate the step-by-step process of important algorithms.

Data Flows and Workflow:

* Document the flow of data and processes through the system. Describe how data moves between components, modules, and external systems, as well as how different workflows are handled.

Error Handling and Exception Management:

* Define how errors, exceptions, and edge cases will be handled. Describe error messages, logging mechanisms, and recovery strategies.

Security and Authentication:

* Outline security measures and mechanisms to ensure data integrity, user authentication, and authorization. Address potential vulnerabilities and describe how they will be mitigated.

Performance and Scalability:

* Discuss considerations for optimizing system performance and ensuring scalability. Detail strategies for handling increased user loads and resource demands.

Testing Strategy:

* Describe the testing approach, including unit testing, integration testing, and system testing. Explain how test cases will be designed, executed, and documented.

Dependencies and Third-Party Integrations:

* List any external dependencies, libraries, frameworks, or third-party integrations required for the project to function properly.

Deployment and Release Plan:

* Provide details about how the software will be deployed, including the deployment environment, installation instructions, and release timeline.

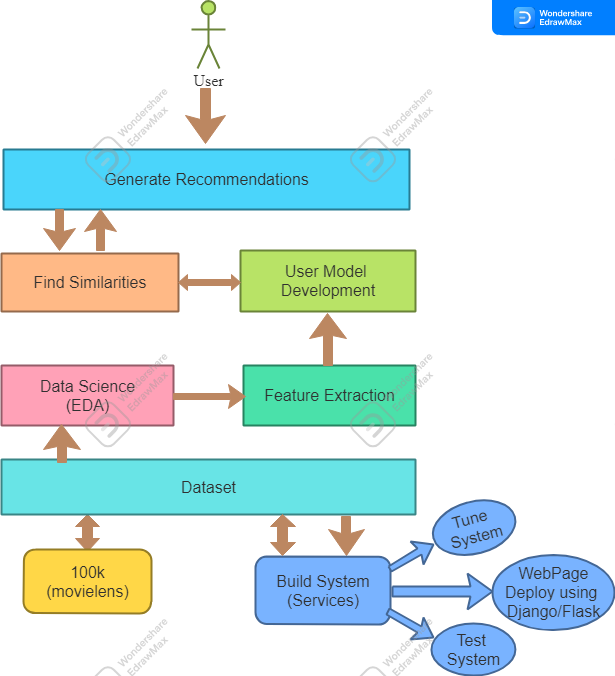
Maintenance and Support:

* Discuss plans for ongoing maintenance, bug fixes, updates, and user support.
* The scope of a Design Document is to provide a comprehensive reference for developers, testers, and stakeholders involved in the project. It ensures that all team members have a clear understanding of the system's architecture, design, and implementation details before development begins. This helps streamline the development process, improve collaboration, and mitigate risks during the software development lifecycle.

# **2.4 Definitions, Acronyms and Abbreviations:**

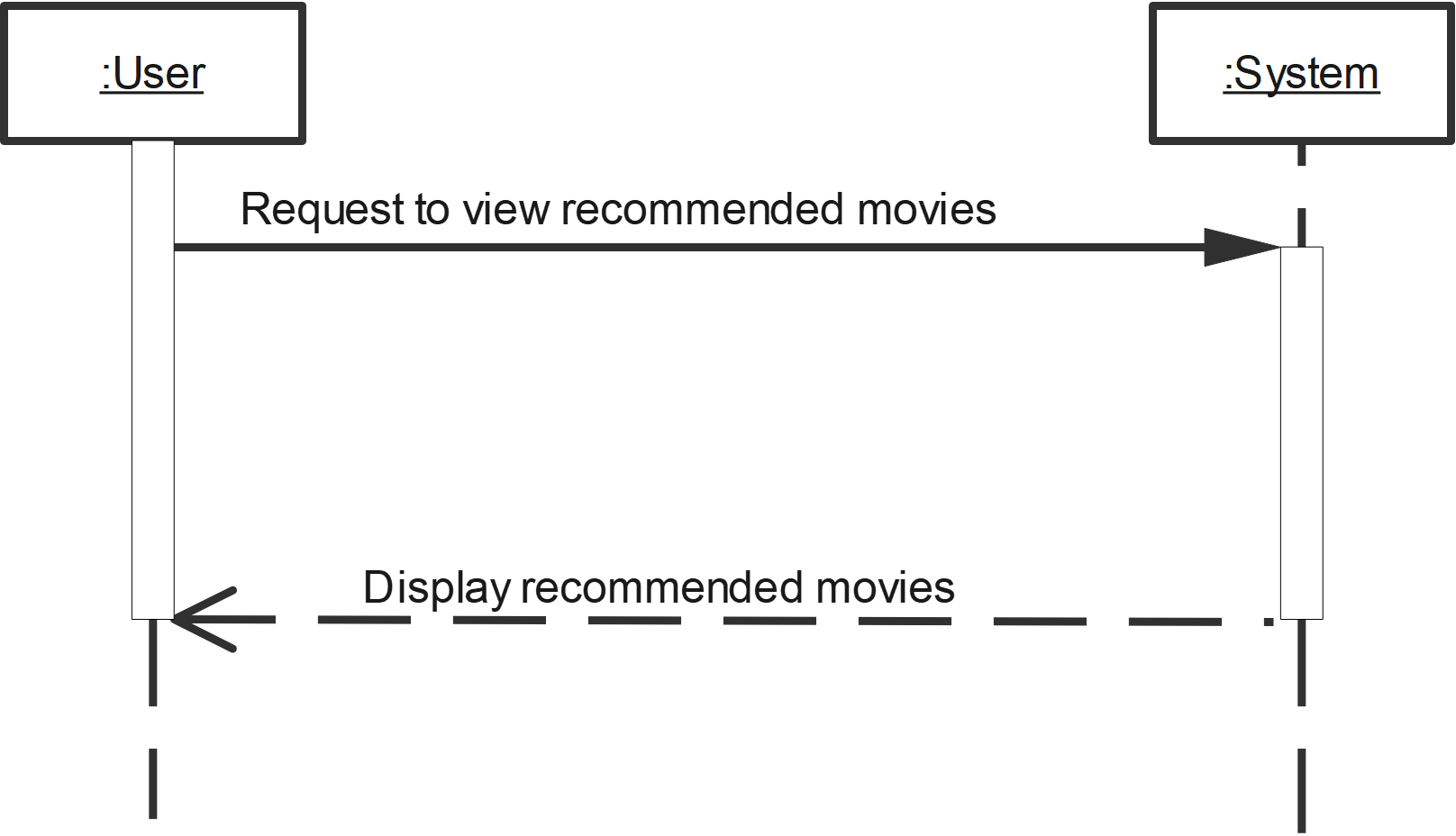
* + A Design Document, also known as a Design Specification or Technical Design Document, outlines the technical details and specifications of a software project or system that is going to be developed. It serves as a blueprint for the development team, providing a comprehensive understanding of the system's architecture, components, interactions, and implementation details. Here are some key definitions, acronyms, and abbreviations commonly used in a Design Document:
* Definitions:
  + - Design Document: A comprehensive technical document that outlines the architecture, components, and implementation details of a software system.
    - Architecture: The high-level structure and organization of a software system, including its components, interactions, and relationships.
    - Component: A modular and self-contained unit of a software system responsible for specific functionality.
    - Module: A smaller unit of a software system that encapsulates a specific set of features or functionality.
    - Interface: The boundary or contract through which different components or modules communicate with each other.
    - Dependency: A relationship between software components where one component relies on the functionality provided by another.
    - Data Flow: The movement of data between components or modules within a software system.
    - Algorithm: A step-by-step sequence of instructions used to perform a specific task or solve a problem.
* Acronyms and Abbreviations:
  + - API: Application Programming Interface
    - GUI: Graphical User Interface
    - DBMS: Database Management System
    - HTTP: Hypertext Transfer Protocol
    - UI: User Interface
    - UX: User Experience
    - ORM: Object-Relational Mapping
    - DTO: Data Transfer Object
    - CRUD: Create, Read, Update, Delete
    - REST: Representational State Transfer
    - SQL: Structured Query Language
    - JSON: JavaScript Object Notation
    - HTML: Hypertext Markup Language
    - CSS: Cascading Style Sheets
    - API: Application Programming Interface
    - URL: Uniform Resource Locator
    - SSL: Secure Sockets Layer
    - SRS: Software Requirements Specification
    - UML: Unified Modeling Language
    - MVC: Model-View-Controller
    - ORM: Object-Relational Mapping
* These definitions, acronyms, and abbreviations are commonly used in a Design Document to ensure clear and consistent communication among team members and stakeholders involved in the software development process.

# **2.5 Architectural Representation (Architecture Diagram):**

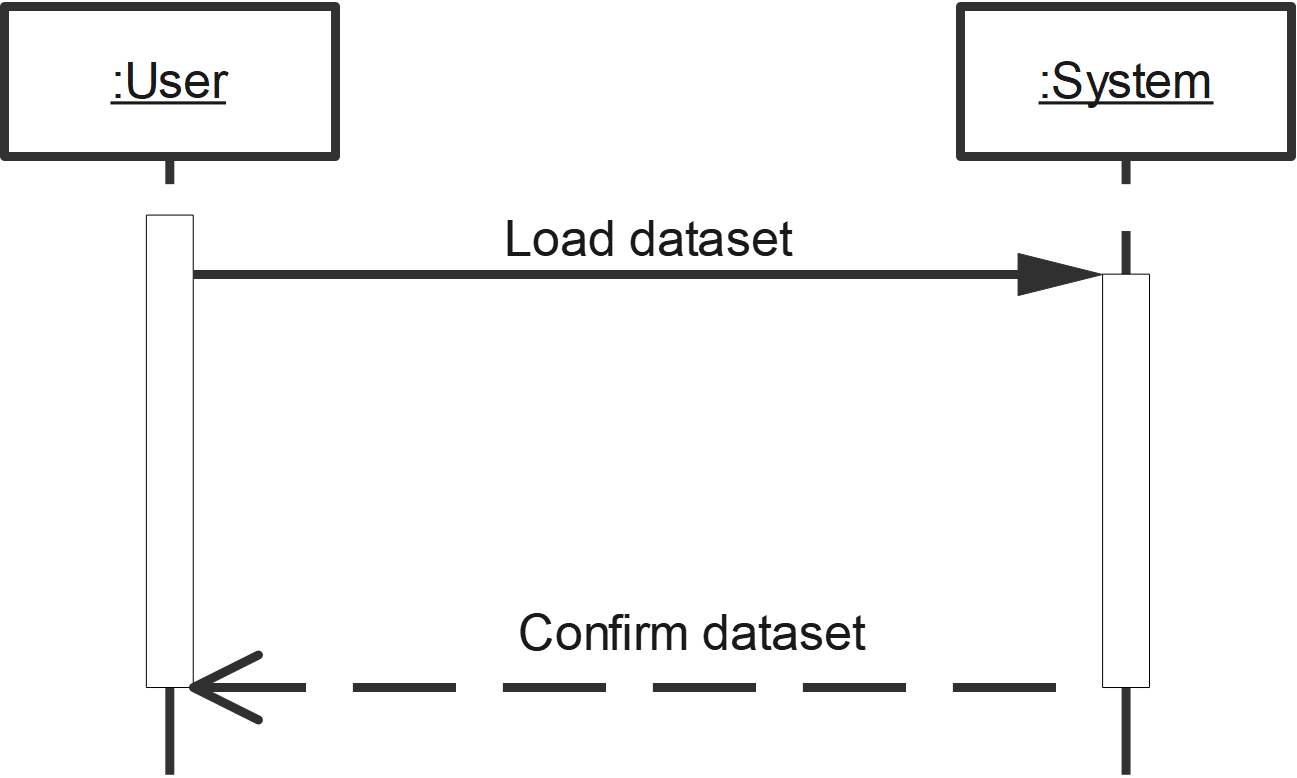


# **2.6 Dynamic Model: Sequence Diagrams:**

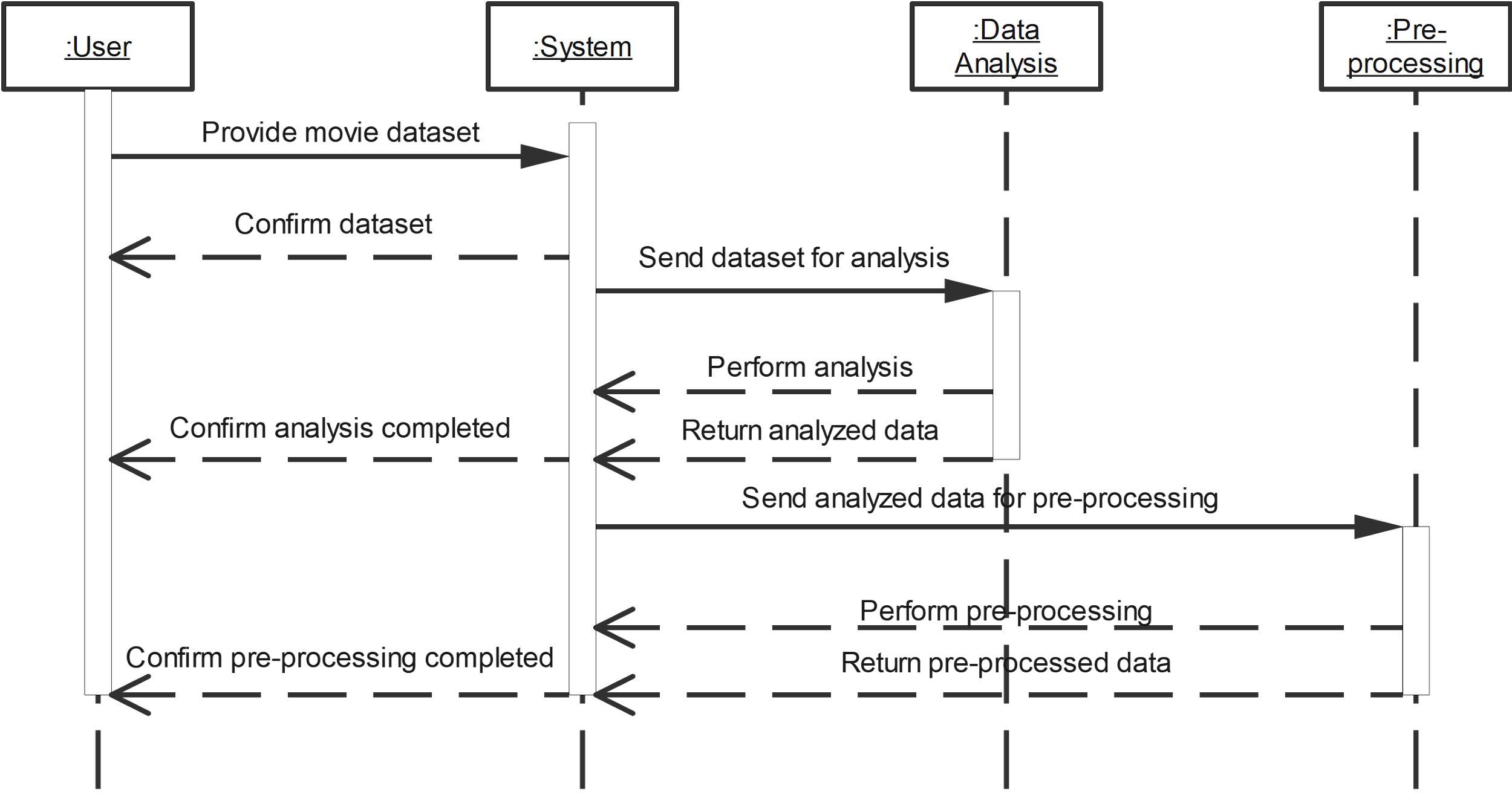
**Sequence Diagram 1: View/Access Recommendations**

****

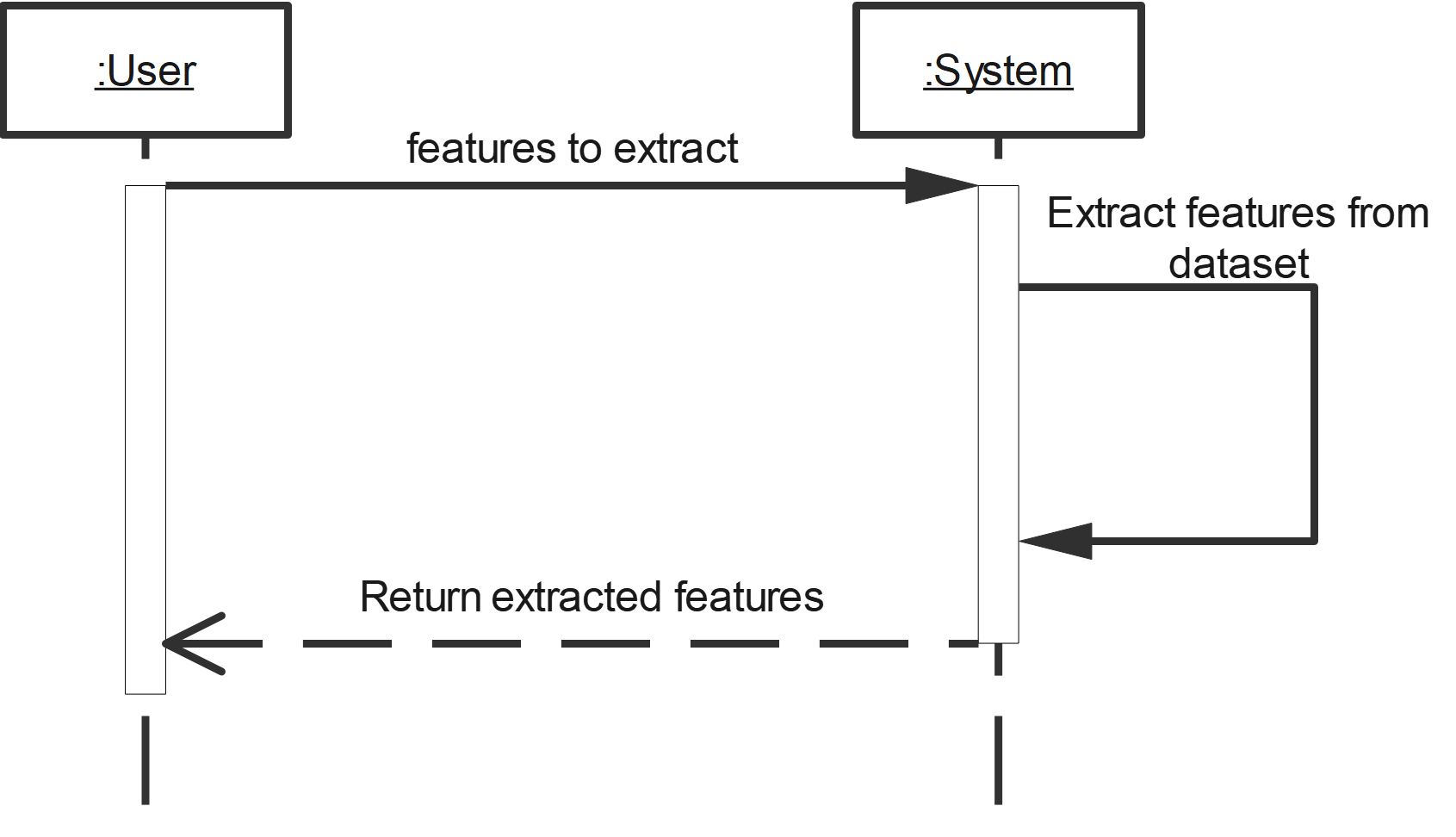
**Sequence Diagram 2: Dataset(s)**

****

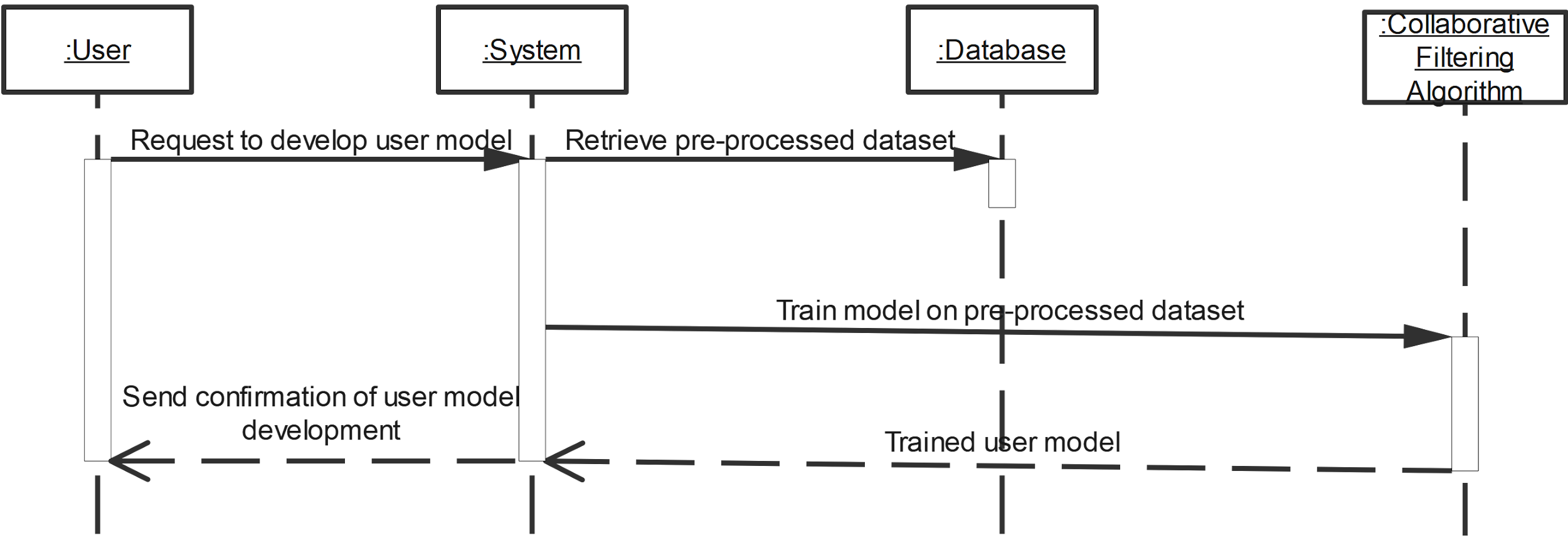
**Sequence Diagram 3: Data Science (EDA)**

****

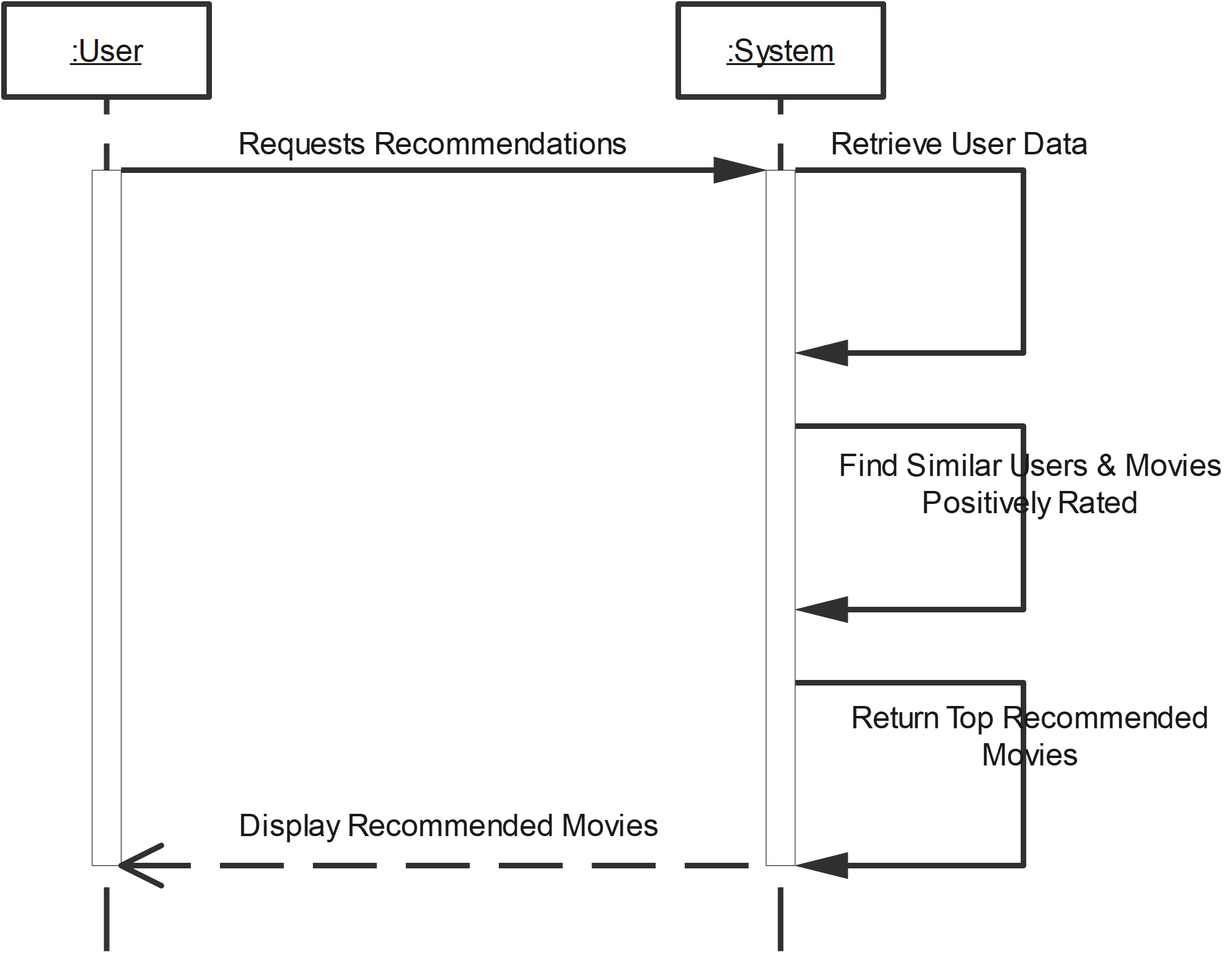
**Sequence Diagram 4: Feature Extraction**

****

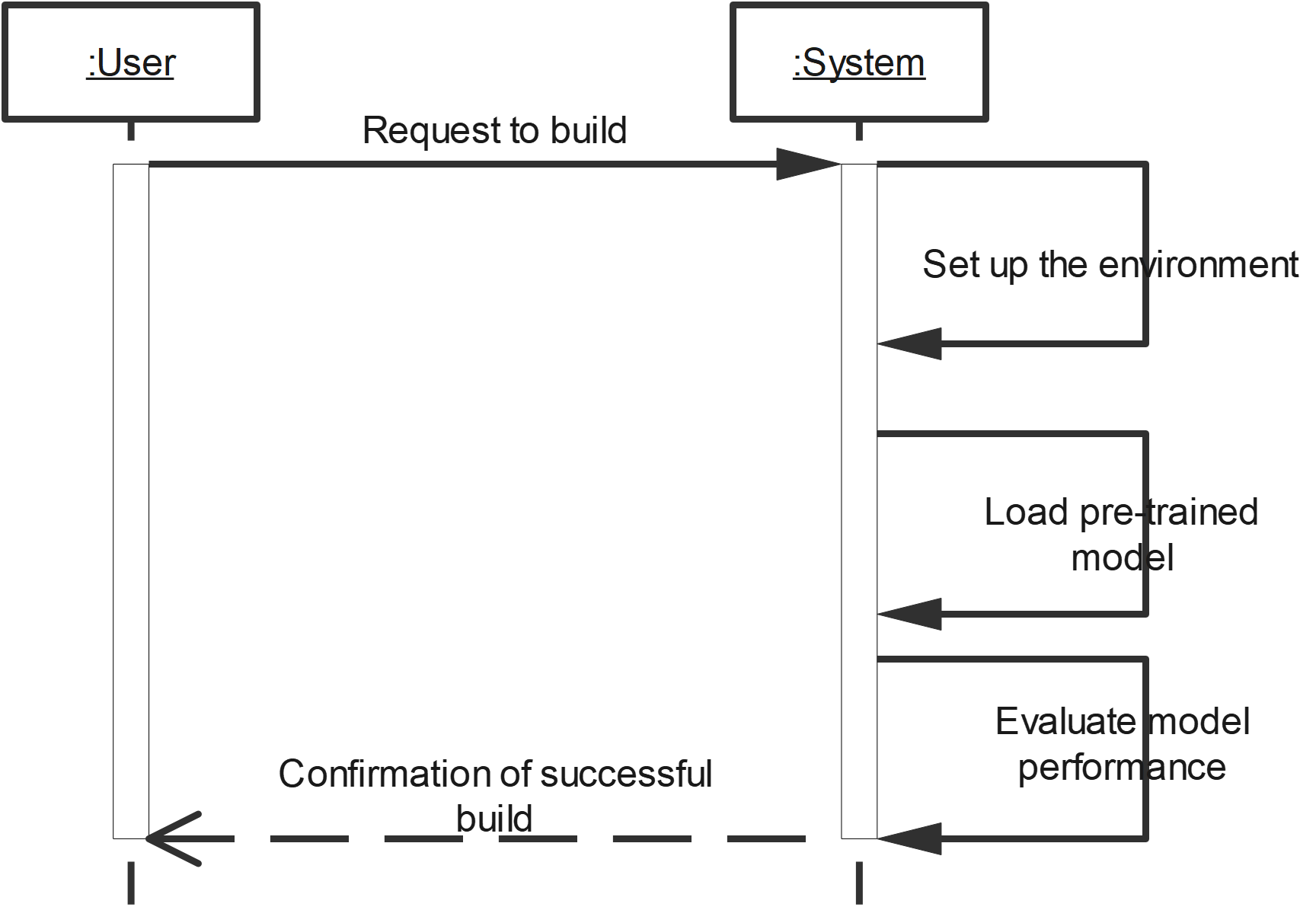
**Sequence Diagram 5: Develop User Model**

****

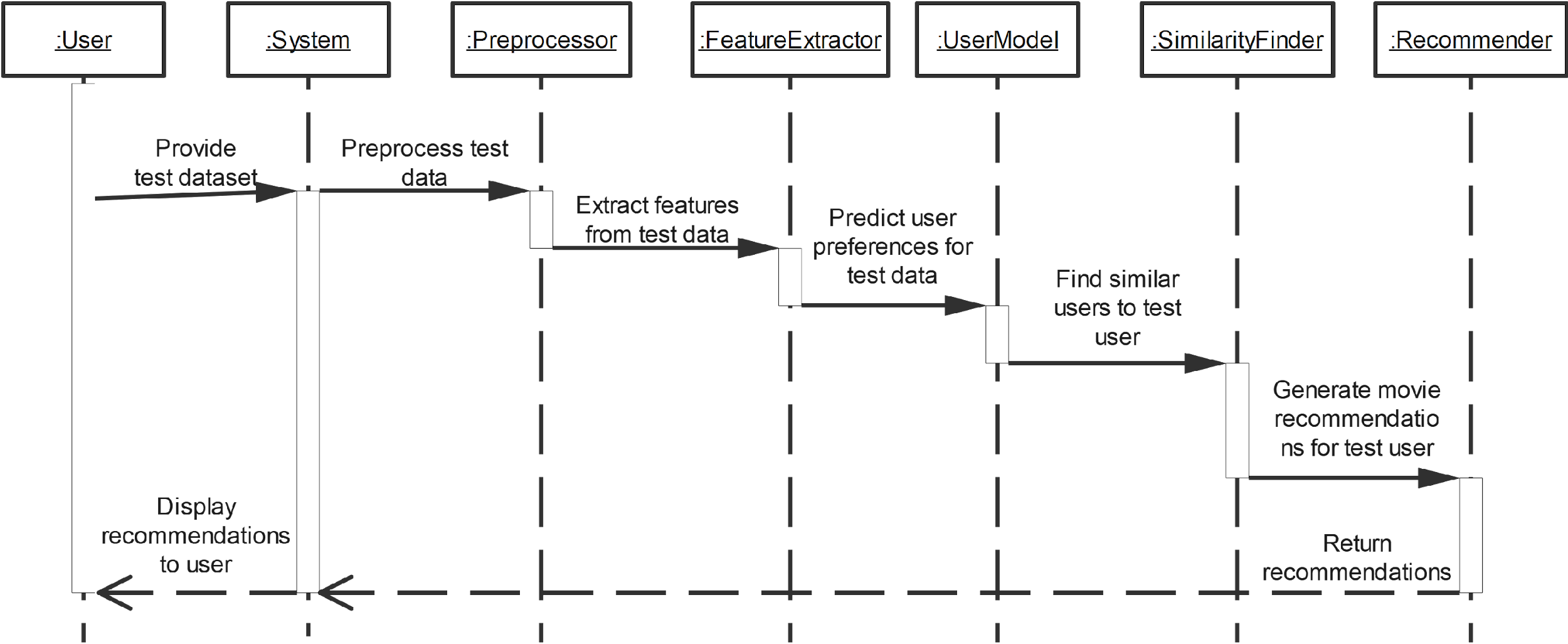
**Sequence Diagram 6: Predict Recommendations**

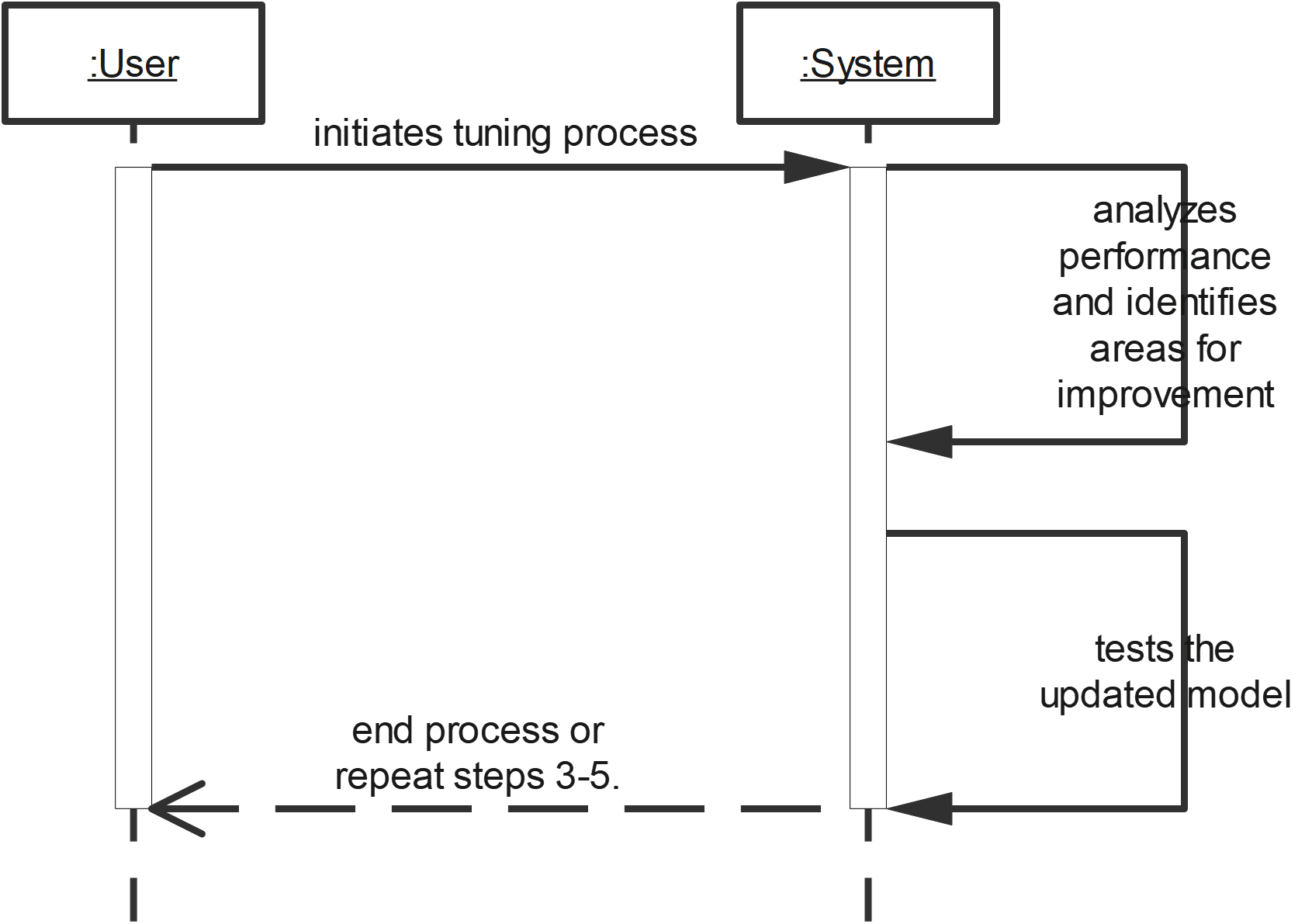
****

**Sequence Diagram 7: Build System**

****

**Sequence Diagram 8: Test System**

****

**Sequence Diagram 09: Tune System  
**

* **WebPage Deploy using Django/Flask Sequence Diagram: 010**

User --> Web Server: Access Web Page

Web Server --> User: Serve Web Page

User -> Web Server: Enter movie name

Web Server -> Recommendation System: Send movie name

Note right of Recommendation System: Backend Process

Recommendation System -> Recommendation System: Process movie name

Recommendation System -> Collaborative Filtering: Generate Recommendations

Collaborative Filtering -> Recommendation System: Return Recommendations

Recommendation System -> Web Server: Send Recommendations

Web Server --> User: Display Recommendations

* Enters movie name **Sequence Diagram: 011**

Actor: User

System: Movie Recommendation System

User -> System: Enters movie name

System -> System: Searches for movie name

System -> User: Displays search results

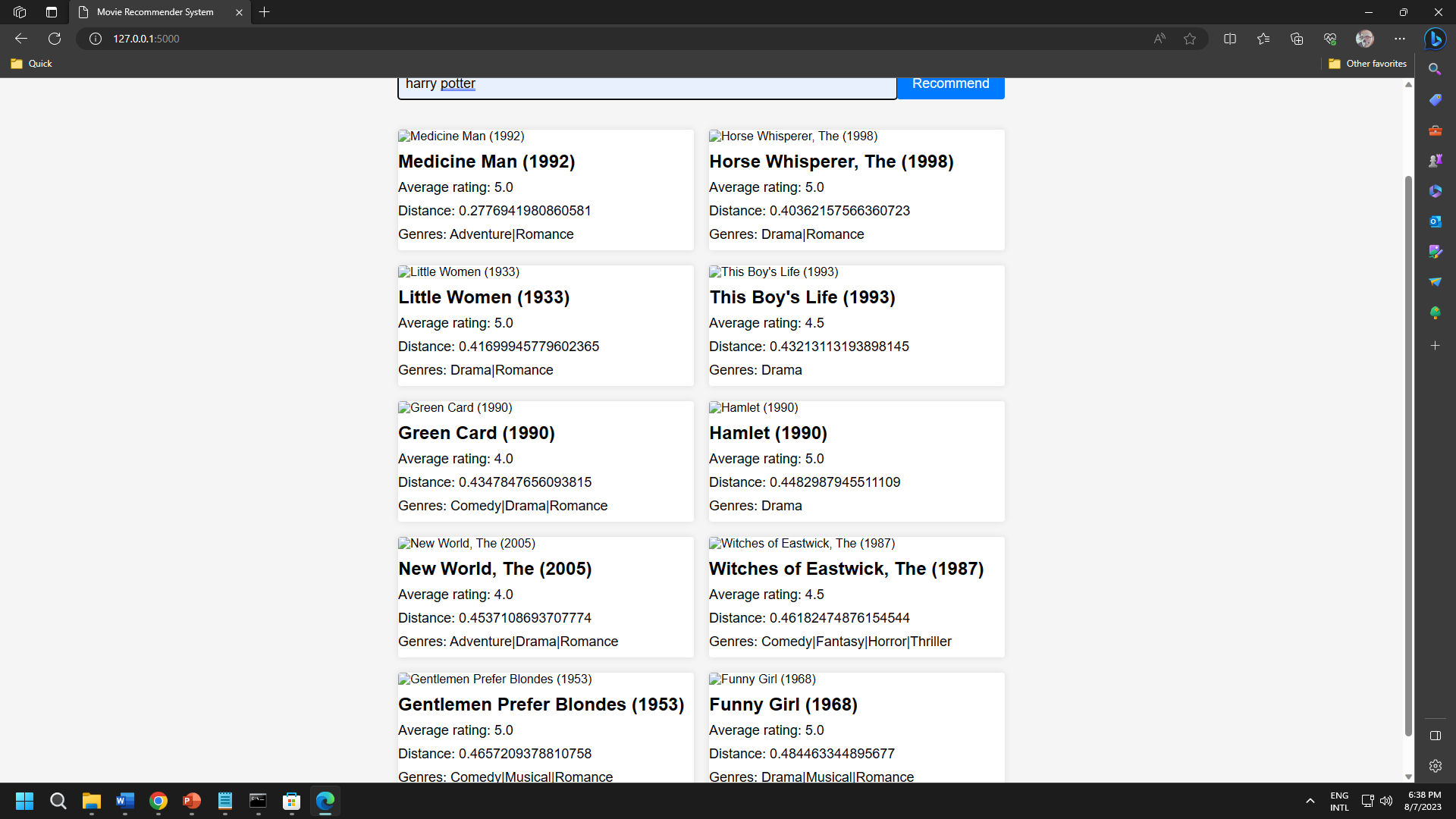
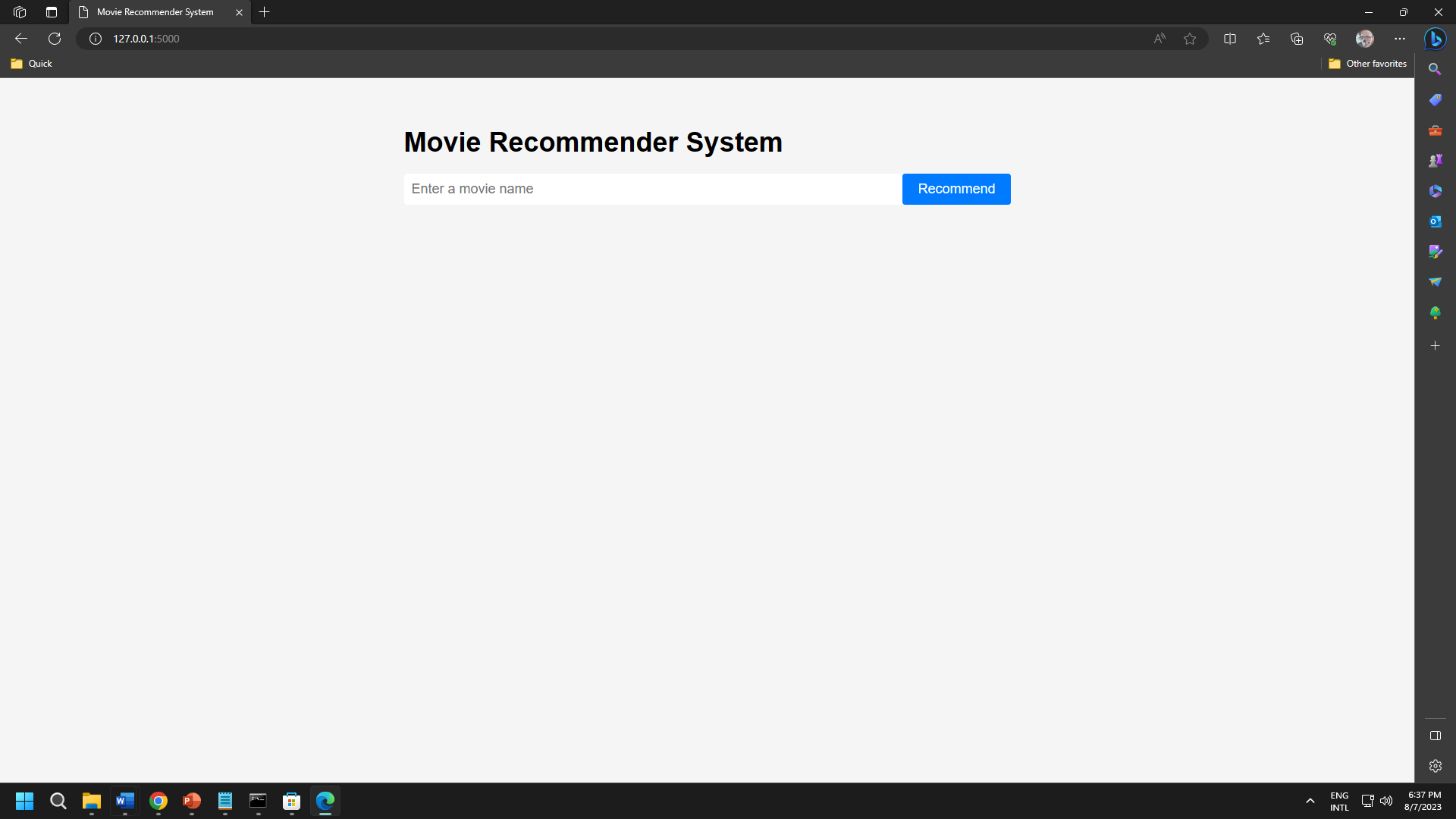
# **2.7 Object Model/Logical Model: Class Diagram:**

# 

# **2.8 Database Model (Database Diagram):**

# 

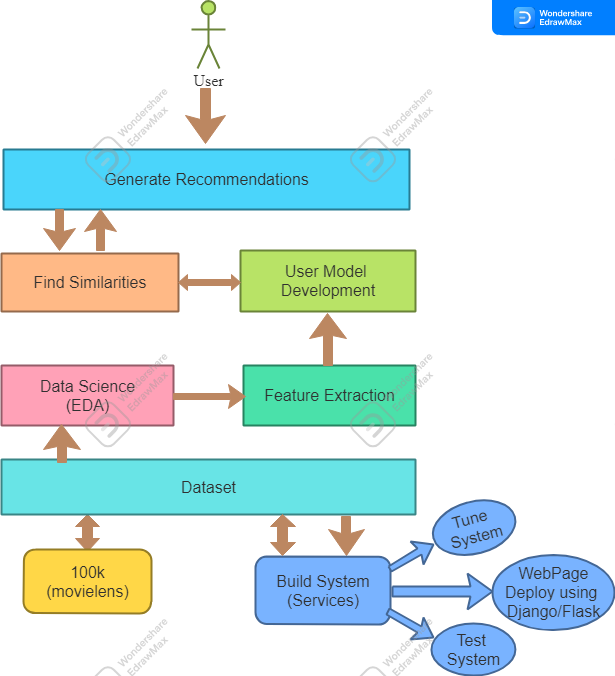
# **2.9 Graphical User Interfaces:**



**CHAPTER 3**

Development

3.1 Development plan (Architecture Diagram)



**REFERENCES**

<https://vulms.vu.edu.pk/Courses/CS619/Downloads/HelpingMaterialAll.htm>

<https://github.com/>

<https://chat.openai.com/>

<https://www.udemy.com/>

<https://www.youtube.com/>

**APPENDIX**

* Collaborative Filtering Movie Recommendation System with K-Nearest Neighbors.
* In this appendix, we provide additional details and code snippets related to the implementation of a movie recommendation system using Collaborative Filtering with K-Nearest Neighbors (KNN) algorithm. This system aims to provide personalized movie recommendations based on user preferences and similarities between users' rating behaviors.
* Data Pre-processing and Feature Extraction:

# Load the necessary libraries

import pandas as pd

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import NearestNeighbors

from fuzzywuzzy import process

# Load movie and rating data

movies = pd.read\_csv('movies.csv')

ratings = pd.read\_csv('ratings.csv')

# Merge movie and rating data

df = pd.merge(movies, ratings, on='movieId')

df = df[['userId', 'movieId', 'rating']]

# Pivot table for user-movie matrix

user\_movie\_matrix = df.pivot\_table(index='userId', columns='movieId', values='rating')

# Fill missing values with zeros

user\_movie\_matrix = user\_movie\_matrix.fillna(0)

* Building K-Nearest Neighbors Model:

# Create a KNN model

model\_knn = NearestNeighbors(metric='cosine', algorithm='brute', n\_neighbors=10)

# Fit the model on user-movie matrix

model\_knn.fit(user\_movie\_matrix.values)

* Movie Recommendation Function:

def get\_recommendations(movie\_name, n\_recommendations=10):

idx = process.extractOne(movie\_name, movies['title'])[2]

distances, indices = model\_knn.kneighbors(user\_movie\_matrix.iloc[idx].values.reshape(1, -1), n\_neighbors=n\_recommendations+1)

rec\_movies = [(movies.loc[indices.flatten()[i]]['title'], distances.flatten()[i]) for i in range(1, len(indices.flatten()))]

return rec\_movies

* Example Usage:

# Get recommendations for a movie

movie\_name = "The Dark Knight"

recommendations = get\_recommendations(movie\_name, n\_recommendations=5)

# Print recommendations

print(f"Recommended movies similar to '{movie\_name}':")

for movie, distance in recommendations:

print(f"Movie: {movie} | Similarity: {1 - distance:.4f}")

* In this example, we demonstrated the implementation of a Collaborative Filtering based movie recommendation system using the K-Nearest Neighbors algorithm. The system uses user-movie rating data to calculate movie similarities and provides personalized recommendations based on users' preferences. The get\_recommendations function takes a movie name and returns a list of recommended movies along with their similarity scores.