SE ASSIGNMENT 1

111803076\_MANISHA MARAKWAD

1. **FINAL PROJECT STATEMENT:**

Movie recommendation system

1. **PROJECT DOMAIN:**

Content-based filtering, collabrative filtering, HTML

1. **PROBLEM IDENTIFICATION :**

Movie Recommendation system that provides online movie and video streaming. but here Netflix need to build a recommendation system to predict a list of movies for users based on the other movies likes or dislikes

1. **RESEARCH:**

In this paper, we used control based filtering for movie recommendation system and to create the website we use HTML.

1. **SPECIFICATION:**
2. Uses HTML for making web interface for the face of project.
3. Uses python language for all coding part.
4. Dataset is created for movies and its type, given a specific id as show\_id, tile of movie, director, cast , rating, duration, description.
5. **TESTING**

Step1 – open up terminal

Step2- navigate to the folder where you have stored the dataset

Step3- run app.py

1. **APPLICATION**
2. Data pre-processing
3. Build Movie recommender machine learning model
4. Build REST API using flask

Test it on local host 127.0.0.1:5000 – API goes online

1. **REFERENCES**

A Movie Recommender System: MOVREC, august 2015, international journal of computer applications, DOI: 10.5120/ijca2015904111

<https://www.researchgate.net/publication/283042228_A_Movie_Recommender_System_MOVREC>

SE ASSIGNEMENT-2

1. GANTT CHART

**Software Requirement**

**Specification**

**for**

**Movie recommendation system**

**Prepared by Manisha Marakwad**

**College of Engineering Pune**

**Table of Contents**

1. Introduction
   1. Purpose
   2. Objectives
   3. Document Convention
   4. Intended Audience and Reading Suggestions
   5. Product Scope
   6. References
2. Overall Description
   1. Product perspective
   2. Product functions
   3. Design and Implementation constraints
   4. Assumptions and Dependencies
   5. User Documentation
3. External Interface Requirements
   1. User Interfaces
   2. Hardware Interfaces
4. System Features
   1. Building Fronted using HTML and CSS
   2. Building Backend
5. Other Non-functional Requirements
   1. Performance requirements
   2. Safety requirements
6. **INTRODUCTION**

**1.1 PURPOSE:**

Recommender systems help to personalize a platform and help the user find something they like. The easiest and simplest way to do this is to recommend the most popular items. However, to really enhance the user experience through personalized recommendations, we need dedicated recommender systems.

Keywords: Movies, Recommandation system, content-based filtering, collaborative Filtering

**1.2 OBJECTIVES:**

1. To better understand into the algorithms used in movie recommender system
2. To gain insight into the variation of customers movie preference on netflix.
3. To build a ML model that would recommend movies on netflix to customers.

**1.3 Document Conventions**:

The document does not have any conventions. All data must be given equal priority.

**1.4 Intended Audience and Reading Suggestions:**

The Overall Description section will provide the reader with a step-by-step view of how the system is intended to be built

**1.5 SCOPE:**

A recommendation system has been implemented based on hybrid approach of collaborative filtering engine and content based engine. We have tried to combine the existing algorithms for recommendation to come with a hybrid one. It improves the performance by overcoming the drawbacks of traditional recommendation system.

Recommender system make the selection process easier for the users. Hybrid recommendation engine is a competent system to recommend movies for E-users , recommender algorithm are quite slow with inaccuracies. The recommender system will be a great web application. Our approach can be extended to various domains to recommend books, music etc.

**REFERNCES:**

A Movie Recommender System: MOVREC, august 2015, international journal of computer applications, DOI: 10.5120/ijca2015904111

<https://www.researchgate.net/publication/283042228_A_Movie_Recommender_System_MOVREC>

**2. OVERALL DESCRIPTION:**

**2.1 Product perspective**

A recommender system predicts the attitude of a user towards an item. As to this project, it makes recommendations on movies to users. First, the system receives movie names, then it suggest movies.

**2.2 product functions**

The product have following functionality:

* Test it on local host 127.0.0.1:5000 – API goes online
* Search a movie
* System recommend movies to user based on similar items

**2.3 Design and implementation constraints**

Recommendation system helps user to find and select items from a large number available on the web. Given a large set of objects and a describing user needs, they offer the user a small set of items that are well suited description.

similarly , a Movie recommendation system provides a level of comfort and personalization Helps user to interact better with the system and watch movies that meet their needs.

Control-Based filtering :

In content-based filtering, It suggest additional items with similar properties and it is based on the items are recommended based on comparison between item profile and user profile.

Advantages-

* it capable of recommending unrated items
* Easily can explain recommender system by listing the content feature of an item

Collaborative filtering :

Collaborative filtering system recommends items based on similarity measures between user and items. The system recommends item preferred by similar user.

It basically have two types:

* User based filtering
* Item based filtering

.

**2.4 Assumptions and Dependencies**

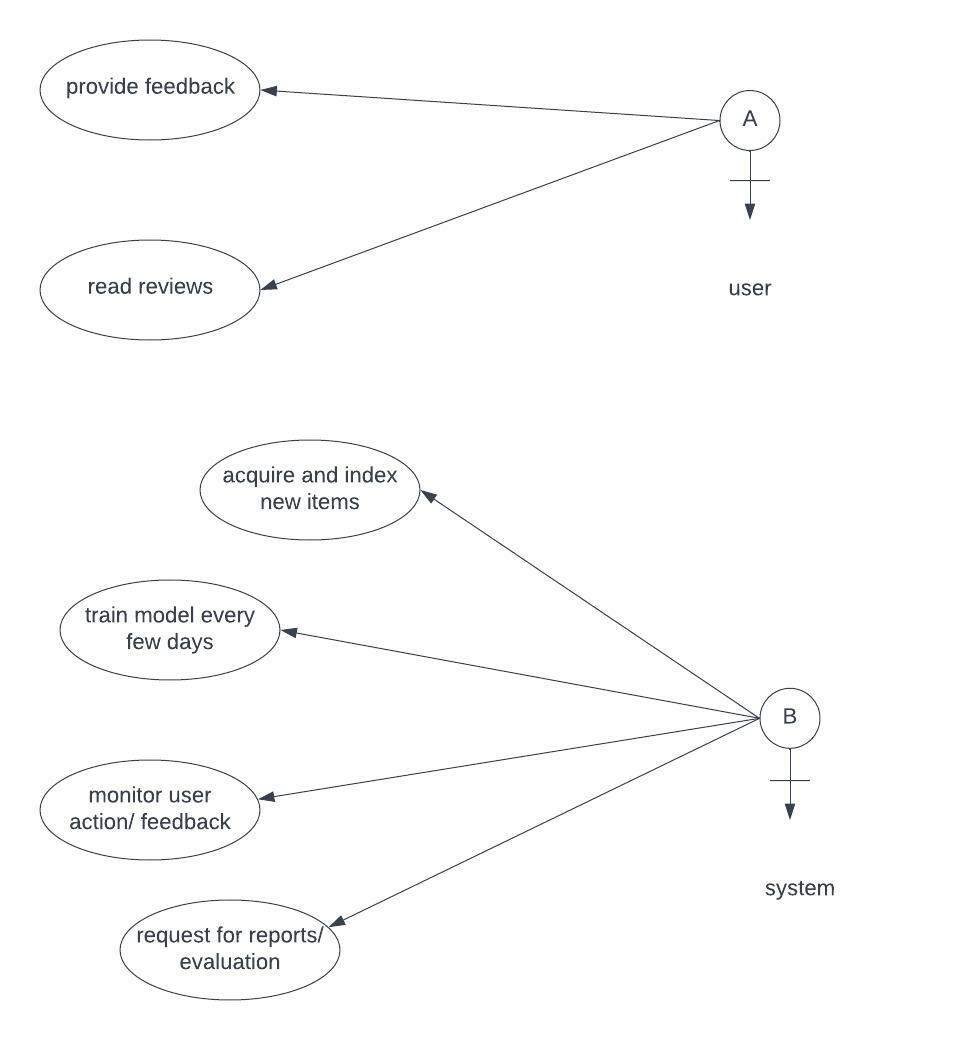
1. It is assumed that at least daily one million logs will be provided for the system’s use and the past 60 days’ logs will be provided

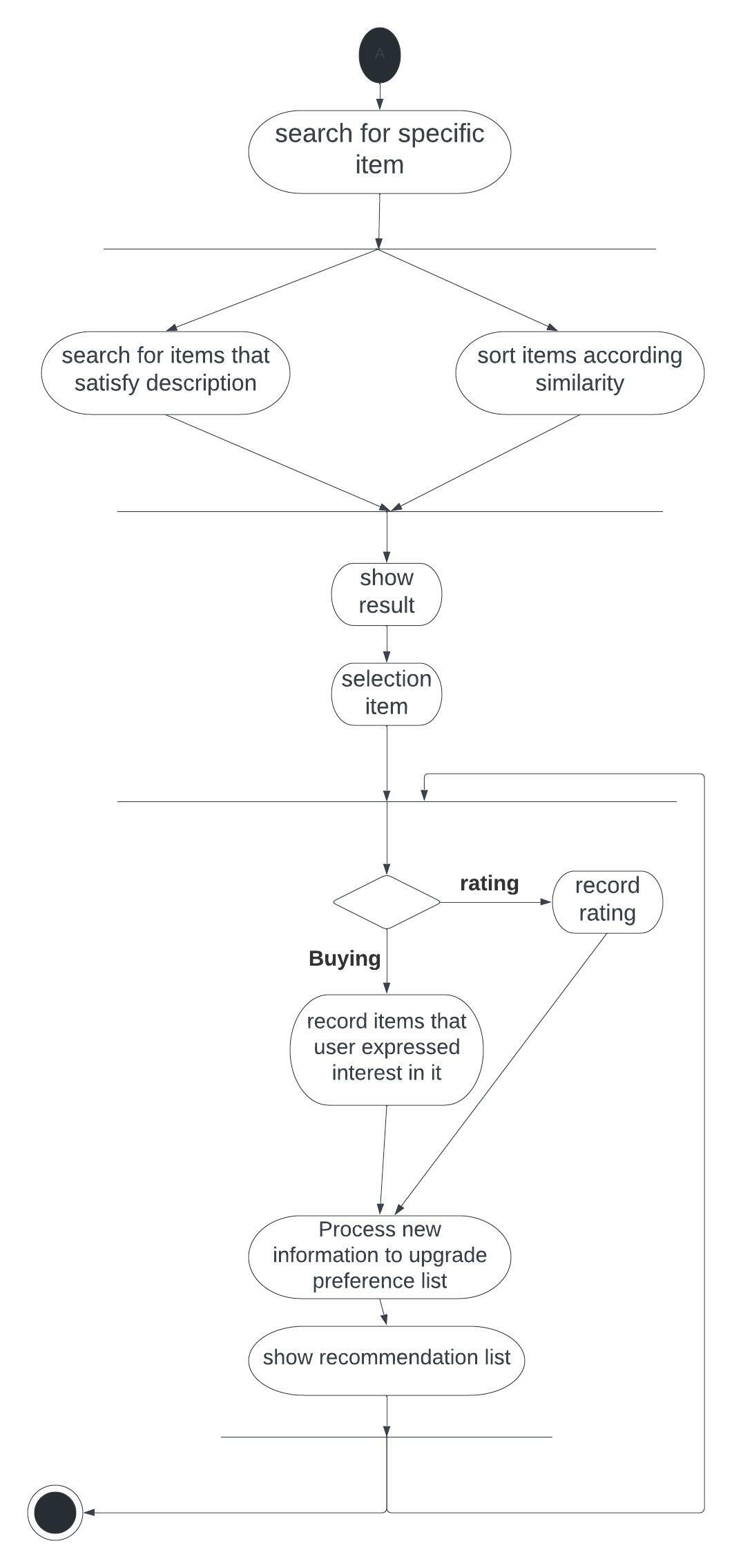
2.Another assumption is that the user has a web browser and a capable hardware in order to launch the website

**2.5 User Documentation**

The following documentation are available for reference regarding the use of product: problem identification, SRS file, UML Diagrams.

USE CASE DIAGRAM:



ACTIVITY DIAGRAM

**3 External Interface Requirements**

* 1. User Interfaces: Following are the specifications of User Interface to be built for the product:

Home page- used to enter the required movie name

Recommendation page – used to show list of recommended movies’ names

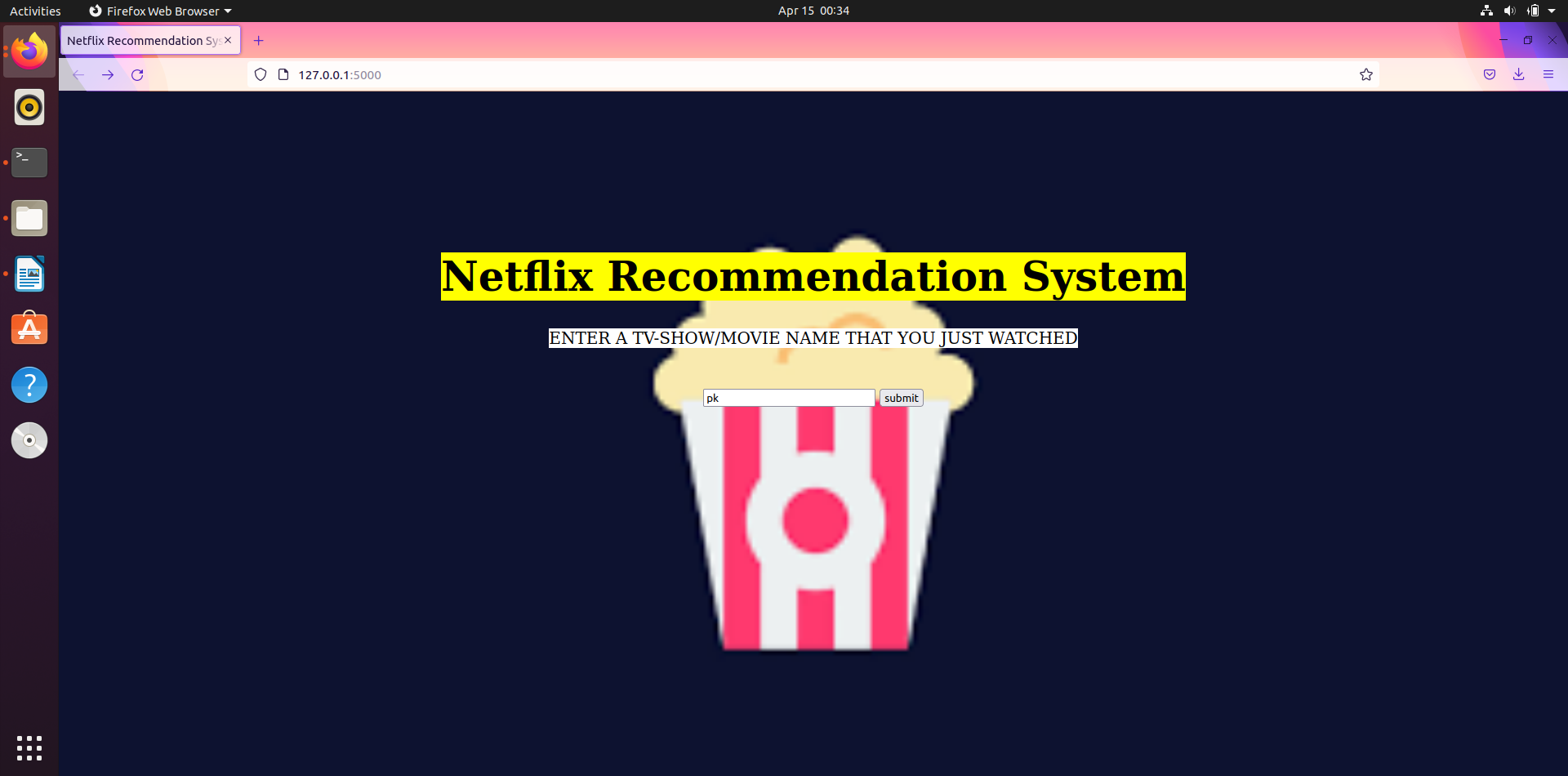
* 1. Hardware Interfaces:

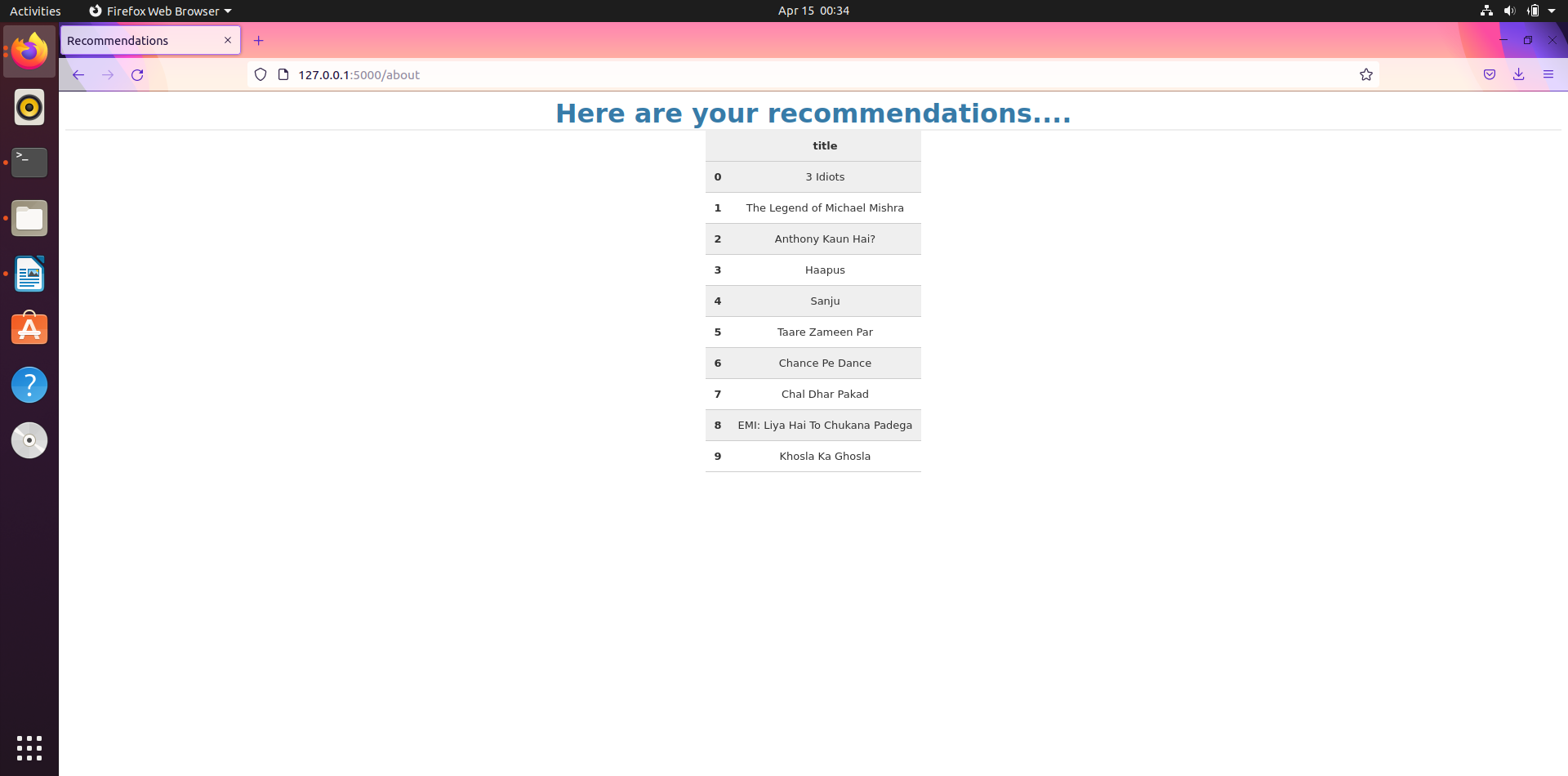
For the purpose of this application the database, logins, access is completely software based. Thus, is does not require an actual hardware interface.

* 1. Communications Interfaces:

Localhost will be used for now, but can be converted fully web based after setting up actual website.

**4. System Features**

4.1 Building Frontend using HTML and CSS:



4.2 Building Backend:

Netflix dataset -

Netflix\_titles.csv: This csv file contains show\_id, type of movie, title, director, cast, rating, duration, description.

CODING SCREENSHOT WITH RESULT:

Here we build a movie recommendation system using Flask based deployment :

In this we have used

* data pre-processing:

-In this we need to go through dataset and load data frames

* build movie recommender system using ML model

-to build this model , here we created a counting matrix generated with the vector counting tool. We create a counting vector with English stop words and fit and transfer the soup column we created in previous section. Technique we used here is cosine similarity. It is simply metric used to determine the similarity of documents .

After constructing a cosine similarity matrix forb our dataset, here we can sort the result to find top 10 similar movies.

In get\_recommendation() we used two parameters:

title , cosine\_sim

* build a REST API using flask

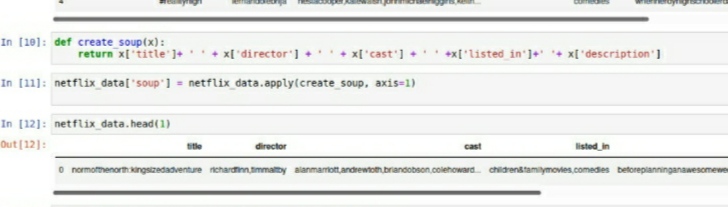
created a RESTful API with Flask—GET Requests

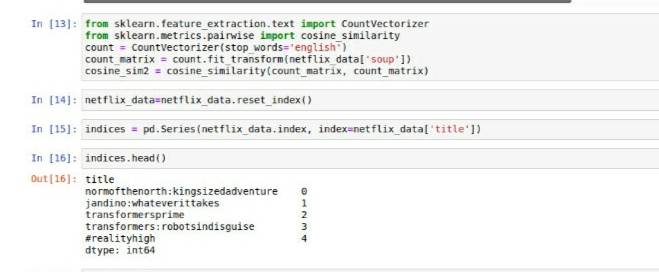
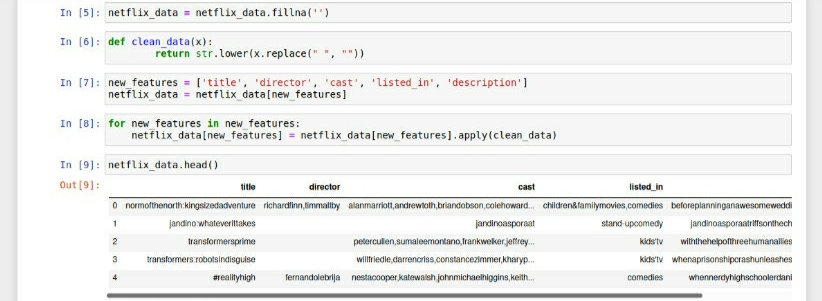
in this we create a instance of class .

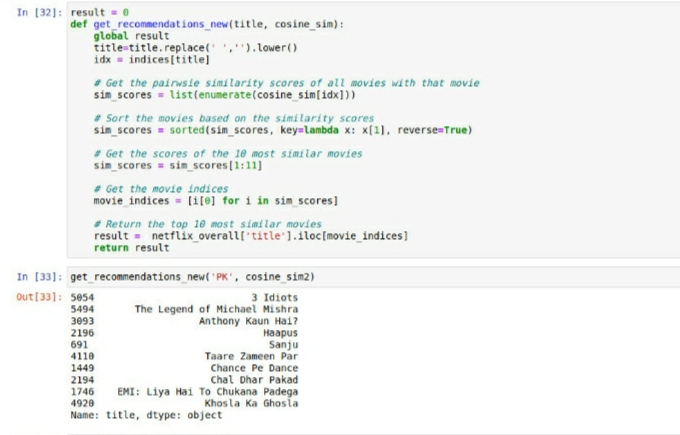
route()-this decorator to tell flask what URL should trigger our function. We use the /about endpoint with the base URL.

* test it on local host 127.0.0.1:5000-API goes online













**5. Other Non-Functional Requirements**

5.1 Performance Requirements: As the movie data is stored and accessed as a database, the time dependency will be extremely less. The recommended movies can be shown asap.

5.2 Safety Requirements: Since the product is completely localhost based and does not use actual web technology it is safe from hacking.

**CONCLUSION:**

Recommender system come up with a strategy that focuses on dealing with user personal interests. In this Netflix recommender system it offers recommendation by matching and searching similar users habits and suggesting movies that share characteristics with films that users have rated highly.

**Github link: https**: https://github.com/manisha1306/movie-recommendation-system.git