CODTECH INTERNSHIP TASK-01

BLOG BACKEND SYSTEM

NAME : VANTHANA.V

INTERN ID : CTO4DY468

Email : [poojavanthana5@gmail.com](mailto:poojavanthana5@gmail.com)

Title: AI-Powered Recommendation Engine

**Instructions:**

You are required to **develop a backend recommendation engine** using machine learning techniques and expose it via a REST API.

### **Task requirements:**

**Goal**:  
Build a Python-based backend that provides smart recommendations based on user behavior or preferences.

**Scope Includes**:

* **Movie or Product Recommendations**  
  Use datasets such as MovieLens or e-commerce product data.
* **Machine Learning**  
  Apply collaborative filtering, content-based filtering, or hybrid models using **pandas**, **scikit-learn**, or **surprise** libraries.
* **API Endpoint**  
  Expose the recommendation engine via **Flask** or **FastAPI**.

### **Deliverable**

* A working **Python backend project**.
* REST **API endpoint(s)** that return recommendations.
* A README explaining how to run the project and test the endpoints.

**Program :**

from flask import Flask, request, jsonify

import pandas as pd

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import linear\_kernel

app = Flask(\_\_name\_\_)

# Sample movie dataset

data = {

'title': ['The Matrix', 'John Wick', 'Inception', 'Interstellar', 'The Notebook'],

'genre': ['Action Sci-Fi', 'Action Thriller', 'Action Sci-Fi', 'Adventure Sci-Fi', 'Romance Drama']

}

df = pd.DataFrame(data)

# TF-IDF on genres

tfidf = TfidfVectorizer(stop\_words='english')

tfidf\_matrix = tfidf.fit\_transform(df['genre'])

# Cosine similarity matrix

cosine\_sim = linear\_kernel(tfidf\_matrix, tfidf\_matrix)

# Title to index mapping

indices = pd.Series(df.index, index=df['title']).drop\_duplicates()

# Recommendation function

def get\_recommendations(title):

if title not in indices:

return []

idx = indices[title]

sim\_scores = list(enumerate(cosine\_sim[idx]))

sim\_scores = sorted(sim\_scores, key=lambda x: x[1], reverse=True)[1:4]

movie\_indices = [i[0] for i in sim\_scores]

return df['title'].iloc[movie\_indices].tolist()

# API endpoint

@app.route('/recommend', methods=['GET'])

def recommend():

title = request.args.get('title')

if not title:

return jsonify({'error': 'Missing title parameter'}), 400

recommendations = get\_recommendations(title)

return jsonify({'recommendations': recommendations})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**Sample output:**

{

"recommendations": [

"The Matrix",

"Interstellar",

"John Wick"

]

}