5. Movie Recommendation System (Content-Based)

Description: Recommend movies to users based on movie descriptions or genres using cosine similarity.

Steps: -

•Load a movie dataset with titles and descriptions (e.g., TMDB dataset).

•Preprocess text (cleaning, vectorization using TF-IDF).

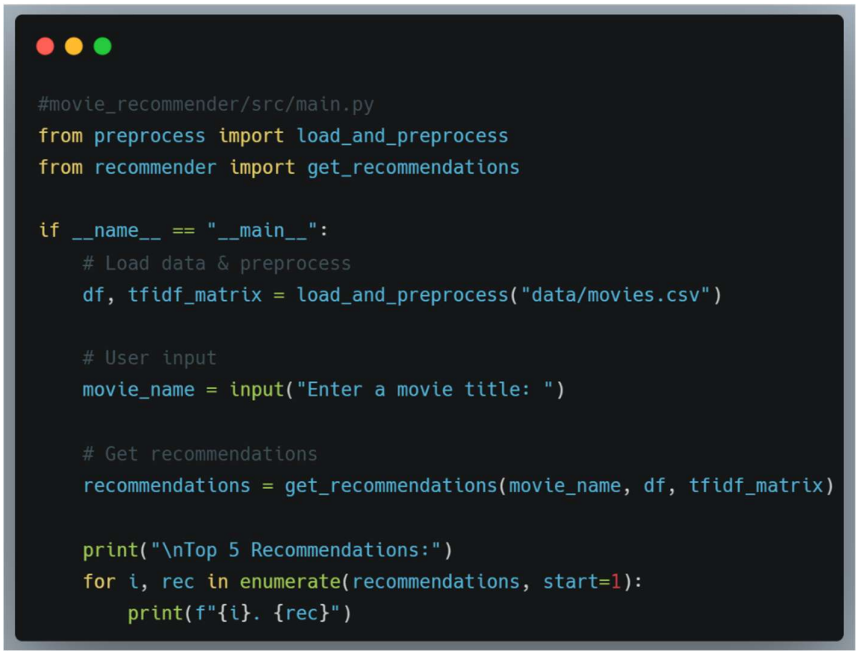
•Compute cosine similarity matrix.

•Define a function to recommend similar movies based on input title.

•Display top 5 recommendations.

Project Code

#movie_recommender/src/main.py



```python
#movie_recommender/src/main.py
from preprocess import load_and_preprocess
from recommender import get_recommendations

if __name__ == "__main__":
    # Load data & preprocess
    df, tfidf_matrix = load_and_preprocess("data/movies.csv")

    # User input
    movie_name = input("Enter a movie title: ")

    # Get recommendations
    recommendations = get_recommendations(movie_name, df, tfidf_matrix)

    print("\nTop 5 Recommendations:")
    for i, rec in enumerate(recommendations, start=1):
        print(f"{i}. {rec}")
```

#movie_recommender/src/preprocess.py

```python
#movie_recommender/src/preprocess.py
import pandas as pd
import os

def load_and_preprocess(file_path):
    # Get absolute path
    base_dir = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
    full_path = os.path.join(base_dir, file_path)

    df = pd.read_csv(full_path)
    df['overview'] = df['overview'].fillna('')

    from sklearn.feature_extraction.text import TfidfVectorizer
    tfidf = TfidfVectorizer(stop_words='english')
    tfidf_matrix = tfidf.fit_transform(df['overview'])

    return df, tfidf_matrix
```

#movie_recommender/src/preprocess.py

```python
#movie_recommender/src/preprocess.py
import numpy as np
import pandas as pd
import os
from sklearn.metrics.pairwise import cosine_similarity

def get_recommendations(title, df, tfidf_matrix):
    cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

    # Map indices to movie titles
    indices = pd.Series(df.index, index=df['title']).drop_duplicates()

    if title not in indices:
        return [f"Movie '{title}' not found in dataset."]

    idx = indices[title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:6]  # Top 5 excluding itself
    movie_indices = [i[0] for i in sim_scores]

    return df['title'].iloc[movie_indices].tolist()
```

Project Output