### Advance Python Project Assignment

### 1. Build a Flask app that scrapes data from multiple websites and displays it on your site

First, install the necessary packages:

pip install Flask requests beautifulsoup4

Create a folder structure like this:

/project

/templates

index.html

app.py

app.py:

from flask import Flask, render\_template

import requests

from bs4 import BeautifulSoup

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

def scrape\_youtube():

url = 'https://www.youtube.com/results?search\_query=flask+tutorial'

response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

titles = [video.text for video in soup.find\_all('a', {'class': 'yt-uix-tile-link'})]

return titles[:5]

def scrape\_amazon():

url = 'https://www.amazon.com/s?k=flask+book'

headers = {'User-Agent': 'Mozilla/5.0'}

response = requests.get(url, headers=headers)

soup = BeautifulSoup(response.text, 'html.parser')

titles = [item.text.strip() for item in soup.find\_all('span', {'class': 'a-size-medium a-color-base a-text-normal'})]

return titles[:5]

@app.route('/')

def home():

youtube\_data = scrape\_youtube()

amazon\_data = scrape\_amazon()

return render\_template('index.html', youtube\_data=youtube\_data, amazon\_data=amazon\_data)

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

app.run(debug=True)

templates/index.html:

<!DOCTYPE html>

<html>

<head>

<title>Scraped Data</title>

</head>

<body>

<h1>YouTube Data</h1>

<ul>

{% for item in youtube\_data %}

<li>{{ item }}</li>

{% endfor %}

</ul>

<h1>Amazon Data</h1>

<ul>

{% for item in amazon\_data %}

<li>{{ item }}</li>

{% endfor %}

</ul>

</body>

</html>

### 2. Create a Flask app that consumes data from external APIs and displays it to users

First, install the necessary packages:

pip install Flask requests

Create a folder structure like this:

/project

/templates

weather.html

app.py

app.py:

from flask import Flask, render\_template, request

import requests

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

@app.route('/')

def home():

return render\_template('weather.html')

@app.route('/weather', methods=['POST'])

def weather():

city = request.form['city']

api\_key = 'your\_api\_key' # Replace with your actual API key

url = f'http://api.openweathermap.org/data/2.5/weather?q={city}&appid={api\_key}'

response = requests.get(url).json()

weather\_data = {

'city': city,

'temperature': response['main']['temp'],

'description': response['weather'][0]['description']

}

return render\_template('weather.html', weather\_data=weather\_data)

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

app.run(debug=True)

templates/weather.html:

<!DOCTYPE html>

<html>

<head>

<title>Weather</title>

</head>

<body>

<h1>Weather App</h1>

<form method="post" action="/weather">

<label for="city">City:</label>

<input type="text" id="city" name="city">

<input type="submit" value="Get Weather">

</form>

{% if weather\_data %}

<h2>Weather in {{ weather\_data.city }}</h2>

<p>Temperature: {{ weather\_data.temperature }}</p>

<p>Description: {{ weather\_data.description }}</p>

{% endif %}

</body>

</html>

### 3. Implement OAuth2 authentication to allow users to log in using their Google or Facebook accounts

First, install the necessary packages:

pip install Flask-OAuthlib

Create a folder structure like this:

/project

/templates

login.html

app.py

app.py:

from flask import Flask, redirect, url\_for, session

from flask\_oauthlib.client import OAuth

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

app.secret\_key = 'your\_secret\_key'

oauth = OAuth(app)

google = oauth.remote\_app(

'google',

consumer\_key='your\_google\_client\_id',

consumer\_secret='your\_google\_client\_secret',

request\_token\_params={

'scope': 'email',

},

base\_url='https://www.googleapis.com/oauth2/v1/',

request\_token\_url=None,

access\_token\_method='POST',

access\_token\_url='https://accounts.google.com/o/oauth2/token',

authorize\_url='https://accounts.google.com/o/oauth2/auth',

)

@app.route('/')

def index():

return 'Welcome to the OAuth2 example'

@app.route('/login')

def login():

return google.authorize(callback=url\_for('authorized', \_external=True))

@app.route('/logout')

def logout():

session.pop('google\_token')

return redirect(url\_for('index'))

@app.route('/login/authorized')

def authorized():

response = google.authorized\_response()

if response is None or response.get('access\_token') is None:

return 'Access denied: reason={} error={}'.format(

request.args['error\_reason'],

request.args['error\_description']

)

session['google\_token'] = (response['access\_token'], '')

user\_info = google.get('userinfo')

return 'Logged in as: ' + user\_info.data['email']

@google.tokengetter

def get\_google\_oauth\_token():

return session.get('google\_token')

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

app.run(debug=True)

### 4. Develop a recommendation system using Flask that suggests content to users based on their preferences

First, install the necessary packages:

pip install Flask pandas scikit-learn

Create a folder structure like this:

/project

/templates

recommend.html

app.py

data.csv

data.csv (example data):

user\_id,item\_id,rating

1,101,5

1,102,3

1,103,4

2,101,4

2,102,2

2,103,5

3,101,3

3,102,4

3,103,2

app.py:

from flask import Flask, render\_template, request

import pandas as pd

from sklearn.metrics.pairwise import cosine\_similarity

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

def get\_recommendations(user\_id):

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

user\_item\_matrix = data.pivot\_table(index='user\_id', columns='item\_id', values='rating')

user\_similarity = cosine\_similarity(user\_item\_matrix.fillna(0))

user\_similarity\_df = pd.DataFrame(user\_similarity, index=user\_item\_matrix.index, columns=user\_item\_matrix.index)

similar\_users = user\_similarity\_df[user\_id].sort\_values(ascending=False).index[1:]

recommendations = user\_item\_matrix.loc[similar\_users].mean().sort\_values(ascending=False).index[:5]

return recommendations

@app.route('/')

def home():

return render\_template('recommend.html')

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

def recommend():

user\_id = int(request.form['user\_id'])

recommendations = get\_recommendations(user\_id)

return render\_template('recommend.html', recommendations=recommendations)

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

app.run(debug=True)

templates/recommend.html:

<!DOCTYPE html>

<html>

<head>

<title>Recommendations</title>

</head>

<body>

<h1>Recommendation System</h1>

<form method="post" action="/recommend">

<label for="user\_id">User ID:</label>

<input type="text" id="user\_id" name="user\_id">

<input type="submit" value="Get Recommendations">

</form>

{% if recommendations %}

<h2>Recommended Items</h2>

<ul>

{% for item in recommendations %}

<li>{{ item }}</li>

{% endfor %}

</ul>

{% endif %}

</body>

</html>

### Deployment Instructions

1. **Choose a Cloud Platform**: You can use platforms like Heroku, AWS, Google Cloud, or Azure. Here, I'll use Heroku as an example.
2. **Install Heroku CLI**: Follow the instructions on the [Heroku website](https://devcenter.heroku.com/articles/heroku-cli) to install the Heroku CLI.
3. **Create a `requirements.txt` file**: This file should list all the dependencies of your project. You can generate it using:
4. **Create a `Procfile`**: This file should specify the command to run your app. For example:
5. **Initialize a Git Repository**:
6. **Deploy to Heroku**: