Copilot

Let’s tackle each of these tasks one by one.

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 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:

   ```
   pip freeze > requirements.txt
   ```

4. **Create a `Procfile`**: This file should specify the command to run your app. For example:

   ```
   web: python app.py
   ```

5. **Initialize a Git Repository**:

   ```
   git init
   git add .
   git commit -m "Initial commit"
   ```

6. **Deploy to Heroku**:

   ```
   heroku create
   git push heroku master
   heroku open
   ```

Feel free to ask if you need any further assistance or explanations!