Topic	FLASK MOCKUP 2	
Class Description	**The student will complete the Flask API for their mobile app on movie recommendation.**	
Class	**PRO C142**	
Class time	**45 mins**	
Goal	● Student completes the Flask API for movie recommendation App	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm-Up** **Teacher-Led Activity 1** **Student-Led Activity 1** **Wrap-Up**	**5 mins** **5 mins** **30 mins** **5 mins**

WARM-UP SESSION - 5 mins

Teacher Starts Slideshow
Slide # to #
<**Note**: Only Applicable for Classes with VA>
Refer to speaker notes and follow the instructions on each slide.

Teacher Action	Student Action
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** ● Greet the student. ● Revision of previous class activities. ● Quizzes.	**ESR**: Hi, thanks! Yes, I am excited about it! Click on the slide show tab and present the slides.

WARM-UP QUIZ
Click on In-Class Quiz

Continue WARM-UP Session
Slide # to #
<**Note**: Only Applicable for Classes with VA>

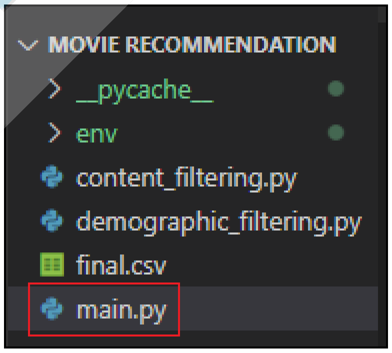
Activity Details

Following are the session deliverables:

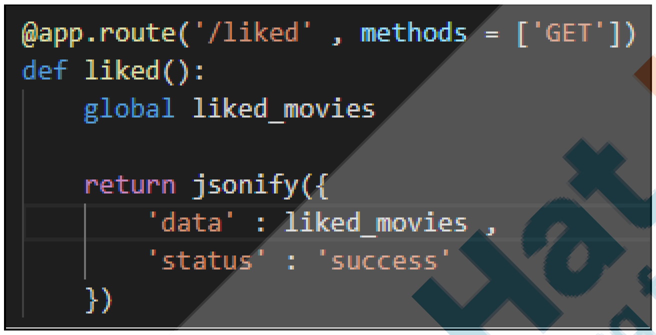
- Appreciate the student.
- Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.

Teacher Action	Student Action
We have completed the Flask API for the first screen of our mobile app. Now, we want to complete the API, by thinking through the second screen of our app so that we can start with the React Native part! For the second page, we will be displaying the movies **liked** by the user and the **recommendations** based on the user's preference, so we will build,	

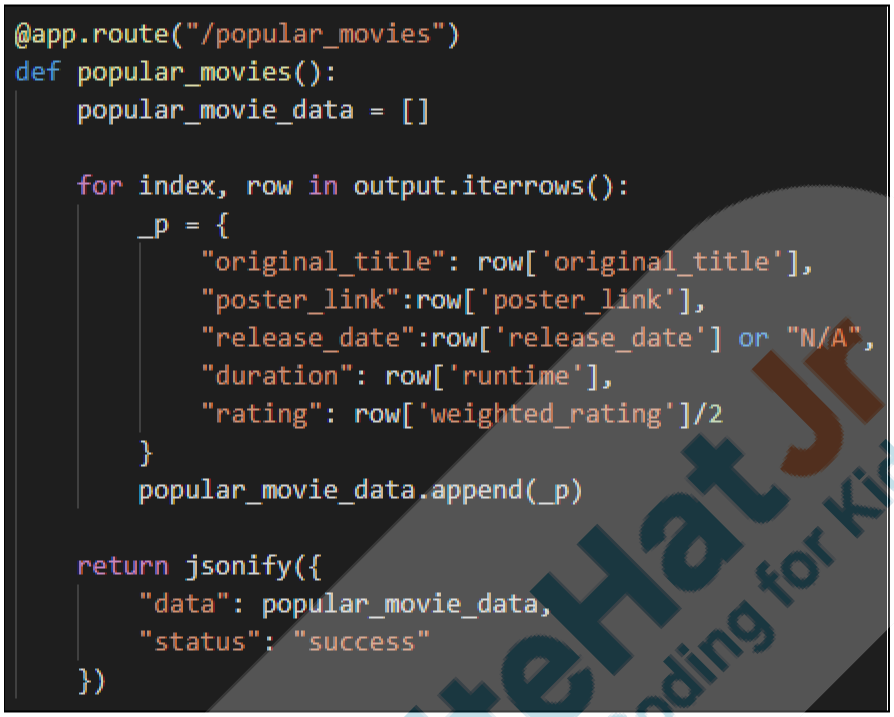
● One API which returns a list of **liked** movies. ● One API which returns a list of **popular** movies. ● One API which returns a list of **recommended** movies. In all, we want to build these three APIs. Sounds like a plan?	**ESR**: Yes!
Teacher Ends Slideshow	
TEACHER-LED ACTIVITY - 5 mins	
Teacher Initiates Screen Share	
ACTIVITY ● **Help the student get the data right.**	
Teacher Action	**Student Action**
Teacher Stops Screen Share	
So now it's your turn. Please share your screen with me.	
Teacher Starts Slideshow **Slide # to #** <**Note**: Only Applicable for Classes with VA> Refer to speaker notes and follow the instructions on each slide.	
STUDENT-LED ACTIVITY - 30 mins	
● **Ask the student to press the ESC key to come back to the panel.** ● **Guide the student to start Screen Share.**	

• **The teacher gets into Full Screen.**	
Student Initiates Screen Share	
<u>ACTIVITY</u> • **Student codes to complete the remaining two APIs.**	
Teacher Action	**Student Action**
To create the APIs for the second screen of your movie recommendation mobile app, download all the files from this [Student activity 1](). Traverse to the downloaded folder using the **command prompt** and create a Python virtual environment in it using the command **python -m venv env**. Activate the environment and install the **flask**, **sklearn,** and **pandas** module using the commands **pip install flask, pip install scikit-learn,** and **pip install pandas.** *Help the student set up a basic Flask Project inside a virtual environment.* Open project folder with the help Visual Studio Code editor and click on **main.py** file.	
	

```
MOVIE RECOMMENDATION
  > __pycache__
  > env
  content_filtering.py
  demographic_filtering.py
  final.csv
  main.py
```

Define a new route and specify the URL as **'/liked**, which will listen for incoming **GET** requests only. Define a decorator method named **liked(),** which will return the list of **liked_movies** and a **success status** in **JSON** format, whenever a **GET** request is received on this API.	
	
Define a new route and specify the URL as **'/popular_movies**, which will listen for incoming **GET** requests only. Define a decorator method named **popular_movies(),** which will extract the **original_title, poster_link, duration, release date** and **rating** for each movie from our **output DataFrame** and append all the extracted information into **popular_movie_data** list. Finally, it will return the **popular_movie_data** list and a **success status** in **JSON** format, whenever a **GET** request is received on this API.	

```
@app.route('/liked' , methods = ['GET'])
def liked():
    global liked_movies

    return jsonify({
        'data' : liked_movies ,
        'status' : 'success'
    })
```



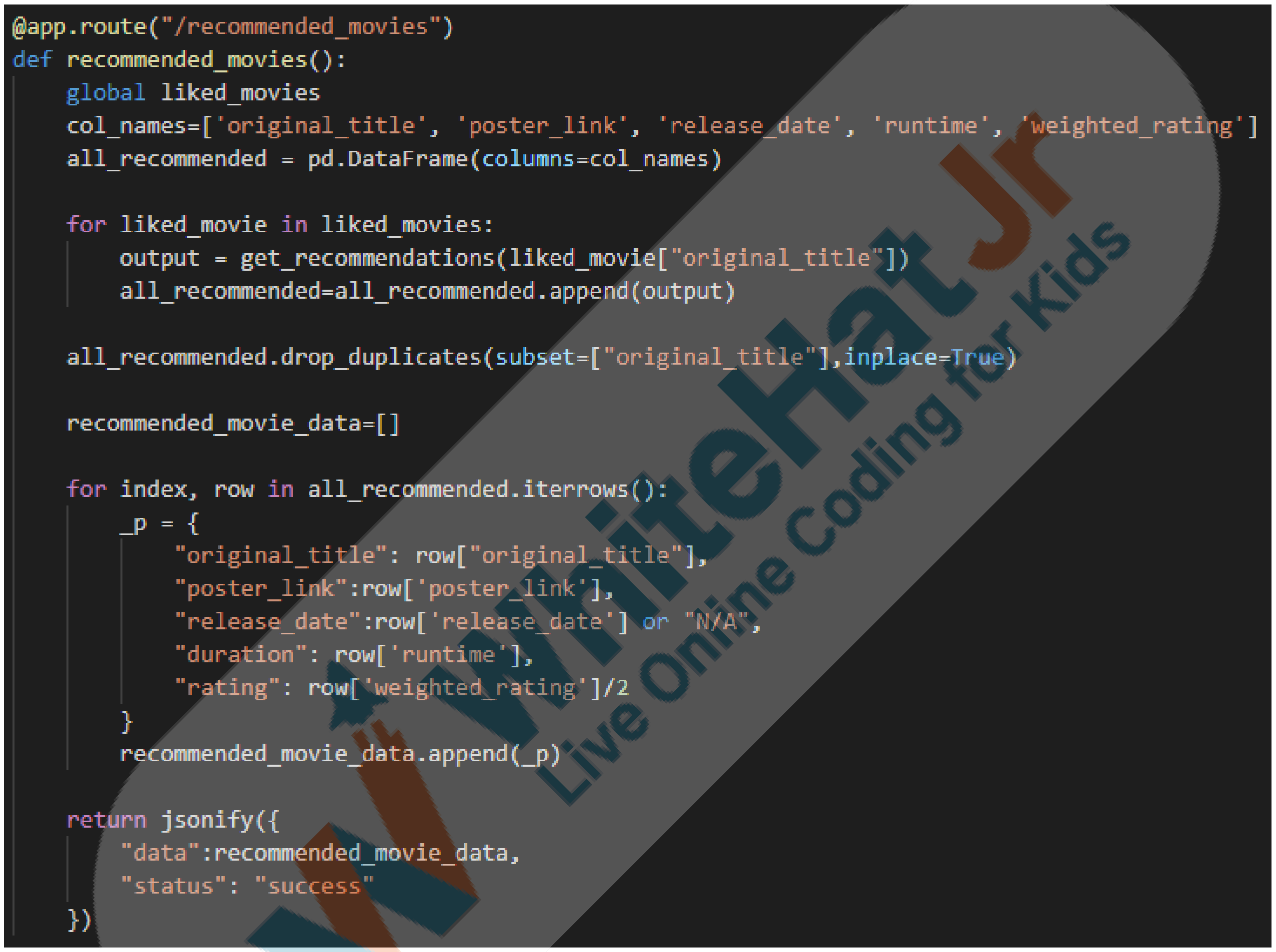
```
@app.route("/popular_movies")
def popular_movies():
    popular_movie_data = []

    for index, row in output.iterrows():
        _p = {
            "original_title": row['original_title'],
            "poster_link":row['poster_link'],
            "release_date":row['release_date'] or "N/A",
            "duration": row['runtime'],
            "rating": row['weighted_rating']/2
        }
        popular_movie_data.append(_p)

    return jsonify({
        "data": popular_movie_data,
        "status": "success"
    })
```

Define a new route and specify the URL as **'/recommended_movies**, which will listen for incoming **GET** requests only.

Define a decorator method named **recommended_movies(),** which will,

- Use the **get_recommendations()** method in order to get the movies which are similar to the ones in the **liked_movies** list.
- Use the **drop_duplicates()** method, to remove the duplicate movies from the DataFrame.
- Iterate over the DataFrame and extract the **original_title, poster_link, duration, release date,** and **rating** for each movie from the **all_recommend DataFrame** and append all the extracted information into **recommended_movie_data** list.

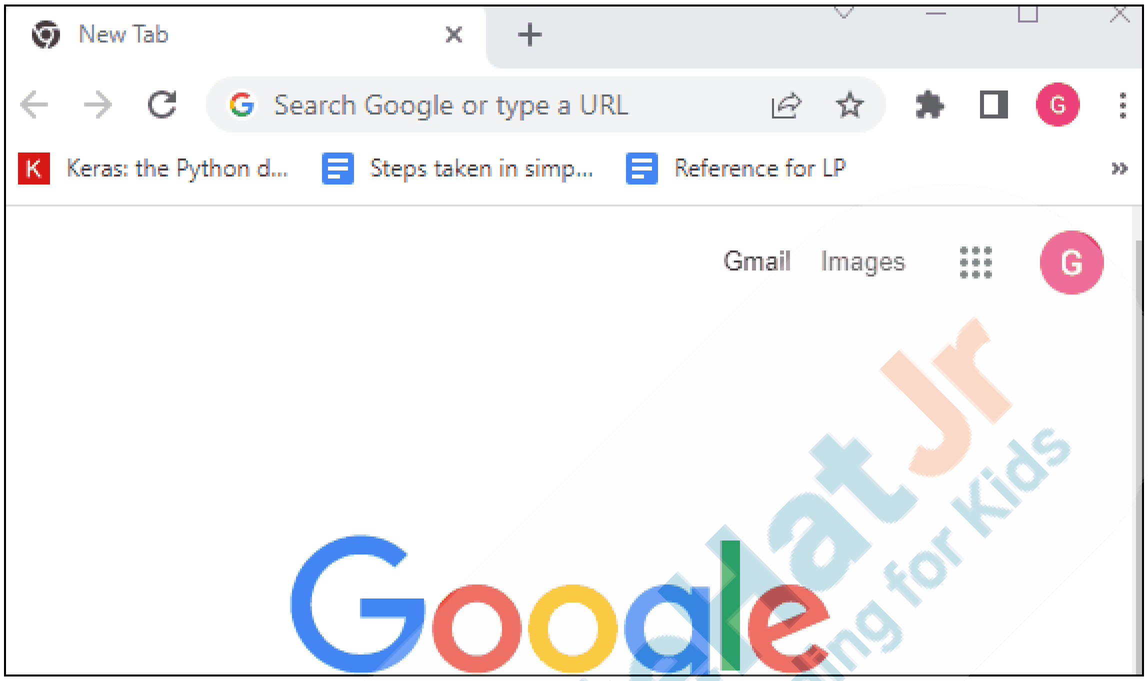
Finally, it will return the **recommended_movie_data** list and a **success status** in **JSON** format, whenever a **GET** request is received on this API.



```
@app.route("/recommended_movies")
def recommended_movies():
    global liked_movies
    col_names=['original_title', 'poster_link', 'release_date', 'runtime', 'weighted_rating']
    all_recommended = pd.DataFrame(columns=col_names)

    for liked_movie in liked_movies:
        output = get_recommendations(liked_movie["original_title"])
        all_recommended=all_recommended.append(output)

    all_recommended.drop_duplicates(subset=["original_title"],inplace=True)

    recommended_movie_data=[]

    for index, row in all_recommended.iterrows():
        _p = {
            "original_title": row["original_title"],
            "poster_link":row['poster_link'],
            "release_date":row['release_date'] or "N/A",
            "duration": row['runtime'],
            "rating": row['weighted_rating']/2
        }
        recommended_movie_data.append(_p)

    return jsonify({
        "data":recommended_movie_data,
        "status": "success"
    })
```

Finally test your APIs after running the **main.py** file:

1. Open localhost link http://127.0.0.1:5000/ in browser.
2. Click on the URL tab to add **/like** or /**liked** or **/popular_movies** or **/recommended_movies** right after the localhost link to check the API response.



Refer link:
https://s3-whjr-curriculum-uploads.whjr.online/28d0940e-2018-403d-bd03-aeee87d67e0a.gif

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins

Teacher Starts Slideshow
Slide # to #
<**Note**: Only Applicable for Classes with VA>

Activity details

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.

- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

Continue WRAP-UP Session
Slide # to #
<**Note**: Only Applicable for Classes with VA>

Activity Details

Following are the session deliverables:
- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

<u>**FEEDBACK**</u>
- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	Student Action
You get "hats-off" for your excellent work! Great! We have successfully completed our Flask API. Now in the next class, we will be starting to work on our mobile app for Movie Recommendation System!	*Make sure you have given at least 2 hats-off during the class for:* Creatively Solved Activities +10 Great Question +10

PROJECT OVERVIEW DISCUSSION
Refer the document below in Activity Links Sections

Teacher Clicks

ACTIVITY LINKS		
Activity Name	**Description**	**Links**
Teacher Activity 1	Reference Code	https://github.com/procodingclass/PRO-C142-Reference-Code.git
Teacher Activity 2	Output	https://s3-whjr-curriculum-uploads.whjr.online/28d0940e-2018-403d-bd03-aeee87d67e0a.gif
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.whjr.online/41c838c1-98d9-4749-8dde-b165e92806c7.pdf
Teacher Reference 2	Project Solution	https://github.com/procodingclass/PRO-C142-Project-Solution.git
Teacher Reference 3	Visual-Aid	Will be added after VA creation
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/8fb69229-0cb9-4a7a-9ca1-cd30420bf27e.pdf
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C142-Student-Activity.git