

Instructions for Running the Application

Prerequisites for running the application:

- **Python:** Ensure Python is installed in the system. Python 3.10.8 is used for building this application.
- **Visual Studio Code (VS Code):** VS Code IDE is used for implementing the source code. Any IDEs that support python such as VS Code, PyCharm, Jupyter Notebook etc can be used. Standalone text editor with command prompt can also be used.
- **Flask:** Flask is a micro web framework for Python that is used for building the REST API microapp. In terminal or command prompt install Flask using pip, the package installer for Python.

>pip install Flask

- **Requests:** The requests library is needed for unit testing, to send HTTP requests to video streaming application endpoints and verify that they respond correctly. Install the requests package using pip,
>pip install requests
- **Postman:** Popular API development and testing tool, Postman tool is used for testing and stimulating the REST APIs.

Steps to Run the Application:

There are 3 python files.

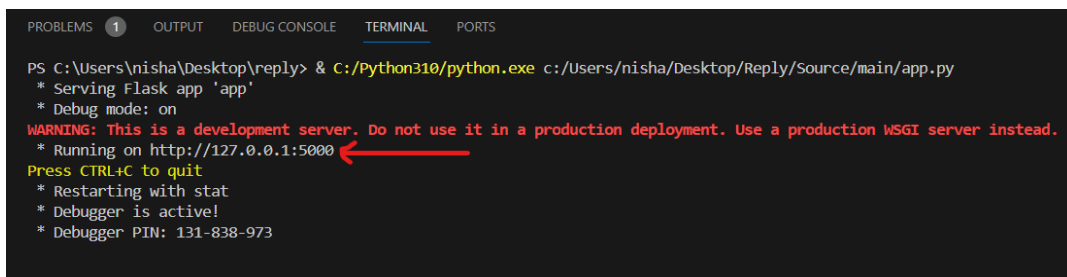
app.py – REST API app for the video streaming application.

reg.py – Script to run a smoke test/ sanity test for the video streaming application.

test_app.py – Unit test cases for the video streaming application.

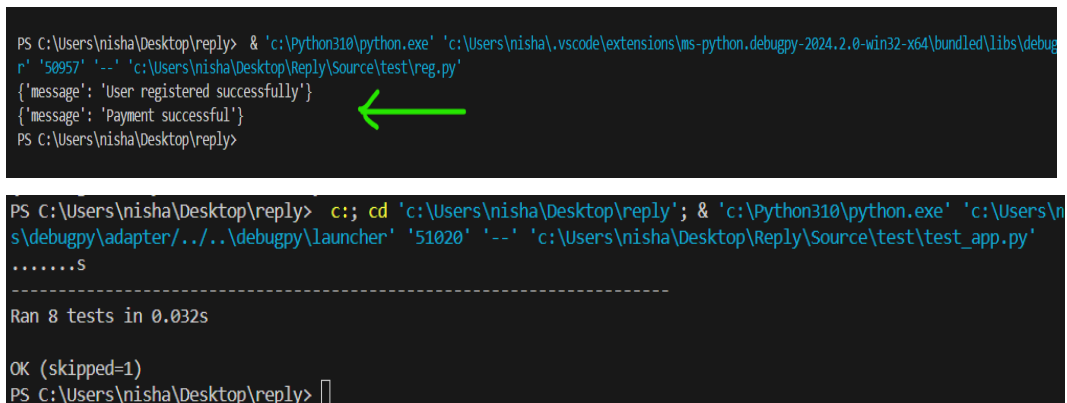
Using IDE

1. Open any IDE
2. Open the project directory where the video streaming application files located. The github repo url can be used for cloning the project - https://github.com/NishaElavunkal/Video_Streaming_Reg_App.git
3. Navigate to app.py file
4. Run the script within the IDE using run button.
5. Once the application is running, access it by opening a web browser and navigating to 'http://localhost:5000/' or 'http://127.0.0.1:5000/' to access the root endpoint.



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\nisha\Desktop\reply> & C:/Python310/python.exe c:/Users/nisha/Desktop/Reply/Source/main/app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 131-838-973
```

6. Access user endpoint by appending the routes to the base URL (<http://127.0.0.1:5000/users>).
7. Once the server is running, execute the reg.py and test_app.py.



```
PS C:\Users\nisha\Desktop\reply> & 'c:\Python310\python.exe' 'c:\Users\nisha\.vscode\extensions\ms-python.debugpy-2024.2.0-win32-x64\bundled\libs\debugpy\50957' '--' 'c:\Users\nisha\Desktop\Reply\Source\test\reg.py'
{'message': 'User registered successfully'}
{'message': 'Payment successful'}
PS C:\Users\nisha\Desktop\reply>

PS C:\Users\nisha\Desktop\reply> c:: cd 'c:\Users\nisha\Desktop\reply'; & 'c:\Python310\python.exe' 'c:\Users\nisha\.vscode\extensions\ms-python.debugpy-2024.2.0-win32-x64\bundled\libs\debugpy\launcher' '51020' '--' 'c:\Users\nisha\Desktop\Reply\Source\test\test_app.py'
.....S
-----
Ran 8 tests in 0.032s

OK (skipped=1)
PS C:\Users\nisha\Desktop\reply>
```

```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 131-838-973
127.0.0.1 - - [02/Apr/2024 14:31:20] "POST /users HTTP/1.1" 201 -
127.0.0.1 - - [02/Apr/2024 14:31:20] "POST /payments HTTP/1.1" 201 -
127.0.0.1 - - [02/Apr/2024 14:37:41] "GET /users HTTP/1.1" 200 -
127.0.0.1 - - [02/Apr/2024 14:37:41] "GET /users?CreditCard=yes HTTP/1.1" 200 -
127.0.0.1 - - [02/Apr/2024 14:37:41] "GET /users?CreditCard=no HTTP/1.1" 200 -
127.0.0.1 - - [02/Apr/2024 14:37:41] "POST /payments HTTP/1.1" 400 -
127.0.0.1 - - [02/Apr/2024 14:37:41] "POST /payments HTTP/1.1" 400 -
127.0.0.1 - - [02/Apr/2024 14:37:41] "GET /users?CreditCard=invalid HTTP/1.1" 400 -
127.0.0.1 - - [02/Apr/2024 14:37:41] "POST /users HTTP/1.1" 201 -
127.0.0.1 - - [02/Apr/2024 14:37:41] "POST /payments HTTP/1.1" 201 -
127.0.0.1 - - [02/Apr/2024 14:37:41] "POST /users HTTP/1.1" 409 -
127.0.0.1 - - [02/Apr/2024 14:44:57] "POST /users HTTP/1.1" 201 -
```

Using the Terminal

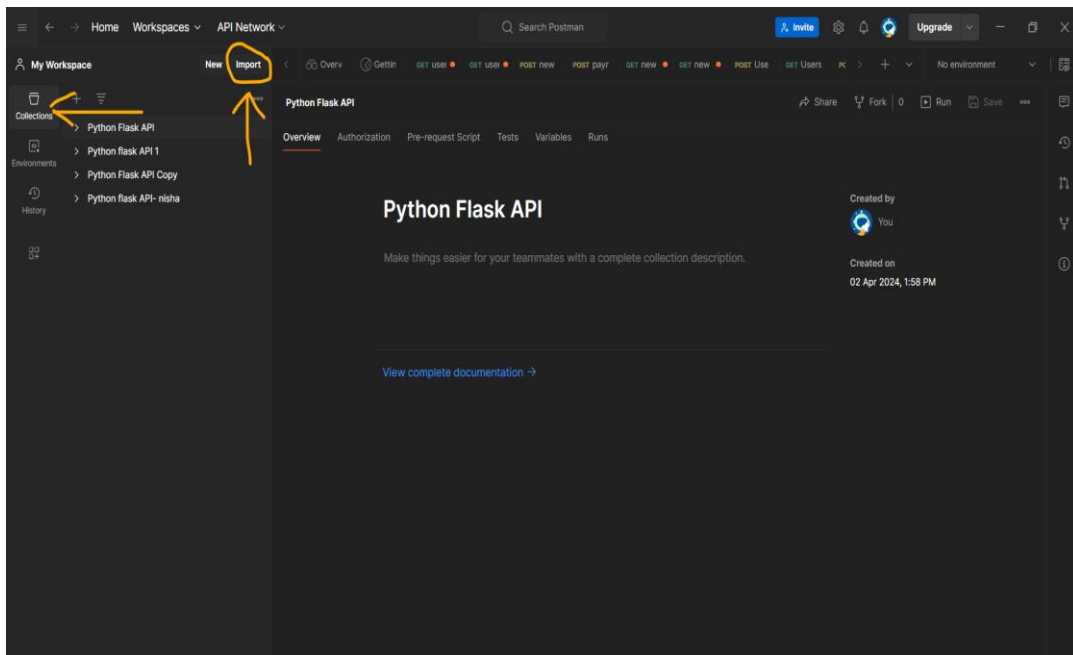
1. Open a terminal or command prompt.
2. Use the `cd` command to navigate to the directory containing the video streaming files.
3. Execute the `app.py` script using the Python interpreter: **`python app.py`**
4. Once the application is running, access it by opening a web browser and navigating to `'http://localhost:5000/'` or `'http://127.0.0.1:5000/'` to access the root endpoint.
5. Access user endpoint by appending the routes to the base URL (<http://127.0.0.1:5000/users>).
6. Once the server is running, execute the `reg.py` and `test_app.py` in different terminal window.

Make sure the `app.py` is executed and server is running before executing the `reg.py` and `test_app.py` files.

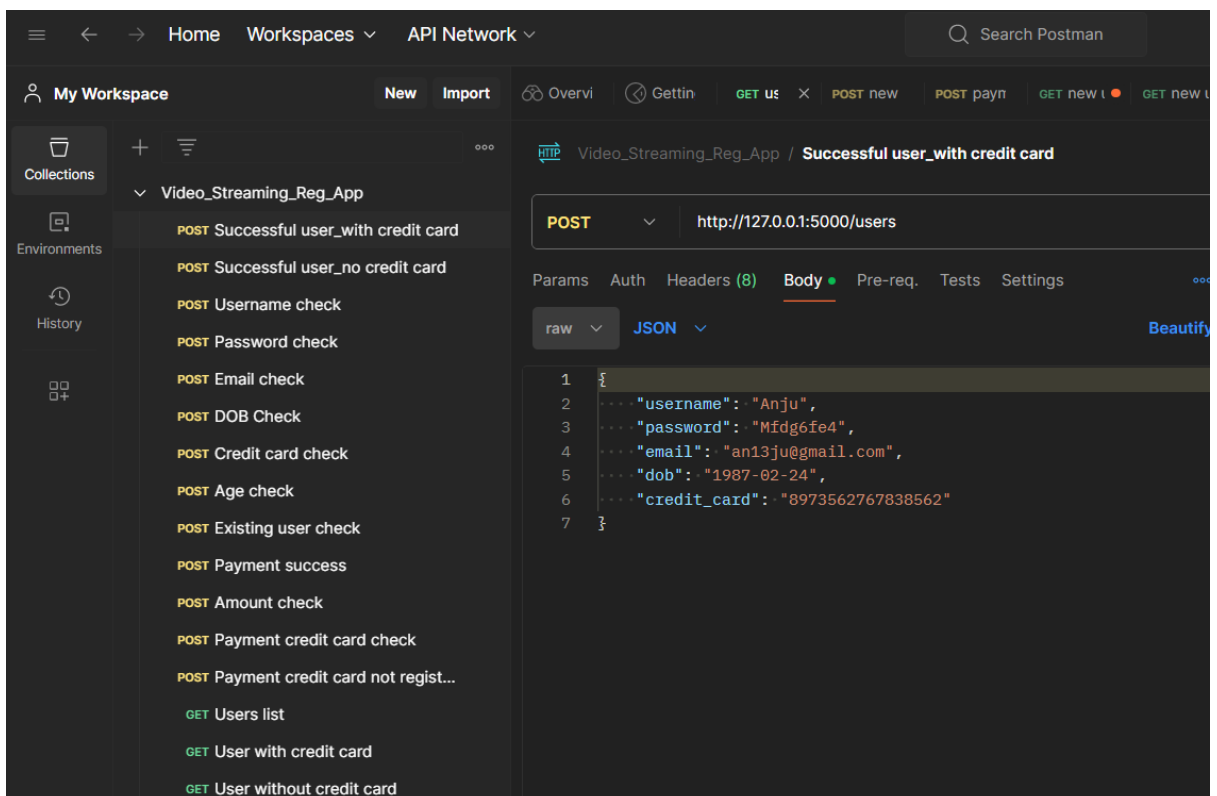
Steps to test the API endpoints:

Once the Flask application is running, use Postman to test the API endpoints.

1. Open the Postman application.
2. Import the Postman collection by clicking on the "Import" button in collections and select the collection file `Video_Streaming_Reg_App.postman_collection`.

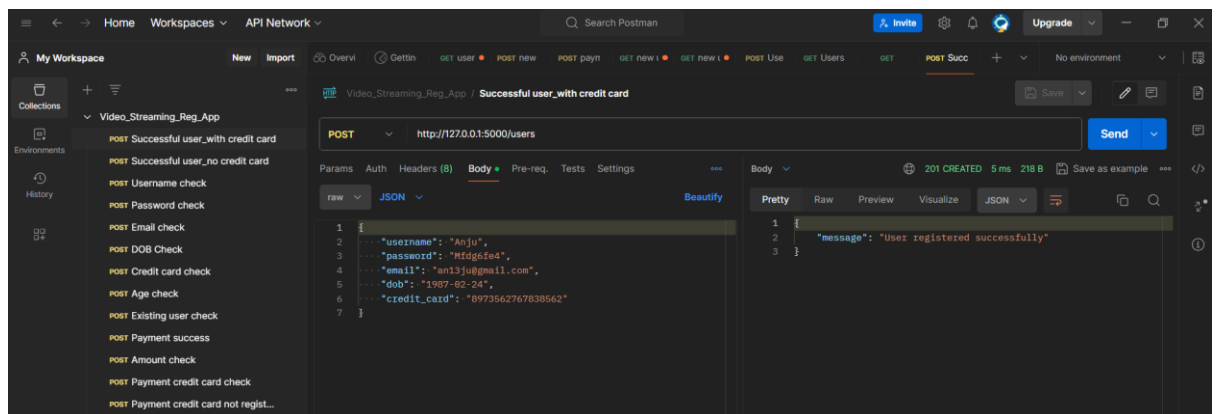


3. Once the collection is imported, there will be a list of requests organized within folders. Click on first request.



4. Review the details of the selected request, including the request method, URL, and body parameters.
5. Click the "Send" button to execute the selected request. Postman will send the request to the Flask application.

- Postman will display the response received from the Flask application, including the response body, status code, etc.



- Repeat the above steps for other requests in the Postman collection to test various scenarios and endpoints of the Flask application.