To launch the code, you need to follow these steps:

Open the folder and navigate to the app.py module.

Start the code by running the app.py file. This will generate an output similar to the following:

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/

Click on the provided link, usually http://127.0.0.1:5000/, to open the application in your web browser.

Please note that the provided link (http://127.0.0.1:5000/) is the address where your Flask application is running locally. In a production environment, it's recommended to use a production-ready WSGI server instead of the built-in development server.

Program description:

HTML Page and User Authentication:

The user engagement begins with a well-designed HTML page facilitating login/signup. Upon successful authentication, the user's details are securely stored in a dedicated database, ensuring a secure and personalized experience.

User Interaction and Destination Selection:

After authentication, users are prompted to indicate their familiarity with a travel destination.

If users are uncertain (responding with "no"), they are seamlessly directed to an HTML-based questionnaire. Java functions collect their preferences and transmit the data to a Python script.

City Recommendations and Database Interaction:

The Python script, interfacing with SQLite, processes the questionnaire responses. Based on this information, it generates a curated list of cities that align with the user's preferences.

Return to HTML Page with Recommendations:

The list of recommended cities is then seamlessly sent back to the HTML page through Java functions. Users are presented with a dynamic selection of destinations derived from their questionnaire responses.

Travel Planning Page:

Users can choose a city from the recommendations and are redirected to a dynamic HTML page. Here, they input specific travel details, including budget, destination, departure location, number of travelers, start and end dates, and activity preferences.

Java Functions for Communication:

The entered details are transmitted to the Python code through Java functions, ensuring a smooth flow of information between the HTML page and the Python script.

Python Code Execution and Recommendations:

Leveraging BeautifulSoup and Selenium, the Python code executes travel planning tasks. It interacts with various websites, considering the user's budget constraints, to provide recommendations. This includes three flight options, three hotel suggestions, and a detailed activity itinerary.

Return of Results to HTML Page:

The Python script sends the processed results back to the HTML page through Java functions. This seamless communication ensures that the user receives personalized recommendations on the HTML page.

Display of Recommendations:

The HTML page displays the travel recommendations in a user-friendly format. Users can review and select from the presented options, including flights, hotels, and activities, aligning with their specified criteria.

By seamlessly integrating HTML, Java, and Python components, this system offers users a comprehensive and interactive travel planning experience.