The following are lessons learned during the development of the different Python components created in order to interact with the OCI Cohere Chat endpoint.

**1. Set Up the Environment Variables:**

* We started by ensuring that all necessary environment variables were correctly set in the .env file. These included:
  + OCI\_COMPARTMENT\_ID
  + OCI\_TENANCY\_ID
  + OCI\_USER\_ID
  + OCI\_FINGERPRINT
  + OCI\_REGION
  + OCI\_PRIVATE\_KEY\_PATH
  + OCI\_PROFILE
  + OCI\_MODEL\_ID
* **Issue Encountered**: We noticed the .env file had to be properly named (as .env rather than .env.txt) and correctly referenced within the Python code. This is a common error when creating files.

**2. Fixing the Python Script to Load Environment Variables:**

* We explicitly loaded the environment variables using the python-dotenv package.
* We fixed any path issues with the .env file by specifying its full path in the Python script:
* dotenv\_path = 'C:/Users/XXX/.env' # Absolute path to .env file
* load\_dotenv(dotenv\_path=dotenv\_path)
* We also made sure the OCI config path (~/.oci/config) was expanded and correctly referenced.

**3. Verifying Environment Variables:**

* We added print statements to debug and check if the environment variables were being correctly loaded into the script.
* **Debugging Step**: If variables like OCI\_REGION, OCI\_MODEL\_ID, and OCI\_PRIVATE\_KEY\_PATH were None, we were alerted and took steps to fix the loading process.
* This was necessary because when calling the Flask API from Postman there were errors, and it was due to the environment variables not being pulled correctly into the python API.

**4. Fixing the Endpoint URL:**

* The endpoint for the API call was dynamically created using the OCI\_REGION:
* endpoint = f"https://inference.generativeai.{region}.oci.oraclecloud.com"
* This ensures that we use the correct region for the API.

**5. Setting Up the OCI SDK Client:**

* The Python SDK was used to create the oci.generative\_ai\_inference.GenerativeAiInferenceClient:
* generative\_ai\_inference\_client = oci.generative\_ai\_inference.GenerativeAiInferenceClient(config=config, service\_endpoint=endpoint)
* This client connects to the OCI Generative AI service using the configuration loaded from the .env file.

**6. Error Handling and Debugging:**

* We added error handling using a try-except block in the API route to ensure we could identify and return errors if anything went wrong with the request to the OCI service.

**7. Testing the Flask API:**

* With the Flask API running on http://127.0.0.1:5000, we tested it using Postman by sending a POST request to the /generate endpoint with a message in the JSON payload.
* **Correct Payload**: The correct payload format was:
* {
* "message": "Hello, are you there?"
* }
* Errors were faced because the content-type header had not been set.

**8. Fixing the 'ChatResult' Error:**

* **Issue Encountered**: Initially, when the ChatResult object was being accessed, we encountered the error "ChatResult object is not subscriptable."
* **Solution**: We reviewed the response object structure and fixed the way we accessed the response data:
* response\_text = chat\_response.data['chat\_response']['text']

**9. Final Test:**

* After making the necessary adjustments, the endpoint successfully returned the AI's response to the message, and Postman was able to receive the response correctly.