GEN AI- ASSIGNMENT MOE

NAME:DELISHA RIYONA DSOUZA
SECTION C
SRN: PES2UG23CS166

Outputs

```
(venv) delishariyonadsouza@Delishas-MacBook-Air MoE_Assignment % python3 moe_router.py

Ask Support (exit to stop): My python code has IndexError.

Router Selected: technical

Response:
 # Step-by-step analysis of the problem:
1. **Understanding the Error**: An `IndexError` in Python occurs when you try to access an element in a sequence (like a list, tuple, or string) using an i
ndex that is out of range. This means the index you're trying to access doesn't exist in the sequence.
2. **Possible Causes**:
    - **Index Out of Bounds**: Trying to access an index that is greater than or equal to the length of the sequence.
    - **Negative Indexing Error**: While Python supports negative indexing (counting from the end), if the absolute value of the negative index exceeds the
 length of the sequence, it will raise an `IndexError`.
    - **Empty Sequence**: Trying to access any index of an empty sequence.

# Fixed solution:
To provide a fixed solution, we need to see your code. However, I'll give you a general example of how to fix an `IndexError` in a common scenario:

```python
def access_list_element(my_list, index):
    # Check if the list is not empty
    if len(my_list) == 0:
        print("The list is empty.")
        return

    # Check if the index is within bounds
    if index < 0 or index >= len(my_list):
        print("Index out of range.")
        return

    # Now it's safe to access the element
    print(my_list[index])

# Example usage
my_list = [1, 2, 3, 4, 5]
access_list_element(my_list, 2)  # Output: 3
access_list_element(my_list, 10)  # Output: Index out of range.
access_list_element([], 0)  # Output: The list is empty.
```

Explanation of changes:
- **Added Checks**: Before accessing the list element, we check if the list is empty and if the index is within the valid range.
- **Handled Edge Cases**: We handle the cases where the list is empty or the index is out of range, preventing the `IndexError`.

Tests and example uses:
- Test with a valid index: `access_list_element([1, 2, 3], 1)` should print `2`.
- Test with an out-of-range index: `access_list_element([1, 2, 3], 5)` should print `Index out of range.`
- Test with an empty list: `access_list_element([], 0)` should print `The list is empty.`

Please provide your code for a more specific solution.
```

```
Ask Support (exit to stop): I was charged twice.

Router Selected: billing

Response:
 I'm so sorry to hear that you were charged twice. I can imagine how frustrating that must be for you. Please know that I'm here to help and support you in
 resolving this issue.

To initiate the refund process, I'll need to investigate this further. Could you please provide me with more details about the duplicate charge, such as th
e date of the transaction, the amount, and the method of payment used? This information will help me to look into the matter and ensure that you receive a
prompt refund for the incorrect charge.

In the meantime, I want to assure you that our company has a clear refund policy in place to protect our customers in situations like this. If we find that
 an error has occurred, we will promptly refund the duplicate charge to your original payment method.

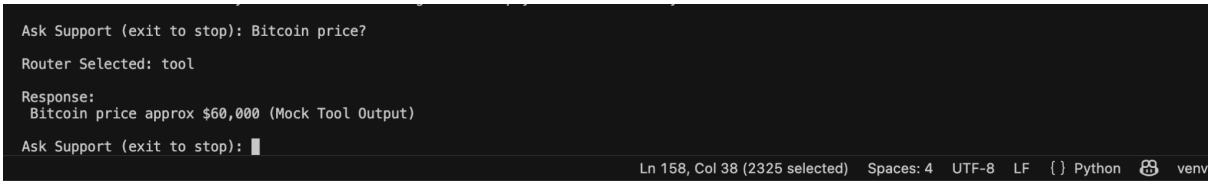
Please be assured that we value your business and appreciate your patience and cooperation as we work to resolve this issue. If you have any questions or c
oncerns, please don't hesitate to reach out to me directly. I'm here to help and support you throughout this process.

Let's work together to get this resolved as quickly as possible. Can you please provide me with the necessary information so I can proceed with the refund?
```

```
Ask Support (exit to stop): Hello.

Router Selected: general

Response:
 Hello. It's nice to meet you. Is there something I can help you with or would you like to chat?
```



Code moe_router.py

```
from groq import Groq
from dotenv import load_dotenv
import os


load_dotenv()

client = Groq(
   api_key=os.getenv("GROQ_API_KEY")
)



MODEL_CONFIG = {

   "technical": {

       "system_prompt":
       """You are a Technical Expert.
Give precise debugging help.
Provide coding solutions."""
   },

   "billing": {

       "system_prompt":
       """You are a Billing Expert.
Be empathetic.
Explain refund policies professionally."""
   },

   "general": {

       "system_prompt":
       """You are a friendly assistant."""
   }

}
```

```
def route_prompt(user_input):

   router_prompt=f"""
Classify into:

technical
billing
general

Return ONLY the word.

Query:
{user_input}
"""

   response=client.chat.completions.create(

       model="llama-3.3-70b-versatile",

       temperature=0,

       messages=[

           {
               "role":"system",
               "content":"You are strict classifier."
           },

           {
               "role":"user",
               "content":router_prompt
           }

       ]

   )

   category=response.choices[0].message.content.strip().lower()

   return category
```

```
def call_expert(category,user_input):
   system_prompt=MODEL_CONFIG[category]["system_prompt"]
   response=client.chat.completions.create(
       model="llama-3.3-70b-versatile",
       temperature=0.7,
       messages=[
           {
               "role":"system",
               "content":system_prompt
           },
           {
               "role":"user",
               "content":user_input
           }
       ]
   )
   return response.choices[0].message.content



def bitcoin_price_tool():
   return "Bitcoin price approx $60,000 (Mock Tool Output)"



def process_request(user_input):
   if "bitcoin" in user_input.lower():
       print("\nRouter Selected: tool")
```

```
        return bitcoin_price_tool()

    category=route_prompt(user_input)

    print("\nRouter Selected:",category)

    if category not in MODEL_CONFIG:

        category="general"

    return call_expert(category,user_input)




if __name__=="__main__":

    while True:

        query=input("\nAsk Support (exit to stop): ")

        if query.lower()=="exit":

            break

        answer=process_request(query)

        print("\nResponse:\n",answer)
```

Along with a .env file with the api key