

GenAI Unit 2 Lab – Assignment

Name	Dhanya Prabhu
SRN	PES2UG23CS169
Section	C

Output Screenshots with explanations:

1. Testing the router

```
print(route_prompt("My Python code has an IndexError"))
print(route_prompt("I was charged twice"))
print(route_prompt("Hello there!"))

technical
billing
general
```

The router function classifies user queries into predefined categories (technical, billing, general) using deterministic inference. This ensures consistent intent detection before forwarding the query to the appropriate expert.

2. Testing the experts

a. Technical Expert

```
# Technical Expert Test
response = client.chat.completions.create(
    model=BASE_MODEL,
    temperature=MODEL_CONFIG["technical"]["temperature"],
    messages=[
        {"role": "system", "content": MODEL_CONFIG["technical"]["system_prompt"]},
        {"role": "user", "content": "My Python code gives an IndexError."}
    ]
)
print(response.choices[0].message.content)

***Understanding IndexError in Python***
An 'IndexError' in Python occurs when you try to access an element in a sequence (like a list, tuple, or string) using an index that does not exist. This can happen when you're trying to access an element at an index that's out of range.

**Example Code**
Here's a simple example:
```python
my_list = [1, 2, 3]
print(my_list[3]) # IndexError: list index out of range
```
In this example, we're trying to access the element at index 3 in the list '[1, 2, 3]', but since the list only has 3 elements (at indices 0, 1, and 2), this raises an 'IndexError'.
**Solution**
To avoid 'IndexError', you can use one of the following approaches:
1. **Check the length of the sequence**: Before accessing an element, check if the index is within the valid range using the `len()` function.
```python
my_list = [1, 2, 3]
index = 3
if index < len(my_list):
 print(my_list[index])
else:
 print("Index out of range")
```
2. **Use a try-except block**: Wrap the code that accesses the element in a try-except block to catch the 'IndexError' exception and handle it accordingly.
```python
my_list = [1, 2, 3]
index = 3
try:
 print(my_list[index])
except IndexError:
 print("Index out of range")
```
3. **Use the `in` operator**: If you're trying to check if an element exists in a sequence, use the `in` operator instead of accessing it by index.
```python
my_list = [1, 2, 3]
if index in my_list:
 print(f"Element {index} exists in the list")
else:
 print("Element does not exist in the list")
```
**Best Practice**
To avoid 'IndexError', always check the length of the sequence or use a try-except block to handle the exception.

**Additional Information**
* If you're working with a list, consider using list indices starting from 0 (e.g., `my_list[0]`, `my_list[1]`, etc.).
* If you're working with a dictionary, use the `keys()` method to get a list of keys and then access the corresponding value using the `get()` method or the `[]` operator.
* If you're working with a NumPy array, use the `ndarray` indexing syntax (e.g., `my_array[i, j]`) to access elements safely.

I hope this helps you fix the 'IndexError' in your Python code. If you have any further questions or concerns, feel free to ask!
```

The technical expert uses a code-focused system prompt to provide structured debugging guidance. The response emphasizes precision and technical clarity, demonstrating domain-specific specialization.

b. Billing Expert

```

# Billing Expert Test
response = client.chat.completions.create(
    model=BASE_MODEL,
    temperature=MODEL_CONFIG["billing"]["temperature"],
    messages=[
        {"role": "system", "content": MODEL_CONFIG["billing"]["system_prompt"]},
        {"role": "user", "content": "I was charged twice this month."}
    ]
)

print(response.choices[0].message.content)

I'm so sorry to hear that you were charged twice this month. I understand how frustrating and confusing that can be.

To assist you further, can you please provide me with some more information about the duplicate charges? Could you tell me:

1. What is the date of the first and second charge?
2. What services or products were you charged for on both occasions?
3. Is there a corresponding invoice or billing statement with the duplicate charges?

Your information will help me investigate this issue and resolve it as quickly as possible.

```

The technical expert uses a code-focused system prompt to provide structured debugging guidance. The response emphasizes precision and technical clarity, demonstrating domain-specific specialization.

c. General Expert

```

● # General Expert Test
response = client.chat.completions.create(
    model=BASE_MODEL,
    temperature=MODEL_CONFIG["general"]["temperature"],
    messages=[
        {"role": "system", "content": MODEL_CONFIG["general"]["system_prompt"]},
        {"role": "user", "content": "What services do you offer?"}
    ]
)

print(response.choices[0].message.content)

... I'm happy to help you with a wide range of services. Here are some of the things I can assist you with:

1. **General Knowledge**: I can provide information on various topics such as history, science, technology, health, and more.
2. **Language Translation**: If you need help translating text or want to learn a new language, I can assist you with that.
3. **Text Summarization**: If you have a long piece of text and want a concise summary, I can help you with that.
4. **Conversation**: We can have a conversation on any topic you'd like, and I'll do my best to respond thoughtfully and helpfully.
5. **Writing Assistance**: If you need help with writing a paper, essay, or other document, I can assist you with research, organization, and more.
6. **Calculator and Math**: I can perform basic arithmetic operations, help with algebra, and more.
7. **Trivia and Games**: We can play games like "Would you rather...", "Two truths and a lie", and more.
8. **Jokes and Humor**: If you need a laugh, I can share some jokes and humorous stories.
9. **Wordplay and Puzzles**: I can help you with word searches, crosswords, and other puzzles.
10. **Learning and Education**: I can assist you with learning a new skill or subject, such as a language, hobby, or academic topic.

What can I help you with today?

```

The general expert handles non-specialized queries with a conversational and helpful tone. This serves as the fallback expert in the MoE architecture.

3. Integrated testing

```

queries = [
    "My login API returns 500 error.",
    "My annual plan was cancelled but I was still billed.",
    "Do you offer student discounts?"
]

for q in queries:
    category, answer = process_request(q)
    print("\nUser query:", q)
    print("Routed To:", category)
    print("Response:", answer)

# Define a custom error handler for unhandled exceptions
@app.errorhandler(Exception)
def unhandled_exception(e):
    logging.error(f"Unhandled Exception: {e}")
    return jsonify({"error": "Unhandled Exception"}), 500

# Define the login API
@app.route('/login', methods=['POST'])
def login():
    try:
        # Get the username and password from the request
        username = request.json.get('username')
        password = request.json.get('password')

        # Validate the username and password
        if not username or not password:
            return jsonify({"error": "Invalid username or password"}), 401

        # Query the database to get the user
        user = User.query.filter_by(username=username).first()
        if not user or not user.check_password(password):
            return jsonify({"error": "Invalid username or password"}), 401

        # Generate a JWT token for the user
        token = jwt.encode({'user_id': user.id}, "secret_key", algorithm="HS256")

        # Return the token
        return jsonify({"token": token})
    except Exception as e:
        logging.error(f"Error logging in: {e}")
        return jsonify({"error": "Internal Server Error"}), 500

if __name__ == '__main__':
    app.run(debug=True)

This example shows how to enable debug mode, define custom error handlers for unhandled exceptions, and log errors. It also shows how to handle the login API request.

If you're still having trouble, please provide more information about your API, including the programming language and framework you're using, the error messages you're seeing, and any relevant code.

```

```
User Query: My annual plan was cancelled but I was still billed.
Routed To: Billing
Response: I'm so sorry to hear that you were billed after your annual plan was cancelled. I can understand how frustrating that must be for you.

To look into this further, I'll need to review your account information. Could you please provide me with your account number or the email address associated with your account? Additionally, can you confirm the date you cancelled your annual plan and the date you were billed?

Also, I'd like to let you know that our policy states that if a customer cancels their plan before the renewal date, they will receive a refund of the remaining contract term. However, if the cancellation occurs after the renewal date, the customer will still be billed for the current period.

I'm here to help resolve this issue for you, and I'll do my best to find out what happened and see if there's anything we can do to rectify the situation.

User Query: Do you offer student discounts?
Routed To: general
Response: Yes, we do offer student discounts to eligible students. To qualify, you'll need to provide a valid student ID or a university email address from a recognized institution. This will verify your student status and allow you to receive the discount.

We also partner with several student organizations and educational institutions to offer exclusive discounts and promotions. If you're a student, I'd be happy to help you find out more about our student discount program and see what savings you can enjoy.

What specific products or services are you interested in purchasing, and I can let you know if we have any student discounts available?
```

The general expert handles non-specialized queries with a conversational and helpful tone. This serves as the fallback expert in the MoE architecture.

4. Bonus Challenge

```
def get_bitcoin_price():
    return "The current price of Bitcoin is $65,000."


def process_request(user_input):
    category = route_prompt(user_input)

    if "bitcoin" in user_input.lower():
        return "crypto", get_bitcoin_price()

    if category not in MODEL_CONFIG:
        category = "general"

    expert = MODEL_CONFIG[category]

    response = client.chat.completions.create(
        model=BASE_MODEL,
        temperature=expert["temperature"],
        messages=[
            {"role": "system", "content": expert["system_prompt"]},
            {"role": "user", "content": user_input}
        ]
    )

    return category, response.choices[0].message.content


category, answer = process_request("What is the current price of Bitcoin?")
print("Routed To:", category)
print("Response:", answer)
```

Routed To: crypto
Response: The current price of Bitcoin is \$65,000.

The bonus implementation introduces tool usage within the MoE framework. Cryptocurrency-related queries are routed to a Python function instead of the LLM, demonstrating hybrid AI architecture and extensibility.