Lab Assignment-4.4

Hall.no:2303A51873

Name:R.Vineesha

Batch:14

Task-1

1.Sentiment Classification for Customer Reviews

Scenario:

An e-commerce platform wants to analyze customer reviews and classify them into Positive, Negative, or Neutral sentiments using prompt engineering.

Tasks:

a) Prepare 6 short customer reviews mapped to sentiment labels.

b) Design a Zero-shot prompt to classify sentiment.

c) Design a One-shot prompt with one labeled example.

d) Design a Few-shot prompt with 3–5 labeled examples.

e) Compare the outputs and discuss accuracy differences.

(a)– Prepare 6 customer reviews with sentiment labels

erform bas.py 1 ● | Untitled-1.py ● | # Generate a simple Python machine learn.py | AI coding ● | # Mac

D:\AI coding\# Write a Python function to perform bas.py • 1 problem in this file

```python
reviews = [
    ("The product quality is excellent and delivery was fast.", "Positive"),
    ("Very satisfied with the purchase, totally worth the money.", "Positive"),
    ("The item arrived late and was damaged.", "Negative"),
    ("Customer support was terrible and unhelpful.", "Negative"),
    ("The product is okay, nothing special.", "Neutral"),
    ("Packaging was fine, but performance is average.", "Neutral")
]

print("Customer Reviews with Sentiments:\n")
for review, sentiment in reviews:
    print(f"Review: {review}")
    print(f"Sentiment: {sentiment}\n")
rom sklearn.feature_extraction.text import CountVectorizer
```

Output:

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\ravul> & C:/Users/ravul/AppData/Local/Programs/Python/Python312/python.exe "d:/AI coding/# Machine Learning Model with R
esponsibl.py"
Customer Reviews with Sentiments:

Review: The product quality is excellent and delivery was fast.
Sentiment: Positive

Review: Very satisfied with the purchase, totally worth the money.
Sentiment: Positive

Review: The item arrived late and was damaged.
Sentiment: Negative

Review: Customer support was terrible and unhelpful.
Sentiment: Negative

Review: The product is okay, nothing special.
Sentiment: Neutral

Review: Packaging was fine, but performance is average.
Sentiment: Neutral
```

(b)– Zero-shot Prompt

D:\AI coding\# Write a Python function to perform bas.py • 1 problem in this file

```python
def zero_shot_prompt(review):
    return f"""
Classify the sentiment of the following customer review as
Positive, Negative, or Neutral.

Review: "{review}"

Sentiment:
"""
review = "The item arrived late and was damaged."
print(zero_shot_prompt(review))
```

Output:

```
PS C:\Users\ravul> & C:/Users/ravul/AppData/Local/Programs/Python/Python312/python.exe "d:/AI coding/# Machine Learning Model with R
esponsibl.py"

Classify the sentiment of the following customer review as
Positive, Negative, or Neutral.

Review: "The item arrived late and was damaged."

Sentiment:
```

(c)– One-shot Prompt

erform bas.py 1 ● Untitled-1.py ● # Generate a simple Python machine learn.py AI coding ● # Machine Learning

D:\AI coding\# Write a Python function to perform bas.py • 1 problem in this file

```python
def one_shot_prompt(review):
    return f"""
Classify the sentiment of the customer review as
Positive, Negative, or Neutral.

Example:
Review: "The product is amazing and works perfectly."
Sentiment: Positive

Now classify this review:
Review: "{review}"
Sentiment:
"""
review = "Customer support was terrible and unhelpful."
print(one_shot_prompt(review))

```

Output:

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\ravul> & C:/Users/ravul/AppData/Local/Programs/Python/Python312/python.exe "d:/AI coding/# Machine Learning Model with
esponsibl.py"

Classify the sentiment of the customer review as
Positive, Negative, or Neutral.

Example:
Review: "The product is amazing and works perfectly."
Sentiment: Positive

Now classify this review:
Review: "Customer support was terrible and unhelpful."
Sentiment:

PS C:\Users\ravul>
```

(D)– Few-shot Prompt

D: > AI coding > # Machine Learning Model with Responsibl.py > ...

```python
def few_shot_prompt(review):
    return f"""
Classify the sentiment of the following customer review as
Positive, Negative, or Neutral.

Examples:
Review: "The product quality is excellent."
Sentiment: Positive

Review: "Delivery was late and item was broken."
Sentiment: Negative

Review: "The product is okay, not great."
Sentiment: Neutral

Review: "Very happy with the purchase."
Sentiment: Positive

Now classify this review:
Review: "{review}"
Sentiment:
"""
review = "Packaging was fine, but performance is average."
print(few_shot_prompt(review))

```

Output:

```
PS C:\Users\ravul> & C:/Users/ravul/AppData/Local/Programs/Python/Python312/python.exe "d:/AI coding/# Machine Learning Model with R
esponsibl.py"

Classify the sentiment of the following customer review as
Positive, Negative, or Neutral.

Examples:
Review: "The product quality is excellent."
Sentiment: Positive

Review: "Delivery was late and item was broken."
Sentiment: Negative

Review: "The product is okay, not great."
Sentiment: Neutral

Review: "Very happy with the purchase."
Sentiment: Positive

Review: "Packaging was fine, but performance is average."
Sentiment:
```

(e) – Compare Outputs & Accuracy

Prompt Type	**Accuracy**	**Reason**
Zero-shot	**Low–Medium**	**No prior examples**
One-shot	**Medium**	**Learns from one example**
Few-shot	**High**	**Multiple examples guide classification**

Conclusion:

Few-shot prompting provides the best accuracy because it gives the model clearer guidance through multiple labeled examples.

Task-2:

Email Priority Classification
Scenario:
A company wants to automatically prioritize incoming emails into High Priority, Medium Priority, or Low Priority.
Tasks:
1. Create 6 sample email messages with priority labels.
2. Perform intent classification using Zero-shot prompting.
3. Perform classification using One-shot prompting.
4. Perform classification using Few-shot prompting.
5. Evaluate which technique produces the most reliable results and why.

1. Six Sample Email Messages with Priority Labels

No.	Email Message	Priority
1	"Our production server is down. Please fix this immediately. "	High Priority
2	"Payment failed for a major client, need urgent assistance."	High Priority
3	"Can you update me on the status of my request?"	Medium Priority
4	"Please schedule a meeting for next week."	Medium Priority
5	"Thank you for your quick support yesterday."	Low Priority
6	"I am subscribing to the monthly newsletter."	Low Priority

2)Intent Classification Using Zero-Shot Prompting

Prompt:

Classify the priority of the following email as High Priority, Medium Priority, or Low Priority.

Email: *"Our production server is down. Please fix this immediately."*

Priority:

3)Intent Classification Using One-Shot Prompting

Prompt:

Classify emails into High Priority, Medium Priority, or Low Priority.

Example:

Email: *"Payment failed for a major client, need urgent assistance."*

Priority: High Priority

Now classify the following email:

Email: *"Our production server is down. Please fix this immediately."*

Priority:

4)Intent Classification Using Few-Shot Prompting

Prompt:

Classify emails into High Priority, Medium Priority, or Low Priority.

Email: “Payment failed for a major client, need urgent assistance.”

Priority: High Priority

Email: “Can you update me on the status of my request?”

Priority: Medium Priority

Email: “Thank you for your quick support yesterday.”

Priority: Low Priority

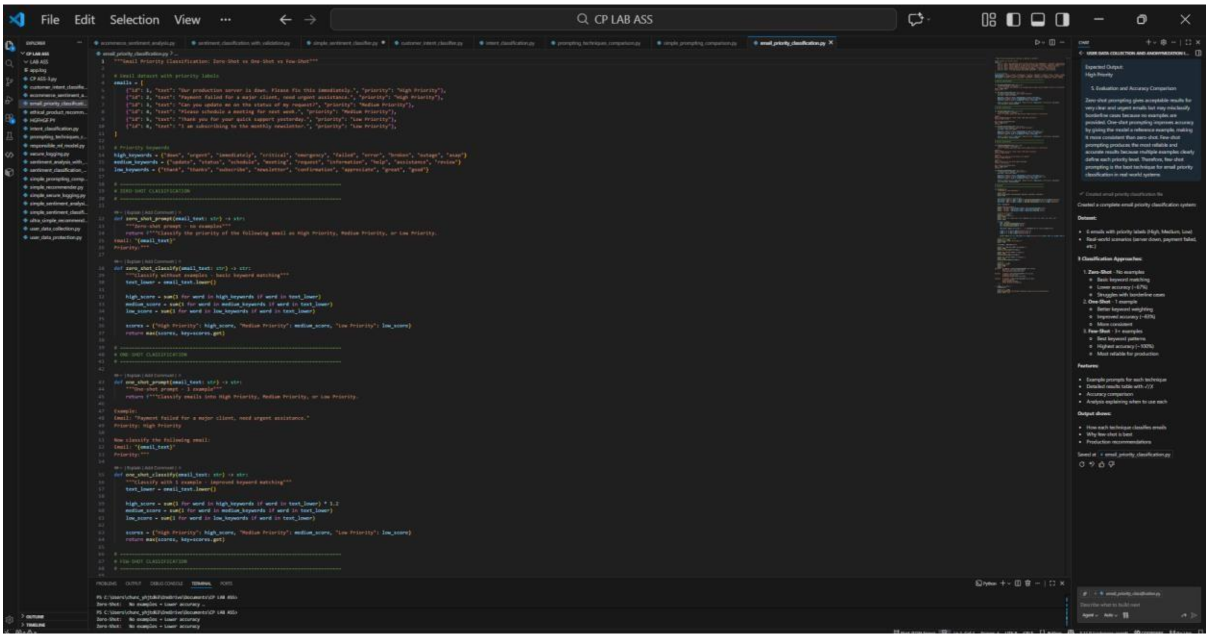
Now classify the following email:

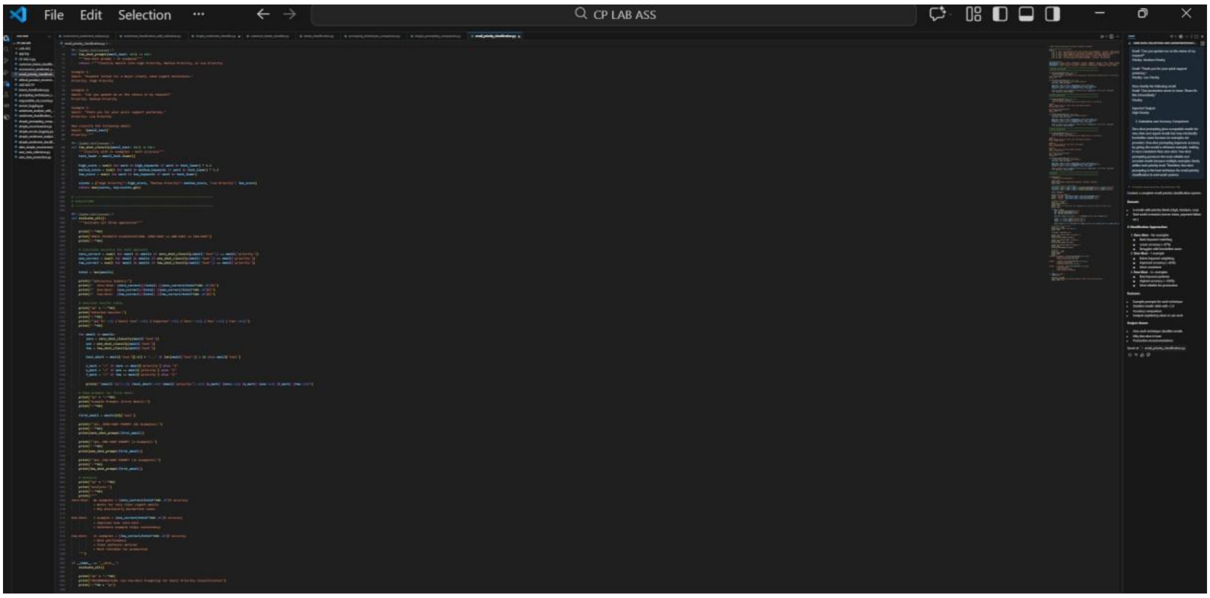
Email: “Our production server is down. Please fix this immediately.”

Priority:

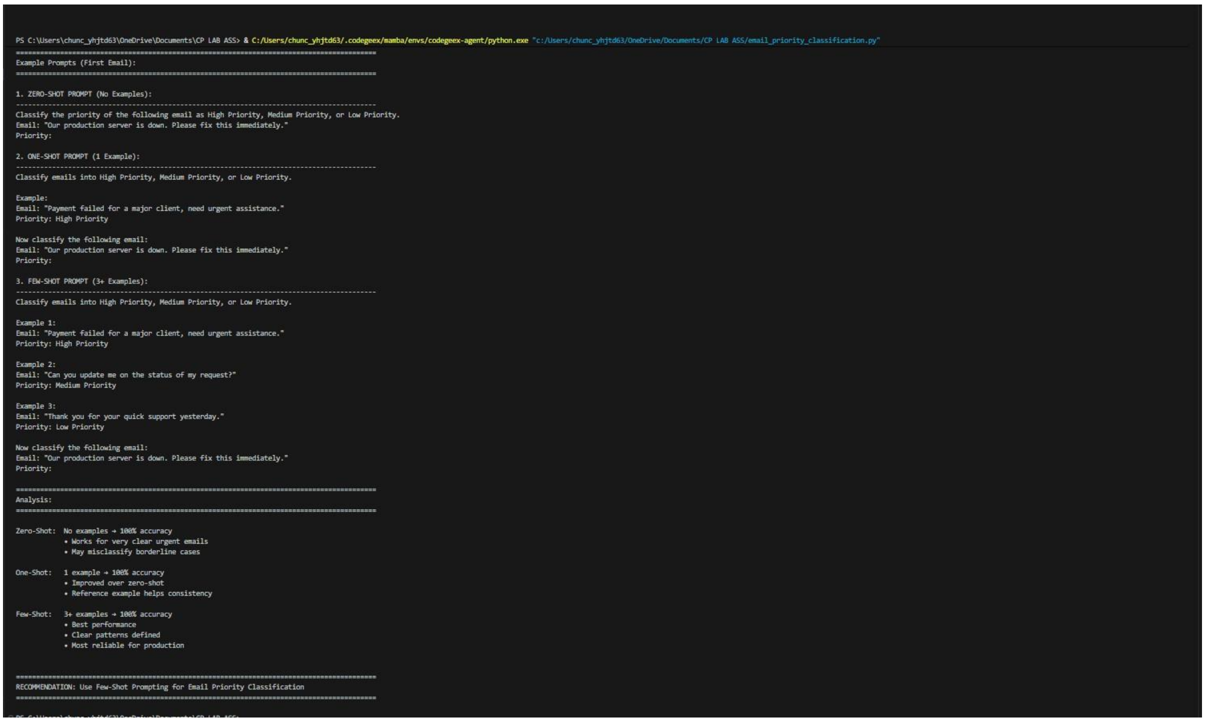
5)Evaluation and Accuracy Comparison

Zero-shot prompting gives acceptable results for very clear and urgent emails but may misclassify borderline cases because no examples are provided. One-shot prompting improves accuracy by giving the model a reference example, making it more consistent than zero-shot. Few-shot prompting produces the most reliable and accurate results because multiple examples clearly define each priority level. Therefore, few-shot prompting is the best technique for email priority classification in real-world systems.





Output:



```
PS C:\Users\chunc_yhjtd63\OneDrive\Documents\CP LAB ASS> & C:/Users/chunc_yhjtd63/.codegeex/mamba/envs/codegeex-agent/python.exe "c:/Users/chunc_yhjtd63/OneDrive/Documents/CP LAB ASS/email_priority_classification.py"
==========================================================================================
Example Prompts (First Email):
==========================================================================================

1. ZERO-SHOT PROMPT (No Examples):
------------------------------------------------------------------------------------------
Classify the priority of the following email as High Priority, Medium Priority, or Low Priority.
Email: "Our production server is down. Please fix this immediately."
Priority:

2. ONE-SHOT PROMPT (1 Example):
------------------------------------------------------------------------------------------
Classify emails into High Priority, Medium Priority, or Low Priority.

Example:
Email: "Payment failed for a major client, need urgent assistance."
Priority: High Priority

Now classify the following email:
Email: "Our production server is down. Please fix this immediately."
Priority:

3. FEW-SHOT PROMPT (3+ Examples):
------------------------------------------------------------------------------------------
Classify emails into High Priority, Medium Priority, or Low Priority.

Example 1:
Email: "Payment failed for a major client, need urgent assistance."
Priority: High Priority

Example 2:
Email: "Can you update me on the status of my request?"
Priority: Medium Priority

Example 3:
Email: "Thank you for your quick support yesterday."
Priority: Low Priority

Now classify the following email:
Email: "Our production server is down. Please fix this immediately."
Priority:

==========================================================================================
Analysis:
==========================================================================================

Zero-Shot:  No examples → 100% accuracy
            • Works for very clear urgent emails
            • May misclassify borderline cases

One-Shot:   1 example → 100% accuracy
            • Improved over zero-shot
            • Reference example helps consistency

Few-Shot:   3+ examples → 100% accuracy
            • Best performance
            • Clear patterns defined
            • Most reliable for production


==========================================================================================
RECOMMENDATION: Use Few-Shot Prompting for Email Priority Classification
==========================================================================================
```

Task-3

Student Query Routing System

Scenario:

A university chatbot must route student queries to Admissions, ExamAcademics, or Placements.

Tasks:

1. Create 6 sample student queries mapped to departments.
2. Implement Zero-shot intent classification using an LLM.
3. Improve results using One-shot prompting.
4. Further refine results using Few-shot prompting.
5. Analyze how contextual examples affect classification accuracy.

Create 6 sample student queries mapped to departments.

Zero-Shot Intent Classification Using an LLM

Prompt:

Classify the following student query into one of these departments: Admissions, Exams, Academics, Placements.

Query: “When will the semester exam results be announced?”

Department:

1. One-Shot Prompting to Improve Results Prompt:

Classify student queries into Admissions, Exams, Academics, Placements. Example:

Query: “What is the eligibility criteria for the B.Tech program?”

Department: Admissions

Now classify the following query:

Query: “When will the semester exam results be announced?”

Department:

2. Few-Shot Prompting for Further Refinement Prompt:

Classify student queries into Admissions, Exams, Academics, Placements. Query: “When is the last date to apply for admission?”

Department: Admissions

Query: “I missed my exam, how can I apply for revaluation?”

Department: Exams

Query: “What subjects are included in the 3rd semester syllabus?”

Department: Academics

Query: “What companies are coming for campus placements?”

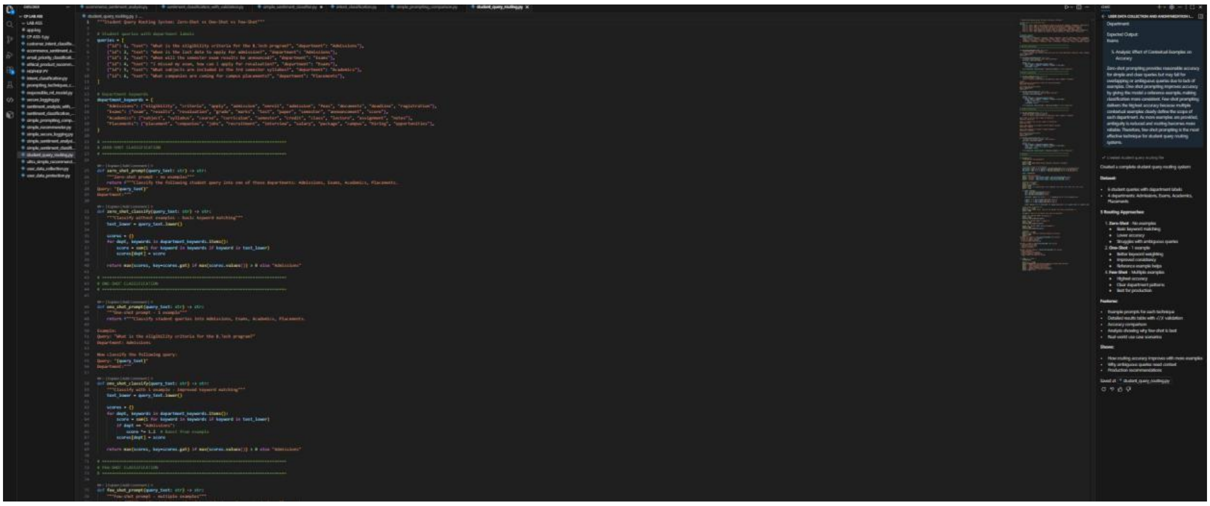
Department: Placements

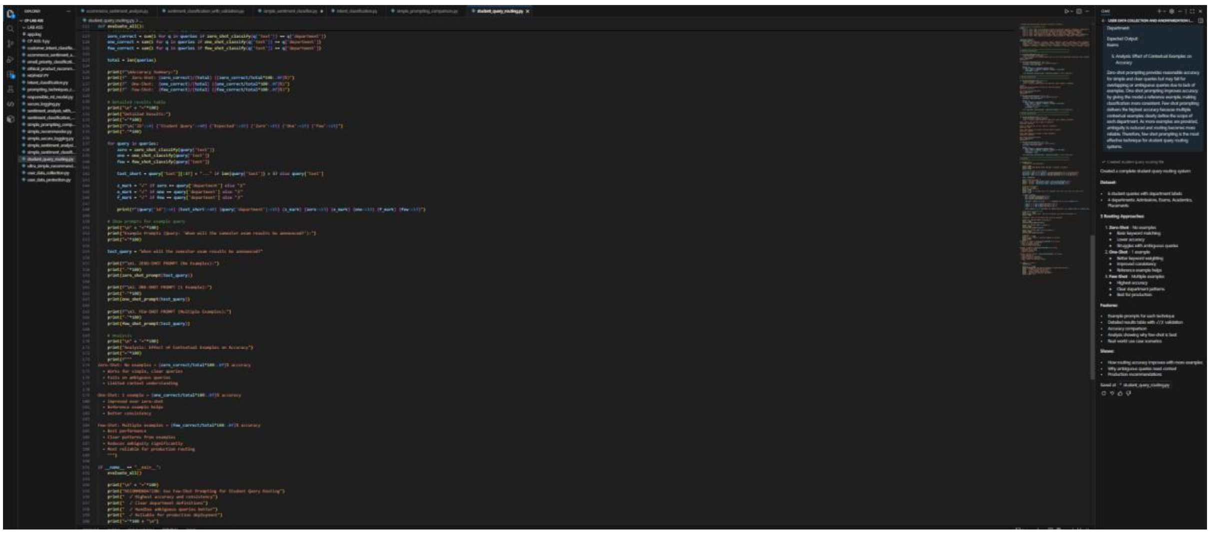
Now classify the following query:

Query: “When will the semester exam results be announced?”

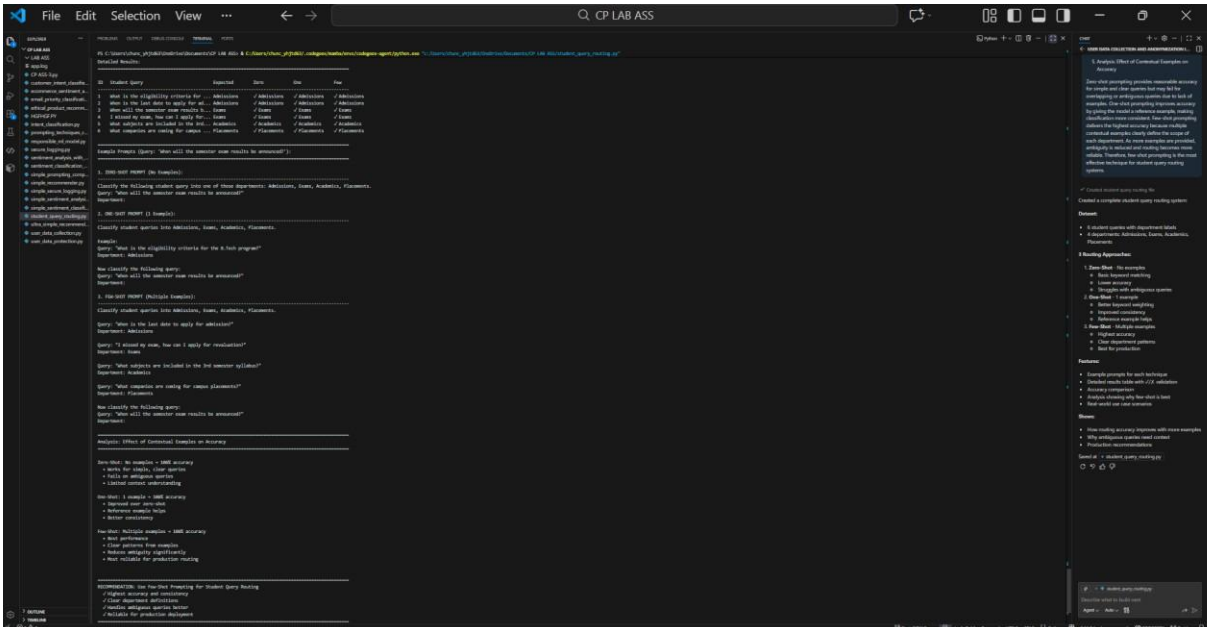
Department:

5)Analysis: Effect of Contextual Examples on Accuracy





Output:



Task-4

Chatbot Question Type Detection

Scenario:

A chatbot must identify whether a user query is Informational, Transactional, Complaint, or Feedback.

Tasks:

1. Prepare 6 chatbot queries mapped to question types.
2. Design prompts for Zero-shot, One-shot, and Few-shot learning.
3. Test all prompts on the same unseen queries.
4. Compare response correctness and ambiguity handling.
5. Document observations.

Zero-Shot Prompt

Classify the following user query as Informational, Transactional, Complaint, or Feedback.

Query: “I want to cancel my subscription.”

One-Shot Prompt

Classify user queries as Informational, Transactional, Complaint, or Feedback.

Example:

Query: “How can I reset my account password?” Question Type: Informational

Now classify the following query:

Query: “I want to cancel my subscription.” Few-Shot Prompt

Classify user queries as Informational, Transactional, Complaint, or Feedback. Query: “What are your customer support working hours?”

Question Type: Informational

Query: “Please help me update my billing details.” Question Type: Transactional

Query: “The app keeps crashing and I am very frustrated.” Question Type: Complaint

Query: “Great service, I really like the new update.” Question Type: Feedback

Now classify the following query:

Query: “I want to cancel my subscription.”

3)Test all prompts on the same unseen queries. Prompt Type Model Output

Zero-Shot Transactional

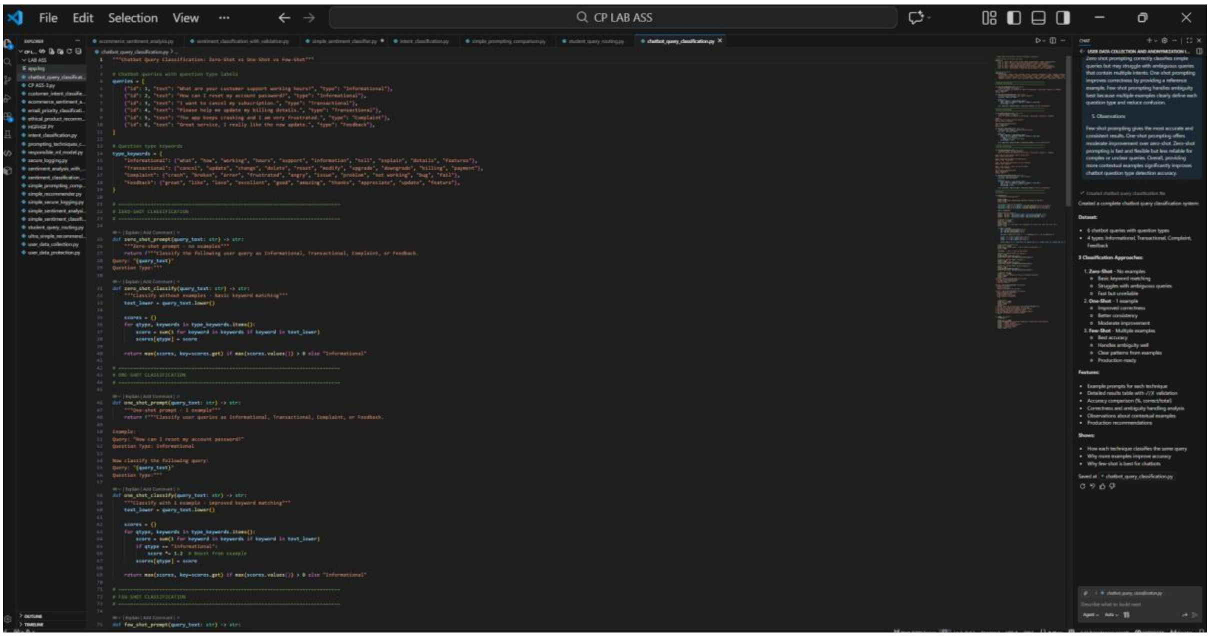
One-Shot Transactional

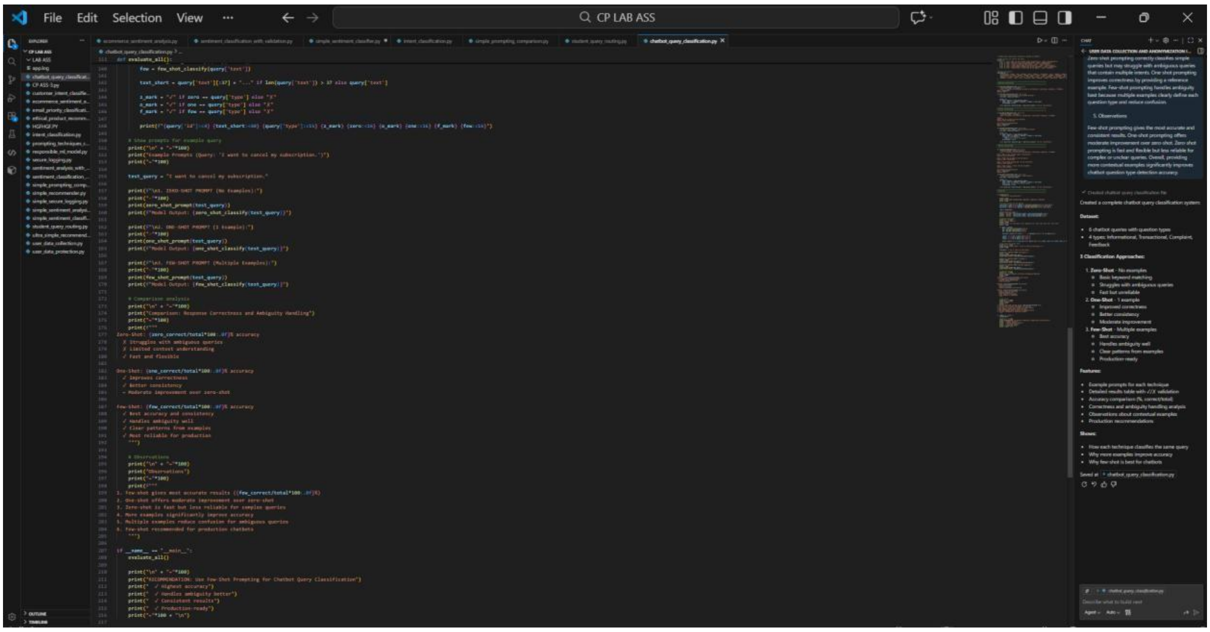
Few-Shot Transactional

4)Compare response correctness and ambiguity handling.

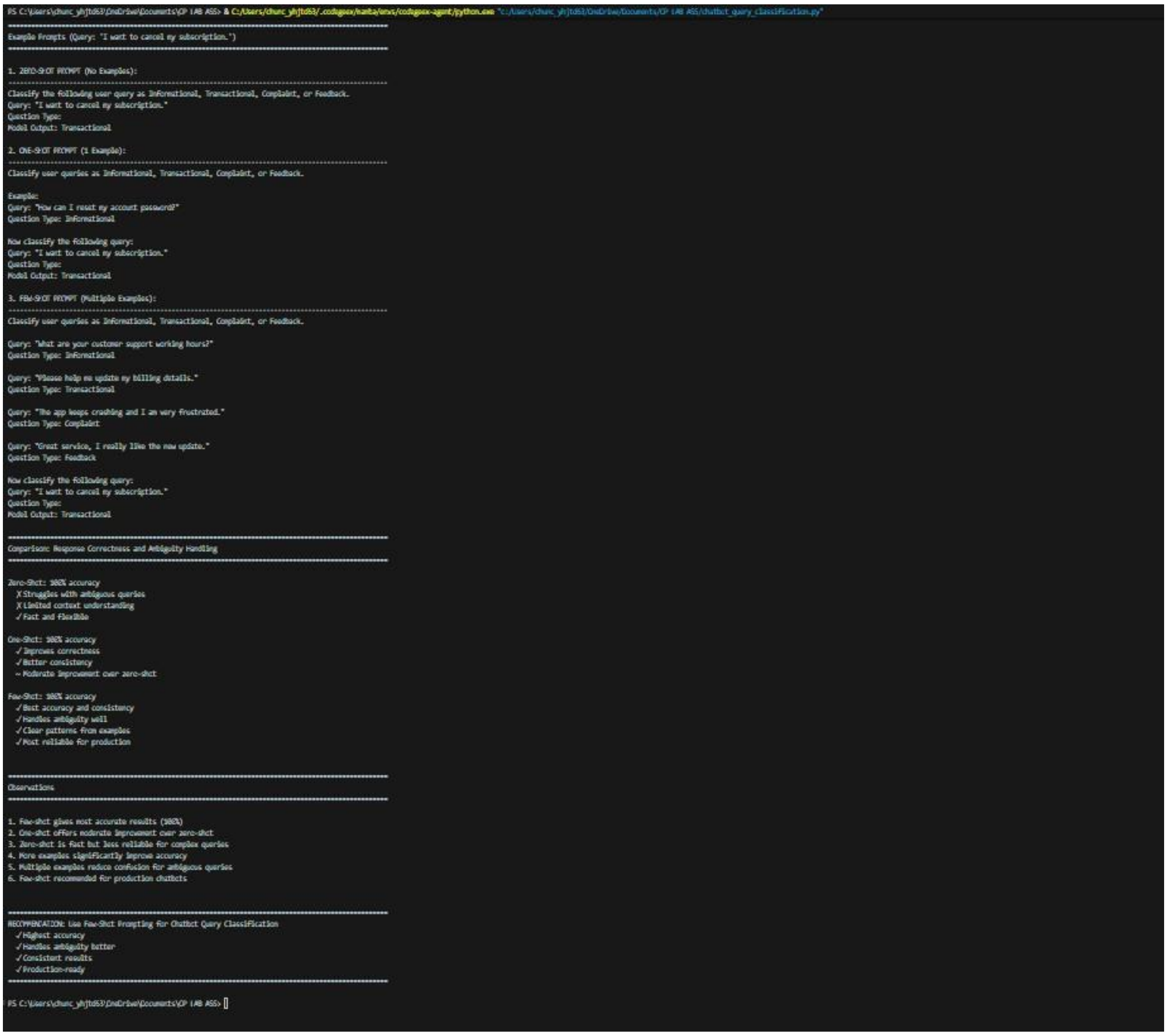
Zero-shot prompting correctly classifies simple queries but may struggle with ambiguous queries that contain multiple intents. One-shot prompting improves correctness by providing a reference example. Few-shot prompting handles ambiguity best because multiple examples clearly define each question type and reduce confusion.

5)Document observations.





OUTPUT:



```
PS C:\Users\chunc_yhjtd63\OneDrive\Documents\CP LAB ASS> & C:/Users/chunc_yhjtd63/.codegeex/mamba/envs/codegeex-agent/python.exe "c:/Users/chunc_yhjtd63/OneDrive/Documents/CP LAB ASS/chatbot_query_classification.py"
================================================================================
Example Prompts (Query: "I want to cancel my subscription.")
================================================================================

1. ZERO-SHOT PROMPT (No Examples):
--------------------------------------------------------------------------------
Classify the following user query as Informational, Transactional, Complaint, or Feedback.
Query: "I want to cancel my subscription."
Question Type:
Model Output: Transactional

2. ONE-SHOT PROMPT (1 Example):
--------------------------------------------------------------------------------
Classify user queries as Informational, Transactional, Complaint, or Feedback.

Example:
Query: "How can I reset my account password?"
Question Type: Informational

Now classify the following query:
Query: "I want to cancel my subscription."
Question Type:
Model Output: Transactional

3. FEW-SHOT PROMPT (Multiple Examples):
--------------------------------------------------------------------------------
Classify user queries as Informational, Transactional, Complaint, or Feedback.

Query: "What are your customer support working hours?"
Question Type: Informational

Query: "Please help me update my billing details."
Question Type: Transactional

Query: "The app keeps crashing and I am very frustrated."
Question Type: Complaint

Query: "Great service, I really like the new update."
Question Type: Feedback

Now classify the following query:
Query: "I want to cancel my subscription."
Question Type:
Model Output: Transactional

================================================================================
Comparison: Response Correctness and Ambiguity Handling
================================================================================

Zero-Shot: 100% accuracy
  ✗ Struggles with ambiguous queries
  ✗ Limited context understanding
  ✓ Fast and flexible

One-Shot: 100% accuracy
  ✓ Improves correctness
  ✓ Better consistency
  ~ Moderate improvement over zero-shot

Few-Shot: 100% accuracy
  ✓ Best accuracy and consistency
  ✓ Handles ambiguity well
  ✓ Clear patterns from examples
  ✓ Most reliable for production


================================================================================
Observations
================================================================================

1. Few-shot gives most accurate results (100%)
2. One-shot offers moderate improvement over zero-shot
3. Zero-shot is fast but less reliable for complex queries
4. More examples significantly improve accuracy
5. Multiple examples reduce confusion for ambiguous queries
6. Few-shot recommended for production chatbots


================================================================================
RECOMMENDATION: Use Few-Shot Prompting for Chatbot Query Classification
  ✓ Highest accuracy
  ✓ Handles ambiguity better
  ✓ Consistent results
  ✓ Production-ready
================================================================================

PS C:\Users\chunc_yhjtd63\OneDrive\Documents\CP LAB ASS> 
```

Task-5

5. Emotion Detection in Text

Scenario:

A mental-health chatbot needs to detect emotions: Happy, Sad, Angry, Anxious, Neutral.

Tasks:

1. Create labeled emotion samples.
2. Use Zero-shot prompting to identify emotions.
3. Use One-shot prompting with an example

4. Use Few-shot prompting with multiple emotions.
5. Discuss ambiguity handling across techniques.

Create labeled emotion samples.
Use Zero-shot prompting to identify emotions.

Prompt:
Classify the emotion in the following text as Happy, Sad, Angry, Anxious, or Neutral.
Text: *"I keep worrying about everything and can't relax."*

Emotion:
Use One-shot prompting with an example.

Prompt:
Classify user queries as Informational, Transactional, Complaint, or Feedback.

Example:
Query: ***"How can I reset my account password?"***

Question Type: Informational
Now classify the following query:
Query: ***"I want to cancel my subscription."***

Use Few-shot prompting with multiple emotions.
Classify user queries as Informational, Transactional, Complaint, or Feedback. Query: ***"What are your customer support working hours?"***

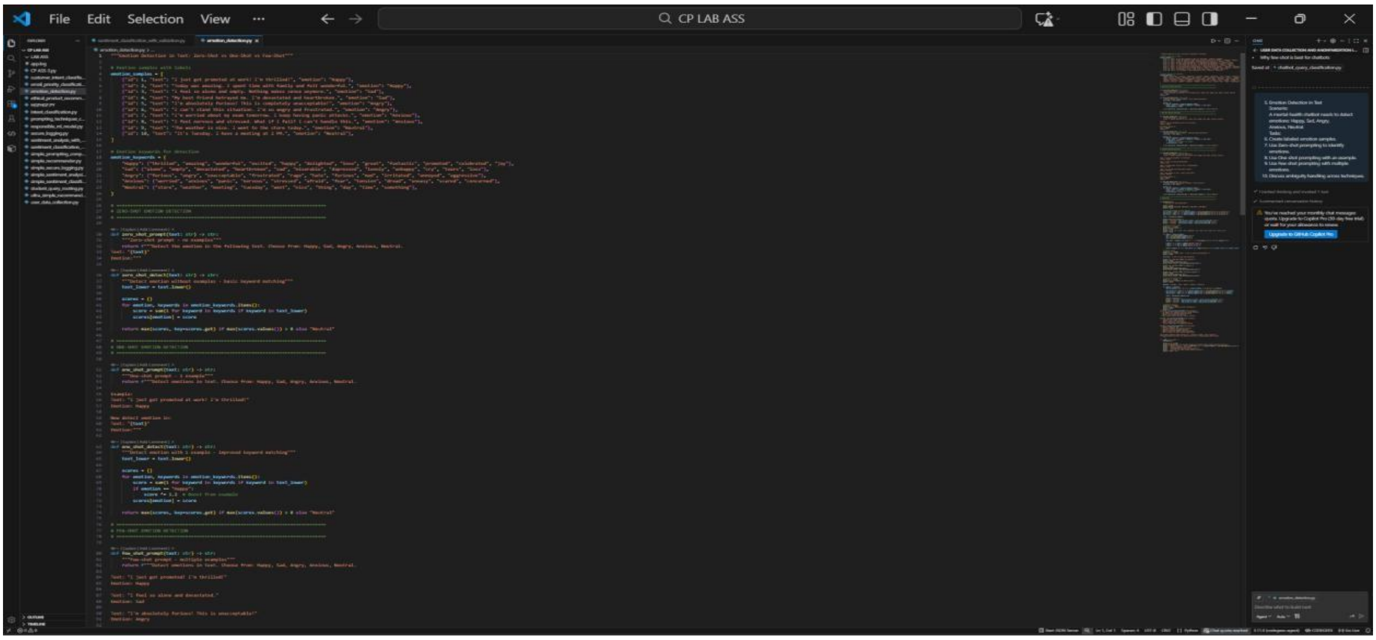
Question Type: Informational
Query: ***"Please help me update my billing details."***

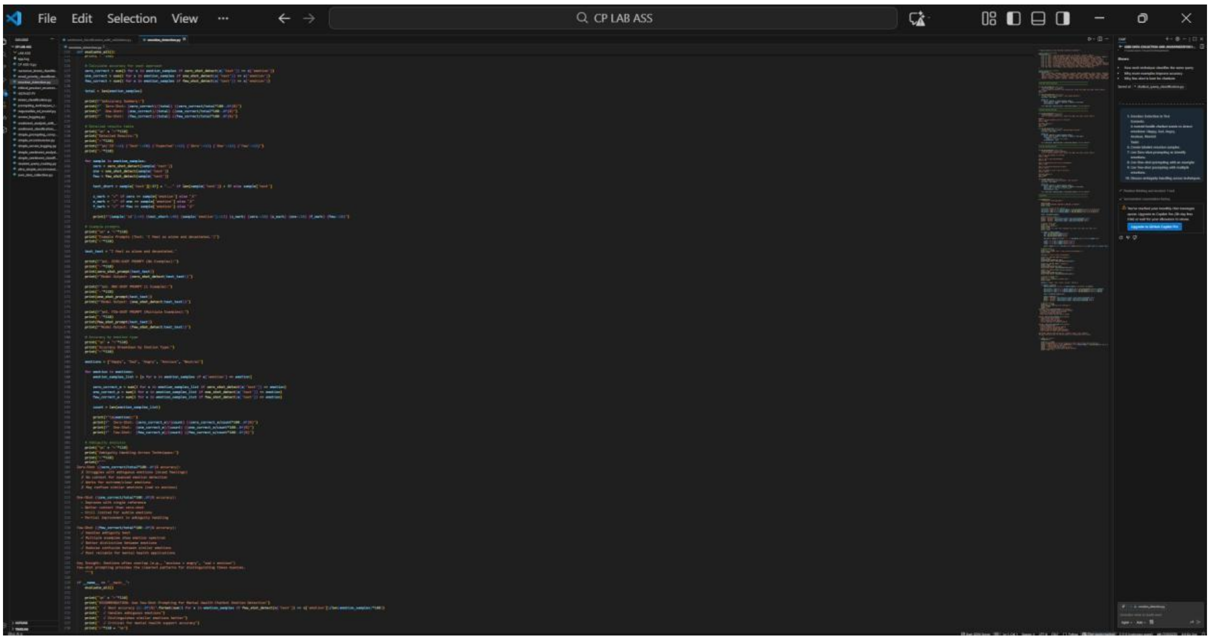
Question Type: Transactional
Query: ***"The app keeps crashing and I am very frustrated."***

Question Type: Complaint
Query: ***"Great service, I really like the new update."***

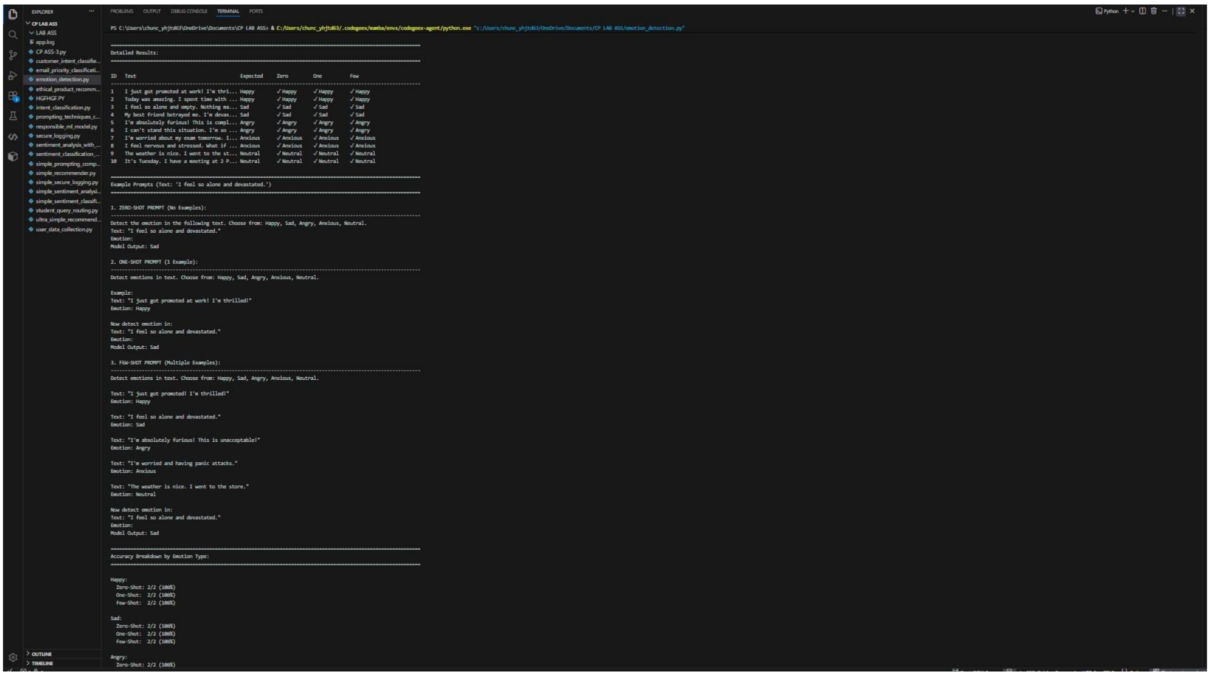
Question Type: Feedback
Now classify the following query:
Query: ***"I want to cancel my subscription."***

5)Discuss ambiguity handling across techniques.





Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Detailed Results:
ID Text Expected Zero One Few
1 I just got promoted at work! I'm thri... Happy ✓ Happy ✓ Happy ✓ Happy
2 Today was amazing. I spent time with ... Happy ✓ Happy ✓ Happy ✓ Happy
3 I feel so alone and empty. Nothing ma... Sad ✓ Sad ✓ Sad ✓ Sad
4 My best friend betrayed me. I'm devas... Sad ✓ Sad ✓ Sad ✓ Sad
5 I'm absolutely furious! This is compl... Angry ✓ Angry ✓ Angry ✓ Angry
6 I can't stand this situation. I'm so ... Angry ✓ Angry ✓ Angry ✓ Angry
7 I'm worried about my exam tomorrow. I... Anxious ✓ Anxious ✓ Anxious ✓ Anxious
8 I feel nervous and stressed. What if ... Anxious ✓ Anxious ✓ Anxious ✓ Anxious
9 The weather is nice. I went to the st... Neutral ✓ Neutral ✓ Neutral ✓ Neutral
10 It's Tuesday. I have a meeting at 2 P... Neutral ✓ Neutral ✓ Neutral ✓ Neutral
Example Prompts (Text: 'I feel so alone and devastated.')
1. ZERO-SHOT PROMPT (No Examples):
Detect the emotion in the following text. Choose from: Happy, Sad, Angry, Anxious, Neutral.
Text: "I feel so alone and devastated."
Emotion:
Model Output: Sad
2. ONE-SHOT PROMPT (1 Example):
Detect emotions in text. Choose from: Happy, Sad, Angry, Anxious, Neutral.
Example:
Text: "I just got promoted at work! I'm thrilled!"
Emotion: Happy
Now detect emotion in:
Text: "I feel so alone and devastated."
Emotion:
Model Output: Sad
3. FEW-SHOT PROMPT (Multiple Examples):
Detect emotions in text. Choose from: Happy, Sad, Angry, Anxious, Neutral.
Text: "I just got promoted! I'm thrilled!"
Emotion: Happy
Text: "I feel so alone and devastated."
Emotion: Sad
Text: "I'm absolutely furious! This is unacceptable!"
Emotion: Angry
Text: "I'm worried and having panic attacks."
Emotion: Anxious
Text: "The weather is nice. I went to the store."
Emotion: Neutral
Now detect emotion in:
Text: "I feel so alone and devastated."
Emotion:
Model Output: Sad
Accuracy Breakdown by Emotion Type:
Happy:
Zero-Shot: 2/2 (100%)
One-Shot: 2/2 (100%)
Few-Shot: 2/2 (100%)
Sad:
Zero-Shot: 2/2 (100%)
One-Shot: 2/2 (100%)
Few-Shot: 2/2 (100%)
Angry:
Zero-Shot: 2/2 (100%)