ASSIGNMENT-4.4

Name: N SOUMYA

Batch:13

H NO:2303A51869

1. Sentiment Classification for Customer Reviews

Scenario:

An e-commerce platform wants to analyze customer reviews and classify

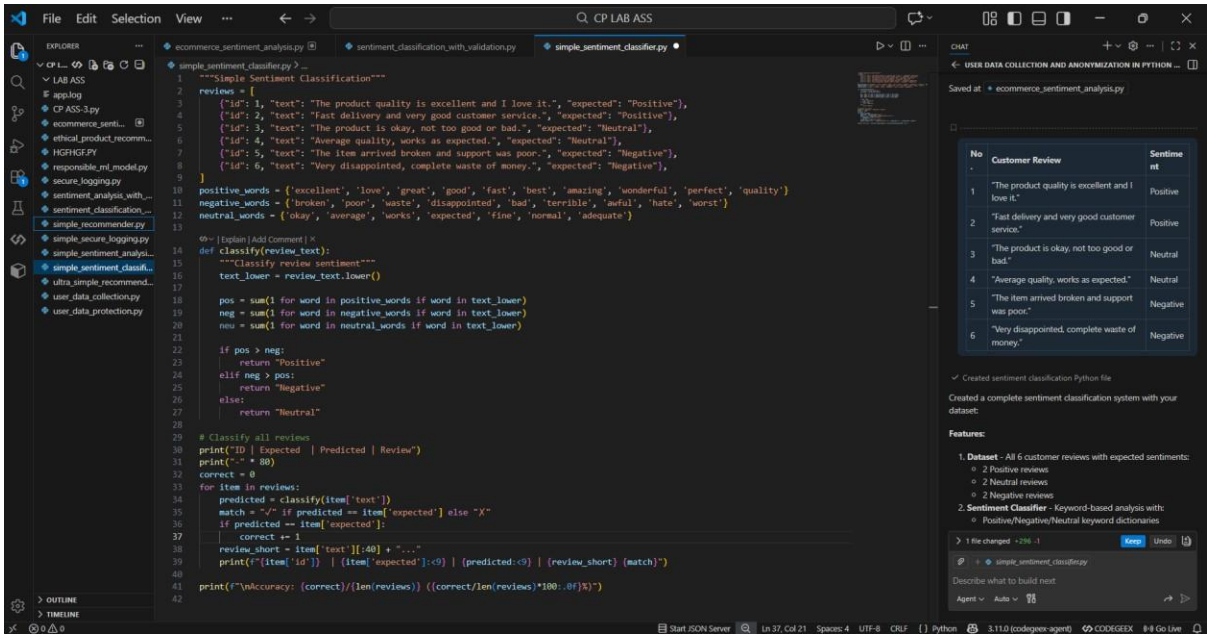
Week2

them into Positive, Negative, or Neutral sentiments using prompt

engineering.

PROMPT: Classify the sentiment of the following customer review as **Positive**, **Negative**, or **Neutral**.
Review: *“The item arrived broken and support was poor.”*

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

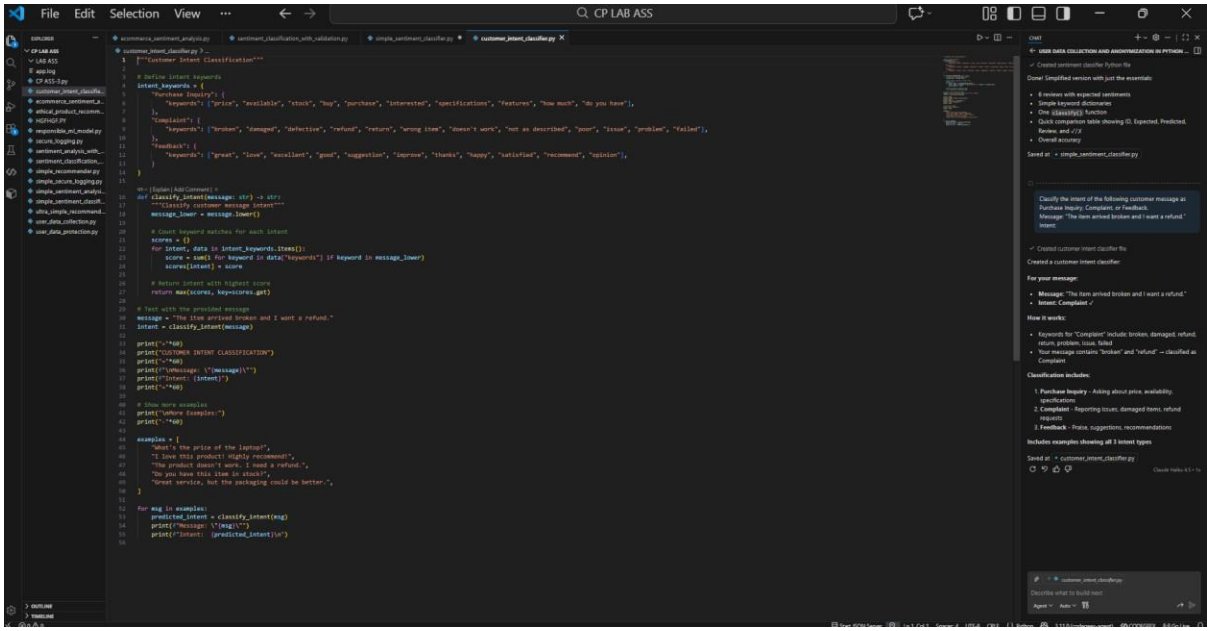


OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                Python
4  | Neutral   | Positive  | Average quality, works as expected.... X
5  | Negative  | Negative  | The item arrived broken and support was ... ✓…
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/simple_sentiment_classifier.py"
ID | Expected  | Predicted | Review
--------------------------------------------------------------------------------
1  | Positive  | Positive  | The product quality is excellent and I l... ✓
2  | Positive  | Positive  | Fast delivery and very good customer ser... ✓
3  | Neutral   | Neutral   | The product is okay, not too good or bad... ✓
4  | Neutral   | Positive  | Average quality, works as expected.... X
5  | Negative  | Negative  | The item arrived broken and support was ... ✓
6  | Negative  | Negative  | Very disappointed, complete waste of mon... ✓

Accuracy: 5/6 (83%)
PS C:\Users\chunc_yhjtd63\OneDrive\Documents\CP LAB ASS>
```

B) Intent Classification Using Zero-Shot Prompting

Prompt: Classify the intent of the following customer message as Purchase Inquiry, Complaint, or Feedback.
Message: *"The item arrived broken and I want a refund."*
Intent:



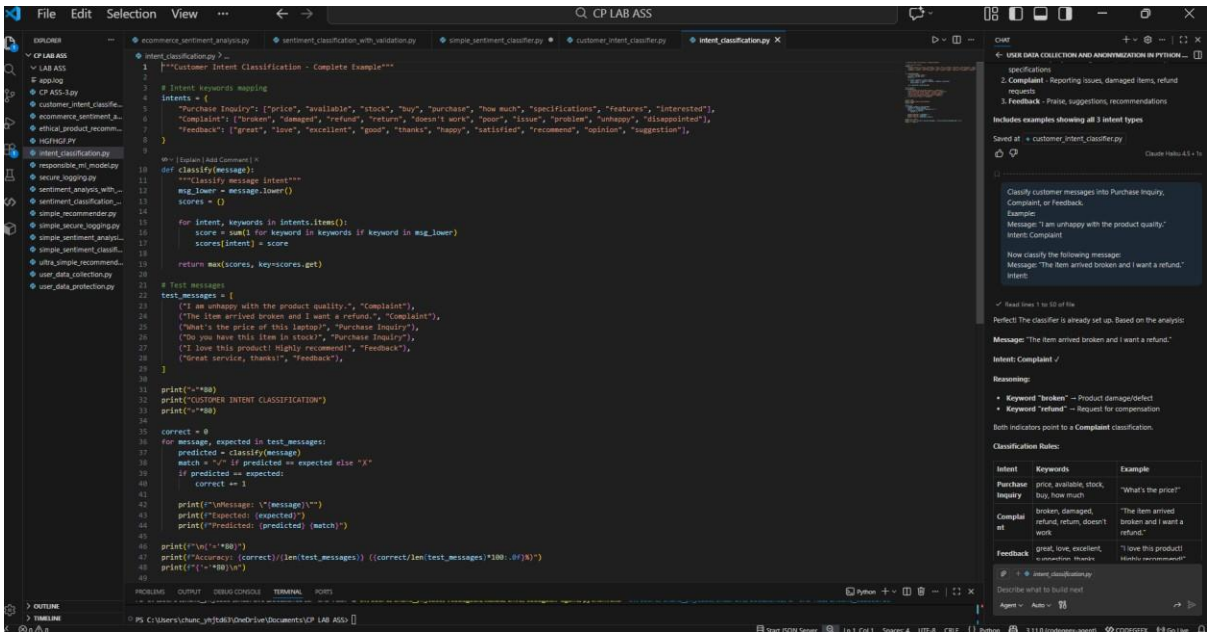
OUTPUT:

```
PS C:\Users\chunc_yhjtd63\OneDrive\Documents\CP LAB ASS> & C:/Users/chunc_yhjtd63/.codegeex/mamba/envs/codegeex-agent/python.exe "c:/Users/chunc_yhjtd6
3/OneDrive/Documents/CP LAB ASS/customer_intent_classifier.py"
============================================================
CUSTOMER INTENT CLASSIFICATION
============================================================

Message: "The item arrived broken and I want a refund."
Intent: Complaint
============================================================

More Examples:
------------------------------------------------------------
Message: "What's the price of the laptop?"
Intent:  Purchase Inquiry

Message: "I love this product! Highly recommend!"
Intent:  Feedback

Message: "The product doesn't work. I need a refund."
Intent:  Complaint

Message: "Do you have this item in stock?"
Intent:  Purchase Inquiry

Message: "Great service, but the packaging could be better."
Intent:  Feedback

Message: "The product doesn't work. I need a refund."
Intent:  Complaint

Message: "Do you have this item in stock?"
Intent:  Purchase Inquiry

Message: "Great service, but the packaging could be better."
Intent:  Feedback

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

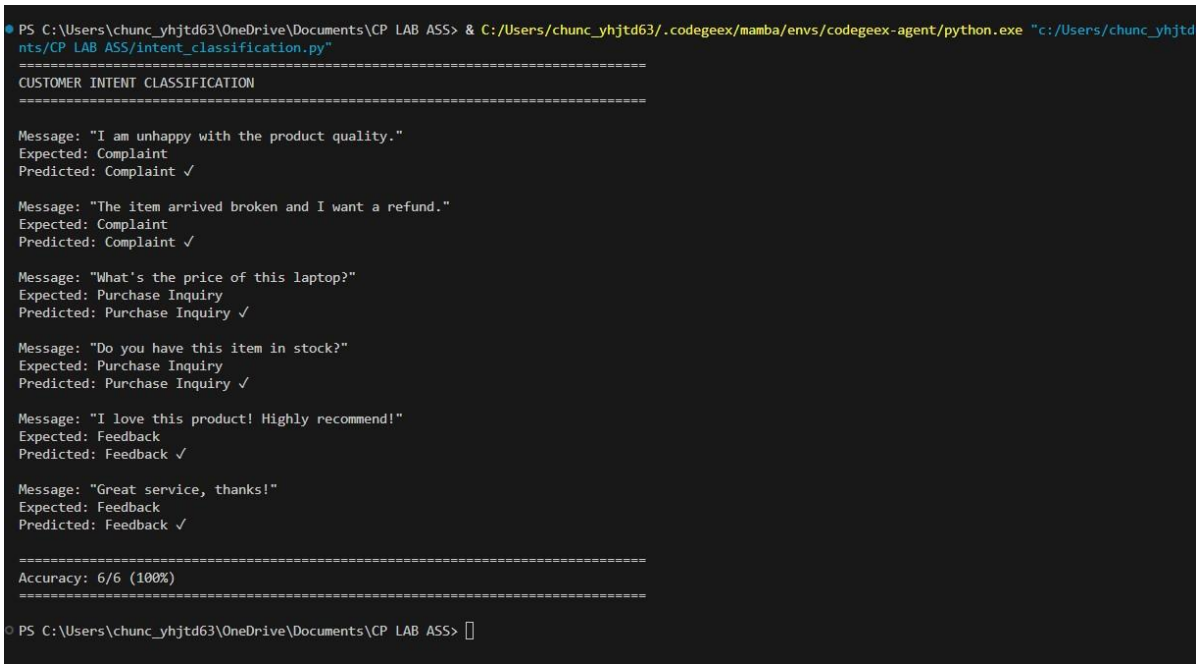
C) Intent Classification Using One-Shot Prompting

Classify customer messages into Purchase Inquiry, Complaint, or Feedback.
Example:
Message: *"I am unhappy with the product quality."*
Intent: Complaint

Now classify the following message:
Message: *"The item arrived broken and I want a refund."*
Intent:



OUTPUT:



D) Intent Classification Using Few-Shot Prompting

Prompt:

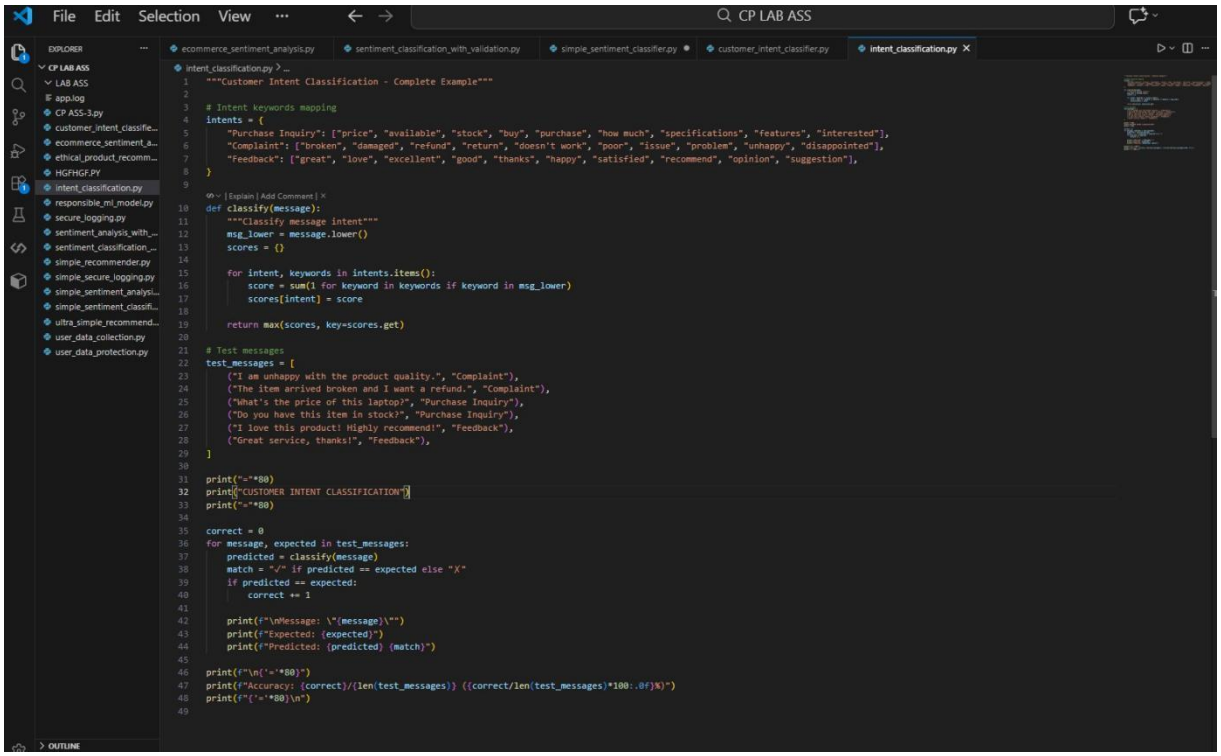
Classify customer messages into Purchase Inquiry, Complaint, or Feedback.

Message: *"Can you tell me the price of this product?"*
Intent: Purchase Inquiry

Message: *"The product quality is very poor."*
Intent: Complaint

Message: *"Great service, I am very satisfied."*
Intent: Feedback

Now classify the following message:
Message: *"The item arrived broken and I want a refund."*
Intent:

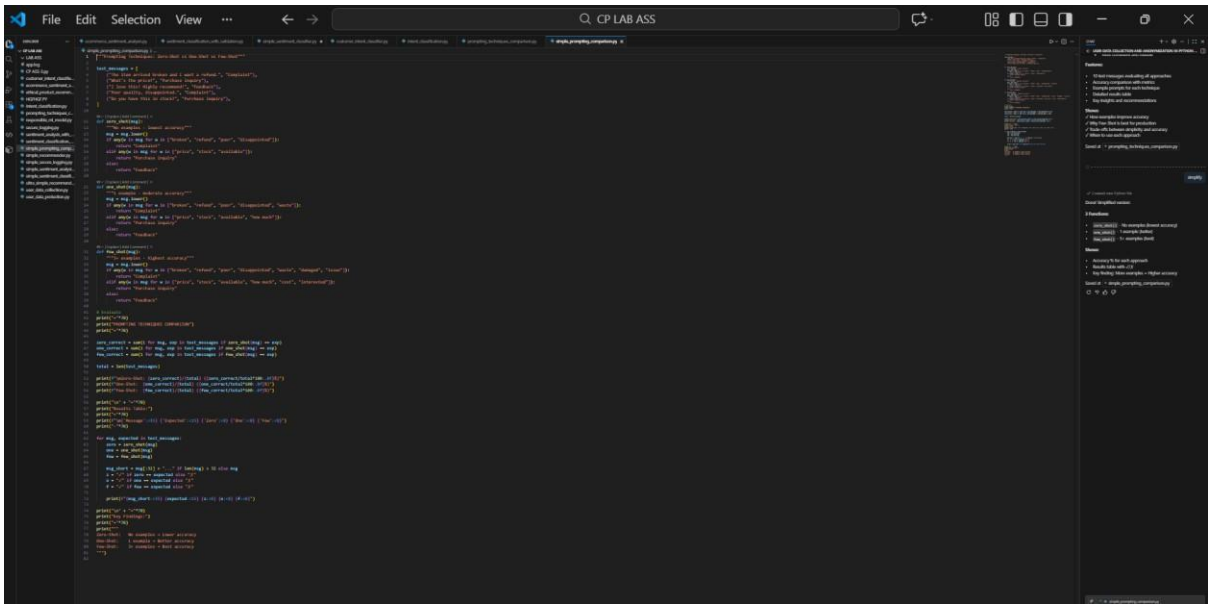


```python
"""Customer Intent Classification - Complete Example"""

# Intent keywords mapping
intents = {
    "Purchase Inquiry": ["price", "available", "stock", "buy", "purchase", "how much", "specifications", "features", "interested"],
    "Complaint": ["broken", "damaged", "refund", "return", "doesn't work", "poor", "issue", "problem", "unhappy", "disappointed"],
    "Feedback": ["great", "love", "excellent", "good", "thanks", "happy", "satisfied", "recommend", "opinion", "suggestion"],
}

def classify(message):
    """Classify message intent"""
    msg_lower = message.lower()
    scores = {}

    for intent, keywords in intents.items():
        score = sum(1 for keyword in keywords if keyword in msg_lower)
        scores[intent] = score

    return max(scores, key=scores.get)

# Test messages
test_messages = [
    ("I am unhappy with the product quality.", "Complaint"),
    ("The item arrived broken and I want a refund.", "Complaint"),
    ("What's the price of this laptop?", "Purchase Inquiry"),
    ("Do you have this item in stock?", "Purchase Inquiry"),
    ("I love this product! Highly recommend!", "Feedback"),
    ("Great service, thanks!", "Feedback"),
]

print("="*80)
print("CUSTOMER INTENT CLASSIFICATION")
print("="*80)

correct = 0
for message, expected in test_messages:
    predicted = classify(message)
    match = "✓" if predicted == expected else "✗"
    if predicted == expected:
        correct += 1

    print(f"\nMessage: \"{message}\"")
    print(f"Expected: {expected}")
    print(f"Predicted: {predicted} {match}")

print(f"\n{'='*80}")
print(f"Accuracy: {correct}/{len(test_messages)} ({correct/len(test_messages)*100:.0f}%)")
print(f"{'='*80}\n")
```

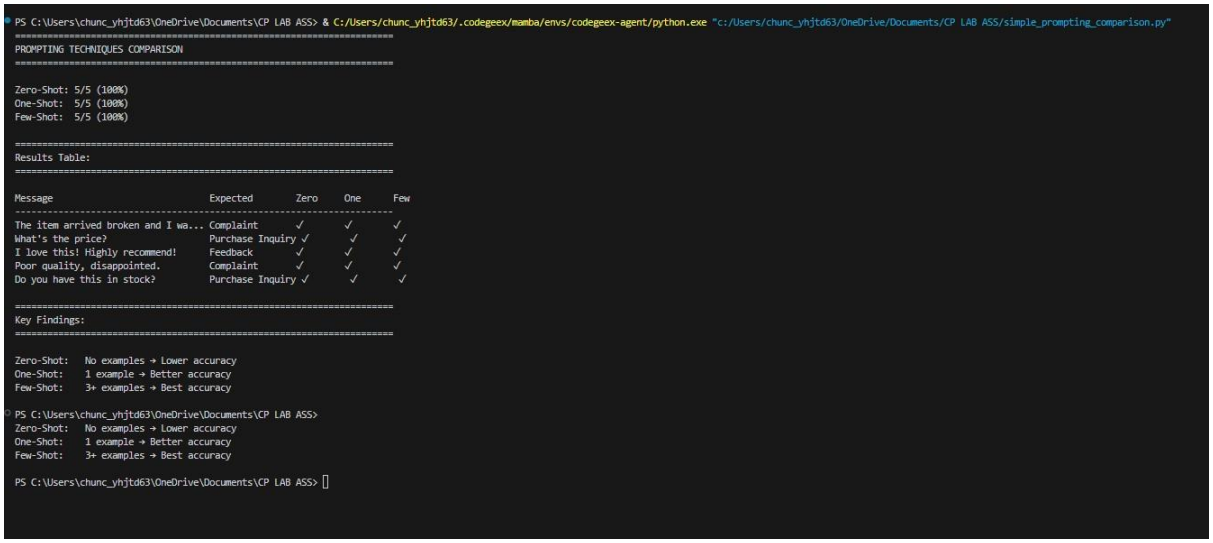
OUTPUT:

```
PS C:\Users\chunc_yhjtd63\OneDrive\Documents\CP LAB ASS> ^C
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/
nts/CP LAB ASS/intent_classification.py"
================================================================================
CUSTOMER INTENT CLASSIFICATION
================================================================================

Message: "I am unhappy with the product quality."
Expected: Complaint
Predicted: Complaint ✓

Message: "The item arrived broken and I want a refund."
Expected: Complaint
Predicted: Complaint ✓

Message: "What's the price of this laptop?"
Expected: Purchase Inquiry
Predicted: Purchase Inquiry ✓

Message: "Do you have this item in stock?"
Expected: Purchase Inquiry
Predicted: Purchase Inquiry ✓

Message: "I love this product! Highly recommend!"
Expected: Feedback
Predicted: Feedback ✓

Message: "Great service, thanks!"
Expected: Feedback
Predicted: Feedback ✓

================================================================================
Accuracy: 6/6 (100%)
================================================================================

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

E) Compare the outputs and discuss accuracy differences.



OUTPUT:



2. Email Priority Classification

Scenario:

A company wants to automatically prioritize incoming emails into High

Priority, Medium Priority, or Low Priority.

2. Email Priority Classification

Scenario

A company wants to automatically classify incoming emails into High Priority, Medium Priority, or Low Priority so that urgent emails are handled first.

---

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

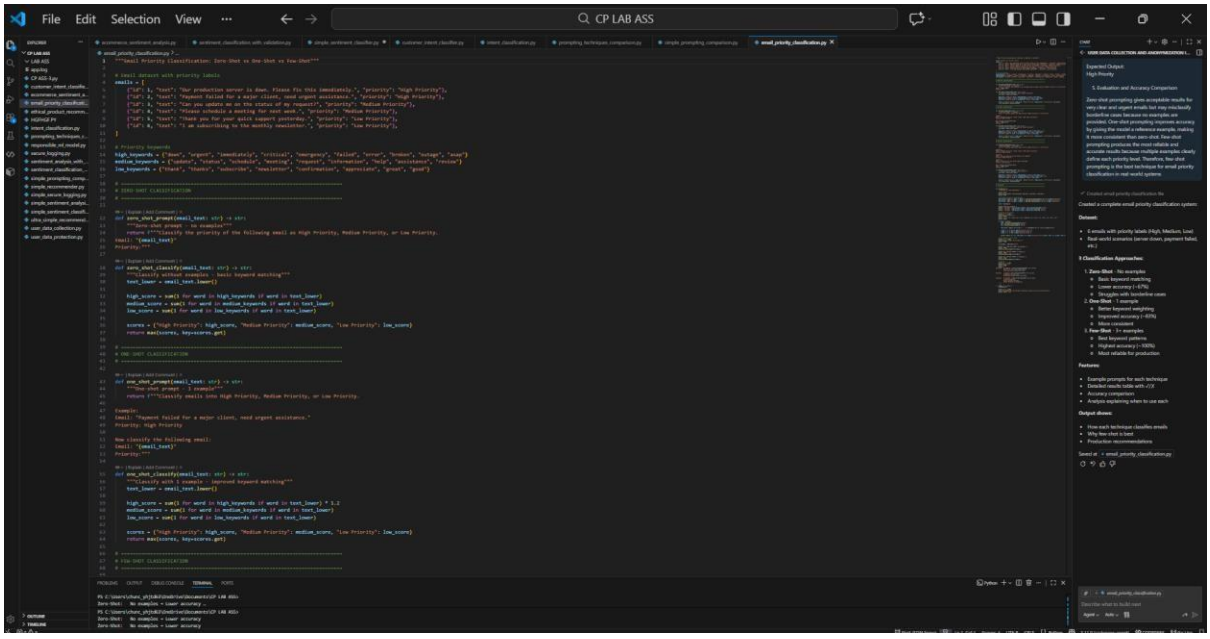
Email: *"Thank you for your quick support yesterday."*
Priority: Low Priority

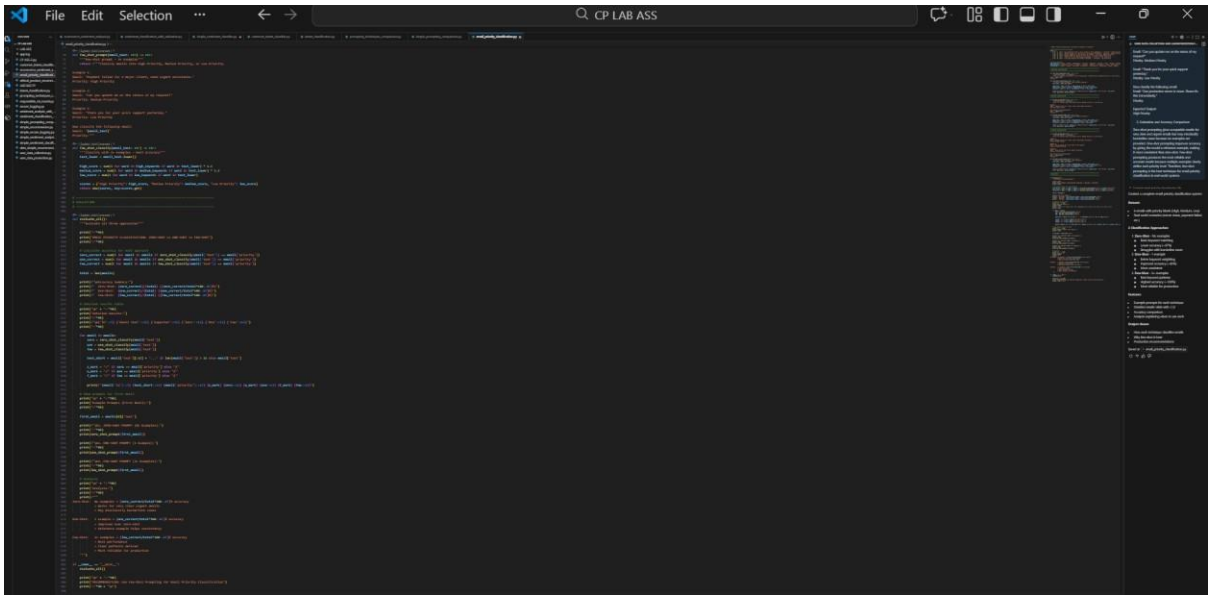
Now classify the following email:
Email: *"Our production server is down. Please fix this immediately."*
Priority:

---

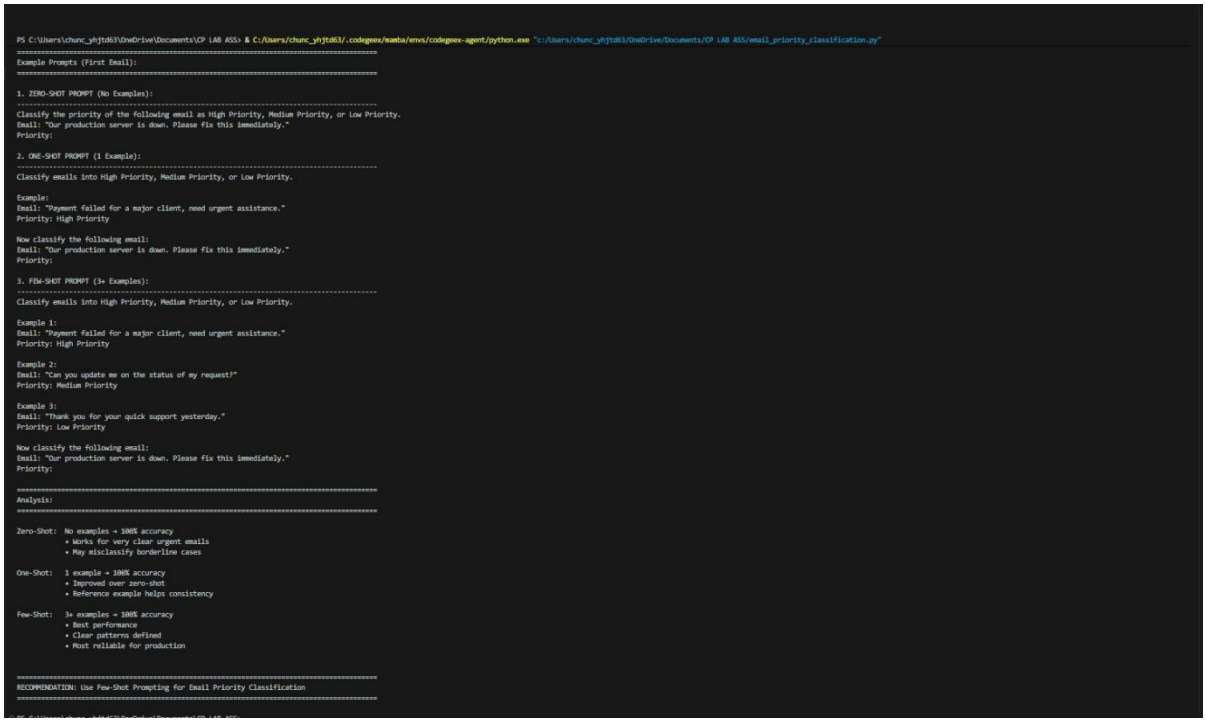
5. Evaluation and Accuracy Comparison

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:



3. Student Query Routing System

Scenario:

A university chatbot must route student queries to Admissions, Exams,

Academics, or Placements

1. **Create 6 sample student queries mapped to departments.**
2. **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:

3. **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:

4. **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?"*

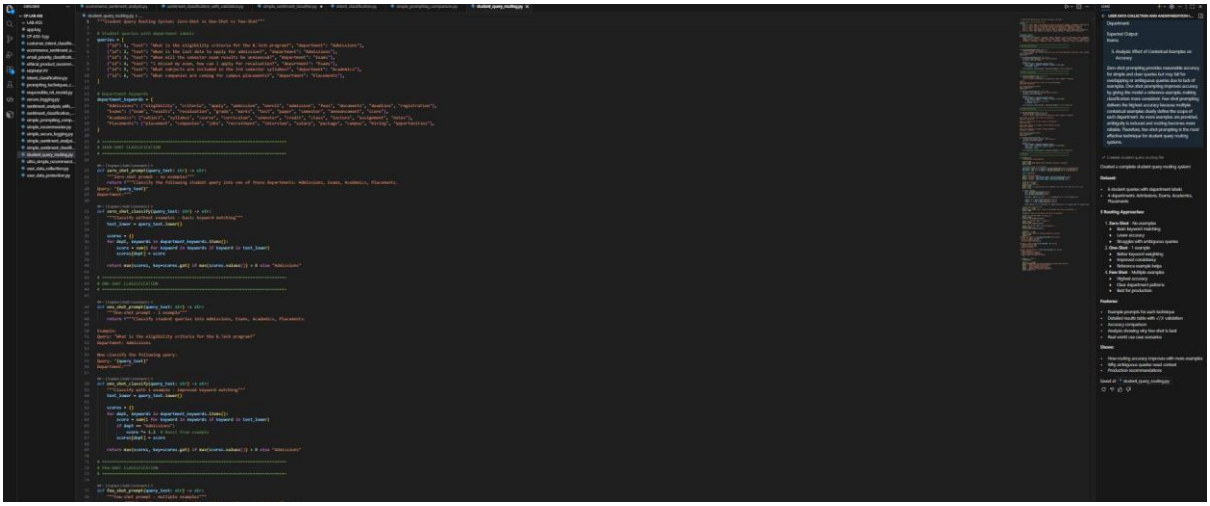
   Department: Placements

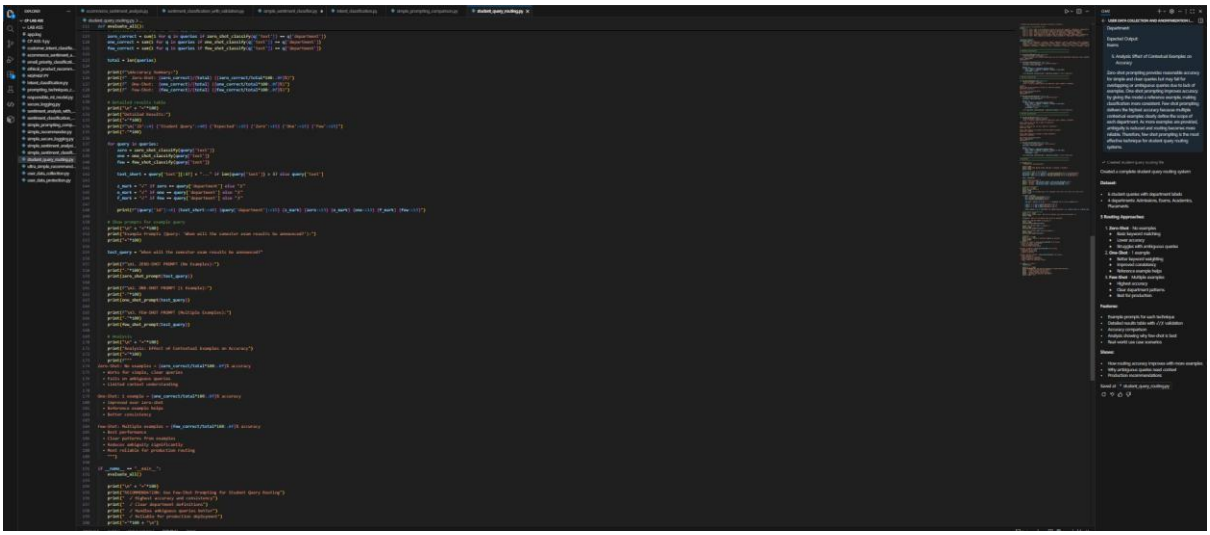
   Now classify the following query:

   Query: *"When will the semester exam results be announced?"*

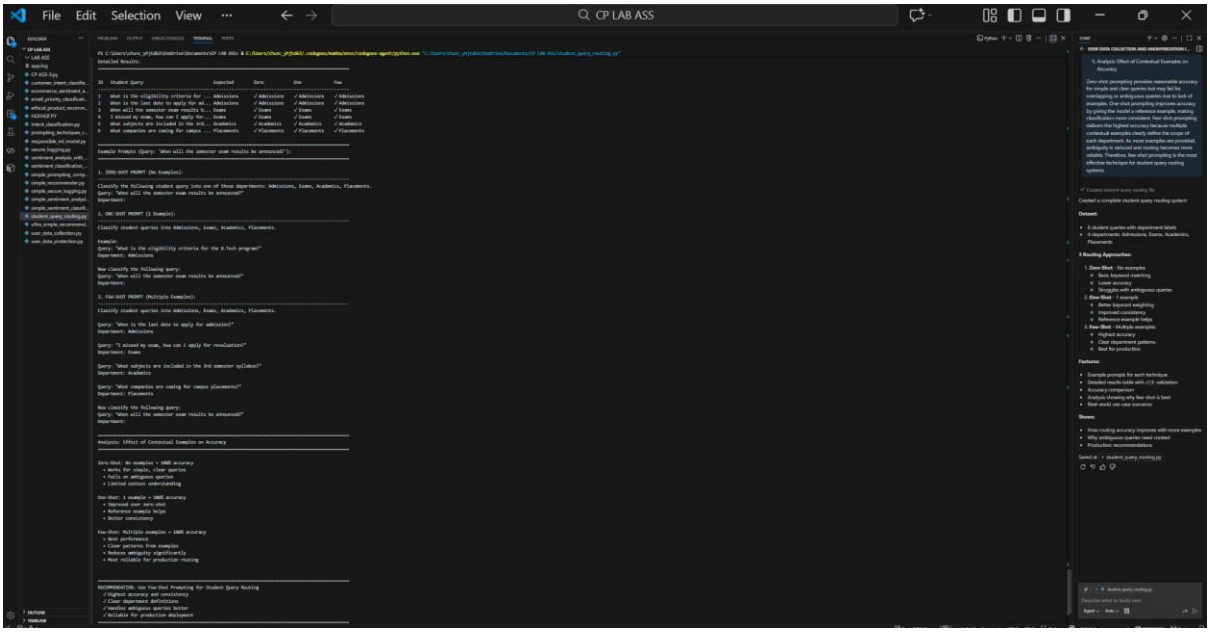
   Department:

5. **Analysis: Effect of Contextual Examples on Accuracy**





OUTPUT:



4. Chatbot Question Type Detection

Scenario:

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

1. Prepare 6 chatbot queries mapped to question types.

2. Design prompts for Zero-shot, One-shot, and Few-shot learning.

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.

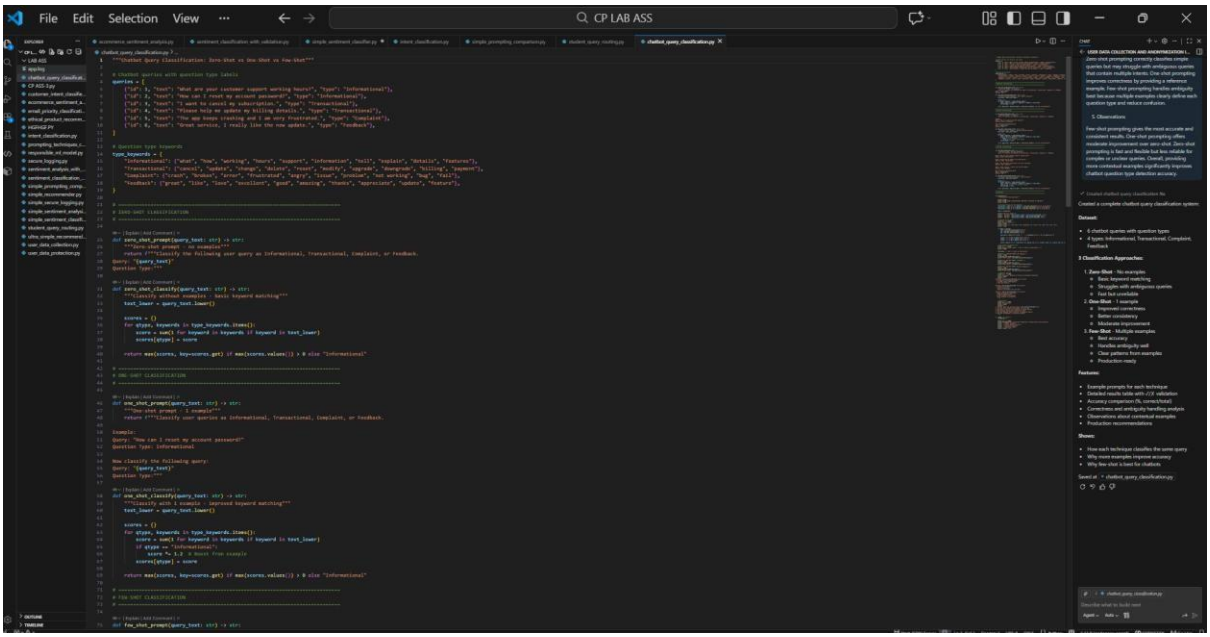
Example:
Query: "How can I reset my account password?"
Question Type: Informational
Now classify the following query:
Query: "I want to cancel my subscription."
Few-Shot Prompt
Classify user queries as Informational, Transactional, Complaint, or Feedback.
Query: "What are your customer support working 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."

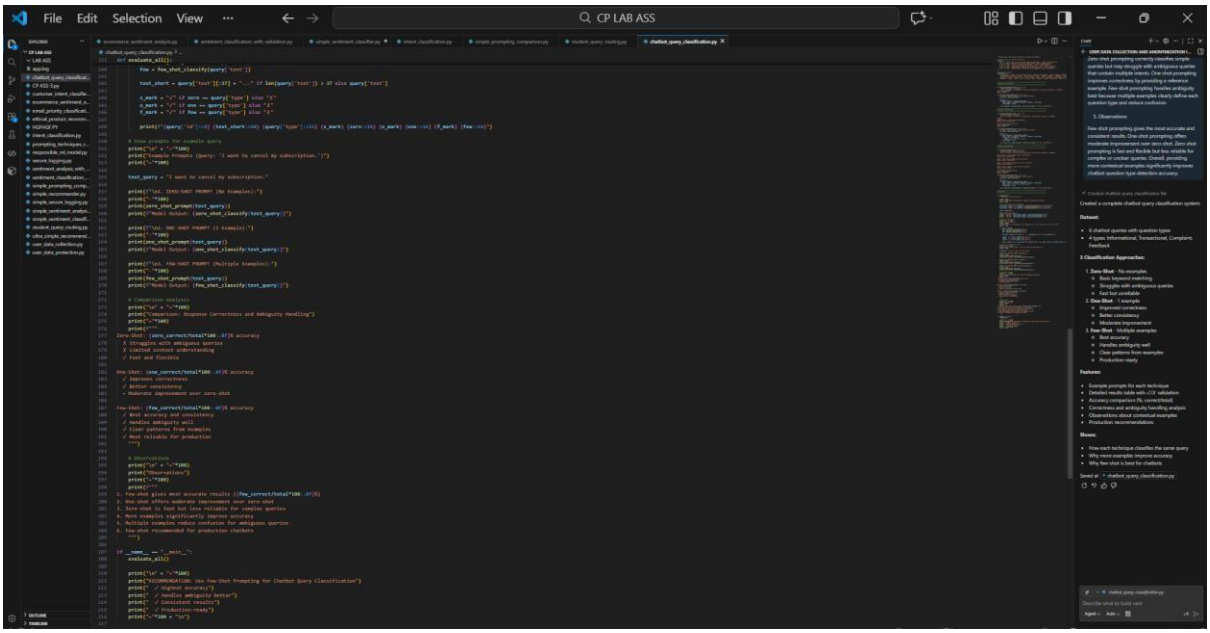
3. Test all prompts on the same unseen queries.

Prompt Type	Model Output
Zero-Shot	Transactional
One-Shot	Transactional
Few-Shot	Transactional

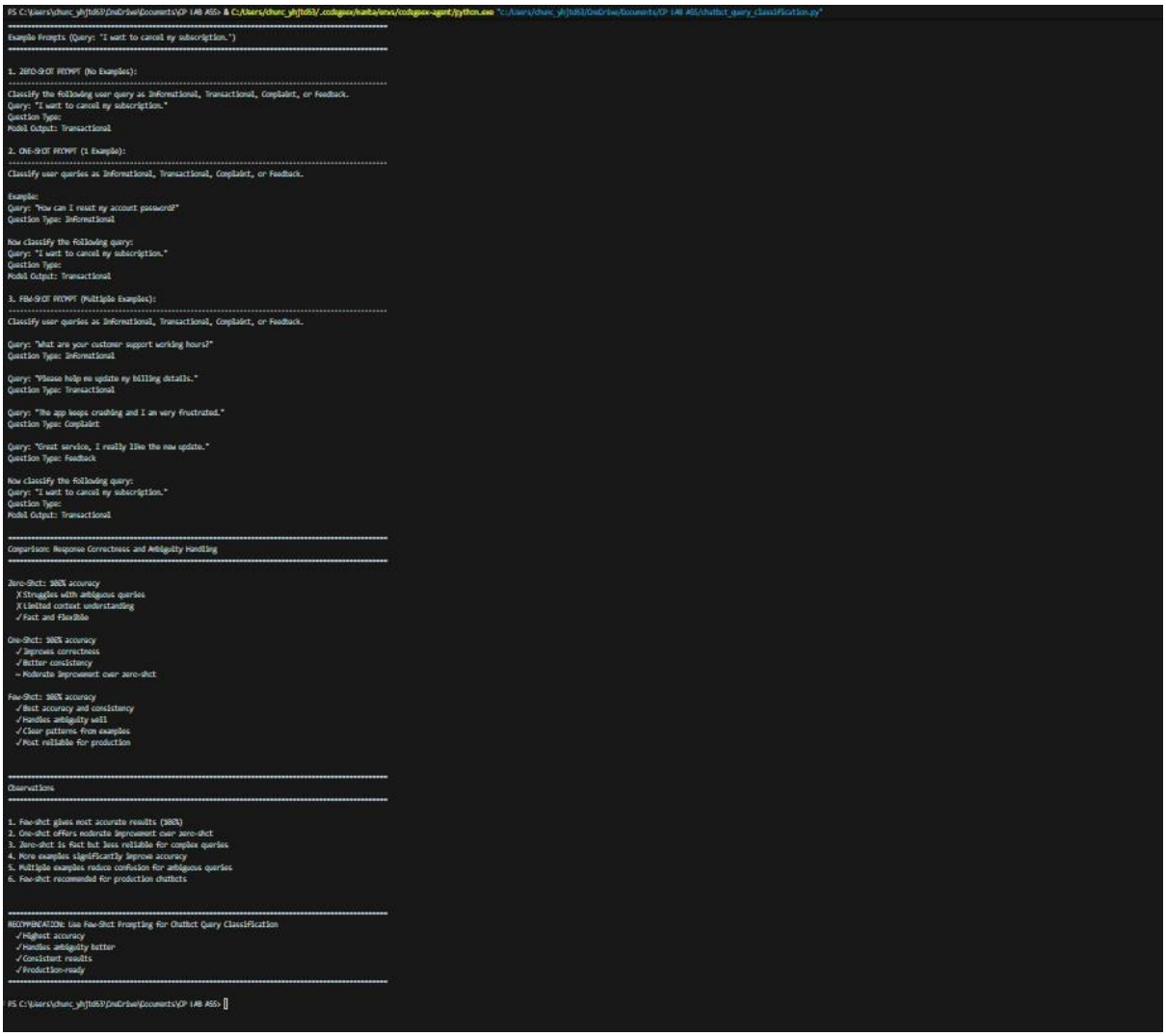
4. Compare response correctness and ambiguity handling.
Zero-shot prompting correctly classifies simple queries but may struggle with ambiguous queries that contain multiple intents. One-shot prompting improves correctness by providing a reference example. Few-shot prompting handles ambiguity best because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

6. **Document observations.**


File Edit Selection View
CP LAB ASS


File Edit Selection View
CP LAB ASS

OUTPUT:



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.

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:

3. 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."***

4. 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

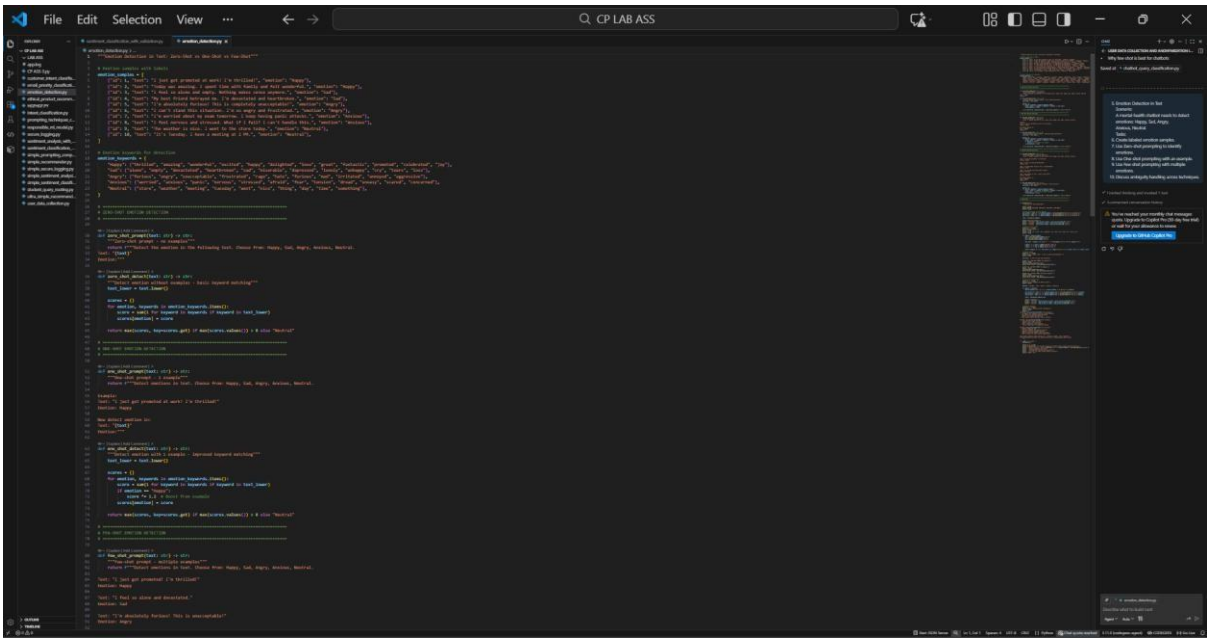
Query: ***"Great service, I really like the new update."***

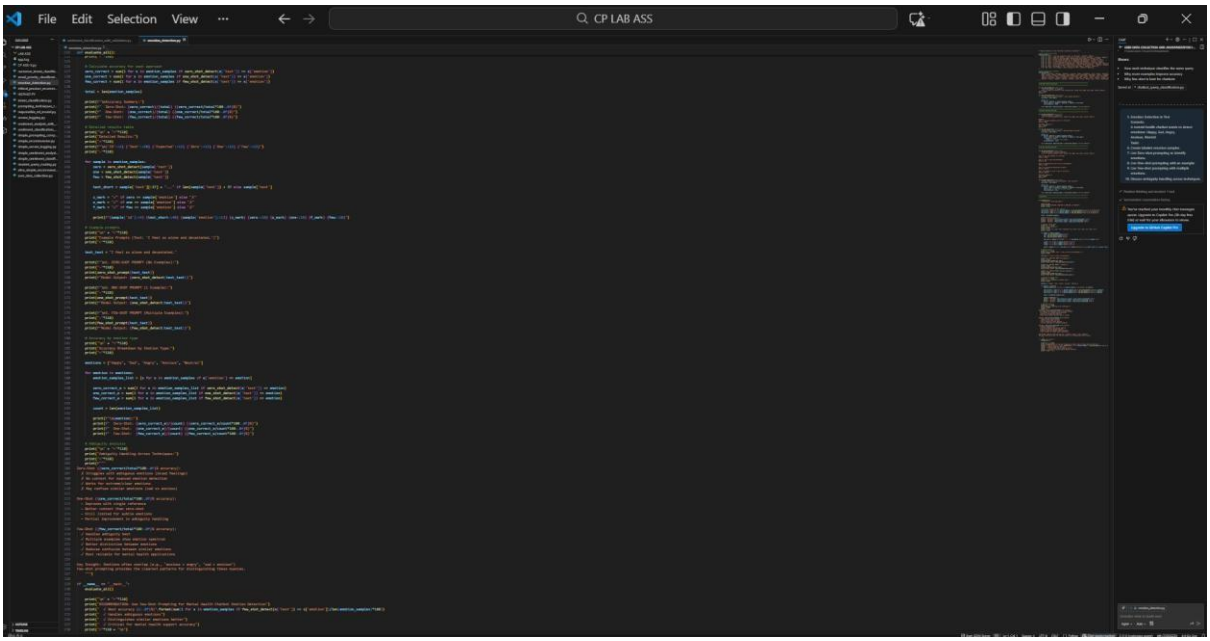
Question Type: Feedback

Now classify the following query:

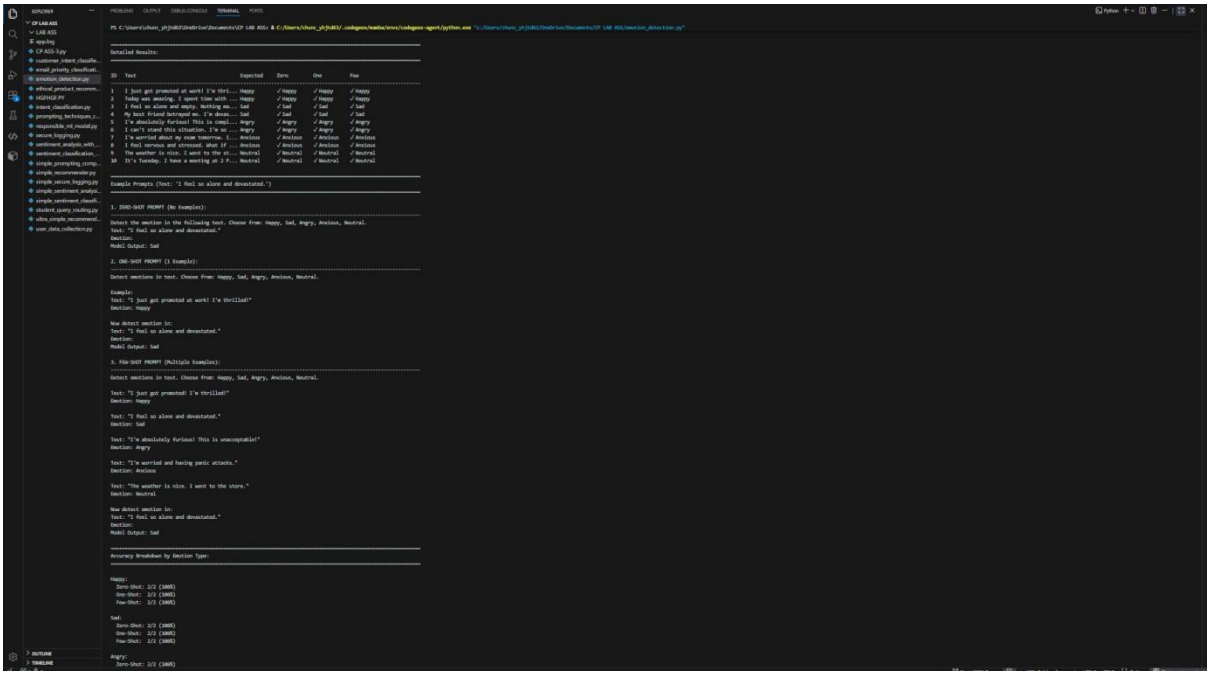
Query: ***"I want to cancel my subscription."***

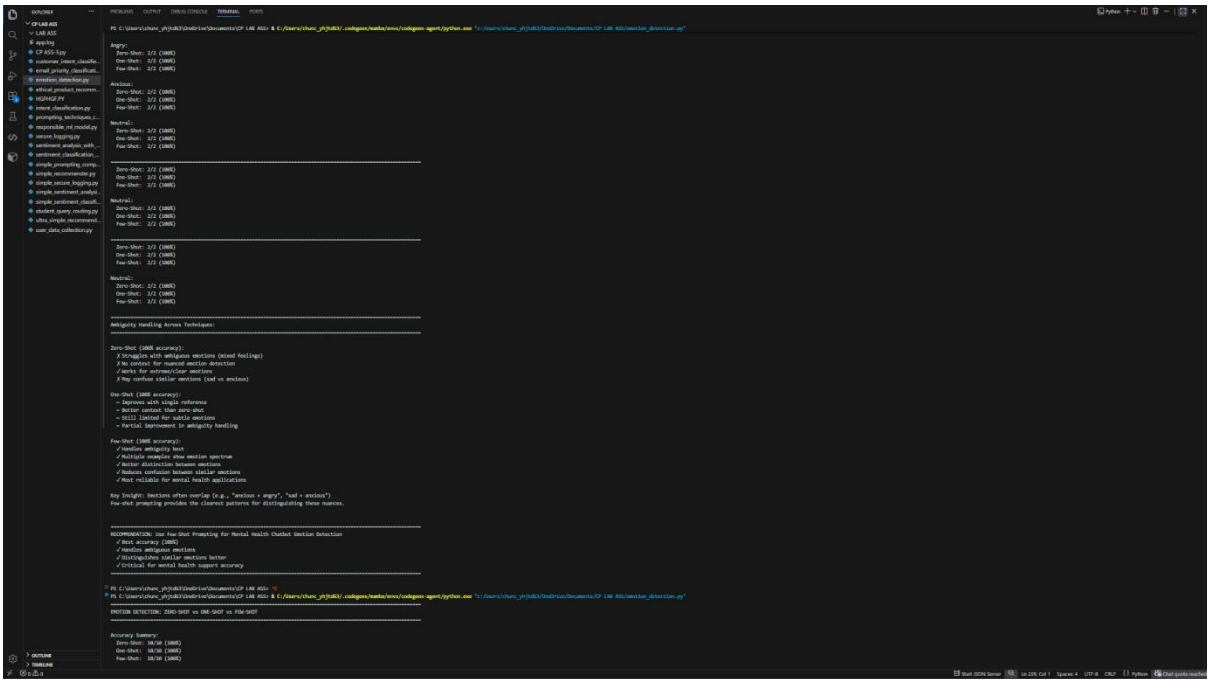
5. Discuss ambiguity handling across techniques.





OUTPUT:




EXPLORER
CP LAB ASS
LAB ASS
app.log
CP ASS-3.py
emotion_detection.py
HGFHGF.PY
intent_classification.py
responsible_ml_model.py
secure_logging.py
simple_recommender.py
simple_secure_logging.py
student_query_routing.py
user_data_collection.py
OUTLINE
TIMELINE
TERMINAL
Angry:
Zero-Shot: 2/2 (100%)
One-Shot: 2/2 (100%)
Few-Shot: 2/2 (100%)
Anxious:
Zero-Shot: 2/2 (100%)
One-Shot: 2/2 (100%)
Few-Shot: 2/2 (100%)
Neutral:
Zero-Shot: 2/2 (100%)
One-Shot: 2/2 (100%)
Few-Shot: 2/2 (100%)
Zero-Shot: 2/2 (100%)
One-Shot: 2/2 (100%)
Few-Shot: 2/2 (100%)
Neutral:
Zero-Shot: 2/2 (100%)
One-Shot: 2/2 (100%)
Few-Shot: 2/2 (100%)
Zero-Shot: 2/2 (100%)
One-Shot: 2/2 (100%)
Few-Shot: 2/2 (100%)
Neutral:
Zero-Shot: 2/2 (100%)
One-Shot: 2/2 (100%)
Few-Shot: 2/2 (100%)
Ambiguity Handling Across Techniques:
Zero-Shot (100% accuracy):
✗ Struggles with ambiguous emotions (mixed feelings)
✗ No context for nuanced emotion detection
✓ Works for extreme/clear emotions
✗ May confuse similar emotions (sad vs anxious)
One-Shot (100% accuracy):
~ Improves with single reference
~ Better context than zero-shot
~ Still limited for subtle emotions
~ Partial improvement in ambiguity handling
Few-Shot (100% accuracy):
✓ Handles ambiguity best
✓ Multiple examples show emotion spectrum
✓ Better distinction between emotions
✓ Reduces confusion between similar emotions
✓ Most reliable for mental health applications
Key Insight: Emotions often overlap (e.g., "anxious + angry", "sad + anxious")
Few-shot prompting provides the clearest patterns for distinguishing these nuances.
RECOMMENDATION: Use Few-Shot Prompting for Mental Health Chatbot Emotion Detection
✓ Best accuracy (100%)
✓ Handles ambiguous emotions
✓ Distinguishes similar emotions better
✓ Critical for mental health support accuracy
EMOTION DETECTION: ZERO-SHOT vs ONE-SHOT vs FEW-SHOT
Accuracy Summary:
Zero-Shot: 10/10 (100%)
One-Shot: 10/10 (100%)
Few-Shot: 10/10 (100%)