ASSIGNMENT – 4.5

NAME:AKSHITHA

HT NO:2303A51360

BATCH NO:29

ADVANCED PROMPT ENGINEERING: ZERO-SHOT, ONE-SHOT & FEW-SHOT

TASK-1:

ZERO-SHOT

A. Preparing Sample data:

```
 test_emails = [
 "My payment failed but money was deducted.",
 "The app is not opening on my phone.",
 "Great customer service, very satisfied.",
 "What is your customer care number?",
 "Invoice amount seems incorrect."
]
```

Expected Labels (for evaluation):

```
true_labels = [
 "Billing",
 "Technical Support",                                         "Feedback",
 "Others",
 "Billing"
]
```

PROMPT:

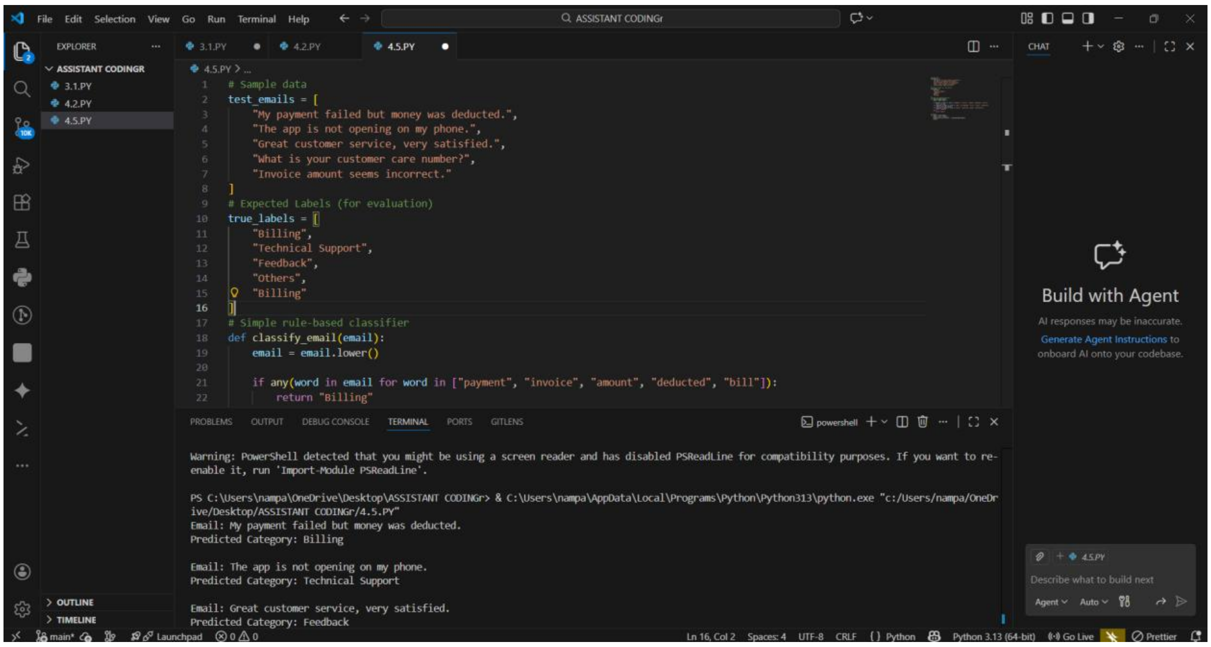
Classify the following email into one of the categories:

Billing, Technical Support, Feedback, Others.

Email: "<email_text>"

Return only the category name.

CODE:



```python
# Sample data
test_emails = [
    "My payment failed but money was deducted.",
    "The app is not opening on my phone.",
    "Great customer service, very satisfied.",
    "What is your customer care number?",
    "Invoice amount seems incorrect."
]
# Expected Labels (for evaluation)
true_labels = [
    "Billing",
    "Technical Support",
    "Feedback",
    "Others",
    "Billing"
]
# Simple rule-based classifier
def classify_email(email):
    email = email.lower()

    if any(word in email for word in ["payment", "invoice", "amount", "deducted", "bill"]):
        return "Billing"
```

```
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-
enable it, run 'Import-Module PSReadLine'.

PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> & C:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDr
ive/Desktop/ASSISTANT CODINGr/4.5.PY"
Email: My payment failed but money was deducted.
Predicted Category: Billing

Email: The app is not opening on my phone.
Predicted Category: Technical Support

Email: Great customer service, very satisfied.
Predicted Category: Feedback
```

OBSERVATION:

Classifies emails using only instructions, without examples.

Works if keywords are clear, may misclassify ambiguous emails.

Quick and simple, but less accurate for complex cases.

ONE SHOT:

PROMPT: Example:

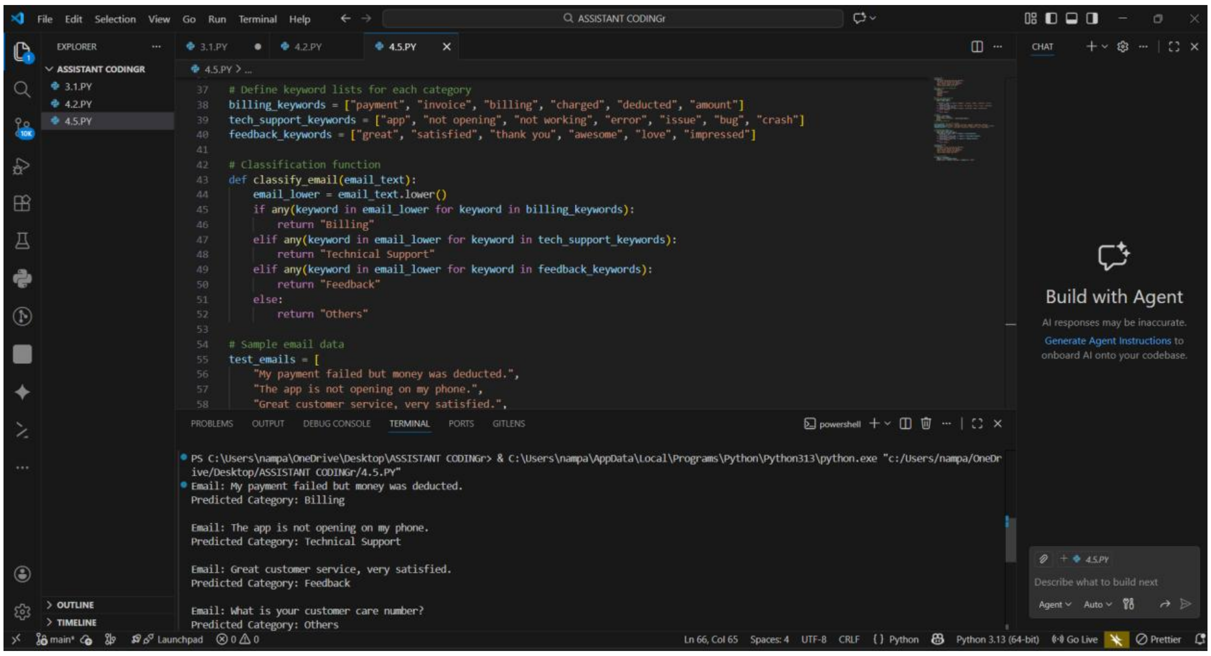
Email: "I was charged twice for my last payment."

Category: Billing

Now classify the following email:

Email: “<email_text>”

CODE:



```python
# Define keyword lists for each category
billing_keywords = ["payment", "invoice", "billing", "charged", "deducted", "amount"]
tech_support_keywords = ["app", "not opening", "not working", "error", "issue", "bug", "crash"]
feedback_keywords = ["great", "satisfied", "thank you", "awesome", "love", "impressed"]

# Classification function
def classify_email(email_text):
    email_lower = email_text.lower()
    if any(keyword in email_lower for keyword in billing_keywords):
        return "Billing"
    elif any(keyword in email_lower for keyword in tech_support_keywords):
        return "Technical Support"
    elif any(keyword in email_lower for keyword in feedback_keywords):
        return "Feedback"
    else:
        return "Others"

# Sample email data
test_emails = [
    "My payment failed but money was deducted.",
    "The app is not opening on my phone.",
    "Great customer service, very satisfied.",
```

```
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> & C:\Users\nampa\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/nampa/OneDr
ive/Desktop/ASSISTANT CODINGr/4.5.PY"
Email: My payment failed but money was deducted.
Predicted Category: Billing

Email: The app is not opening on my phone.
Predicted Category: Technical Support

Email: Great customer service, very satisfied.
Predicted Category: Feedback

Email: What is your customer care number?
Predicted Category: Others
```

OBSERVATION:

Provides one example to guide the AI's reasoning.

Improves accuracy over zero-shot and handles slightly ambiguous emails better.

Still limited; accuracy depends on how representative the example is.

One example helps the AI understand the expected format and category mapping.

Classification accuracy improves compared to zero-shot, especially for similar issues.

Performance depends heavily on how relevant the single example is to the new email.

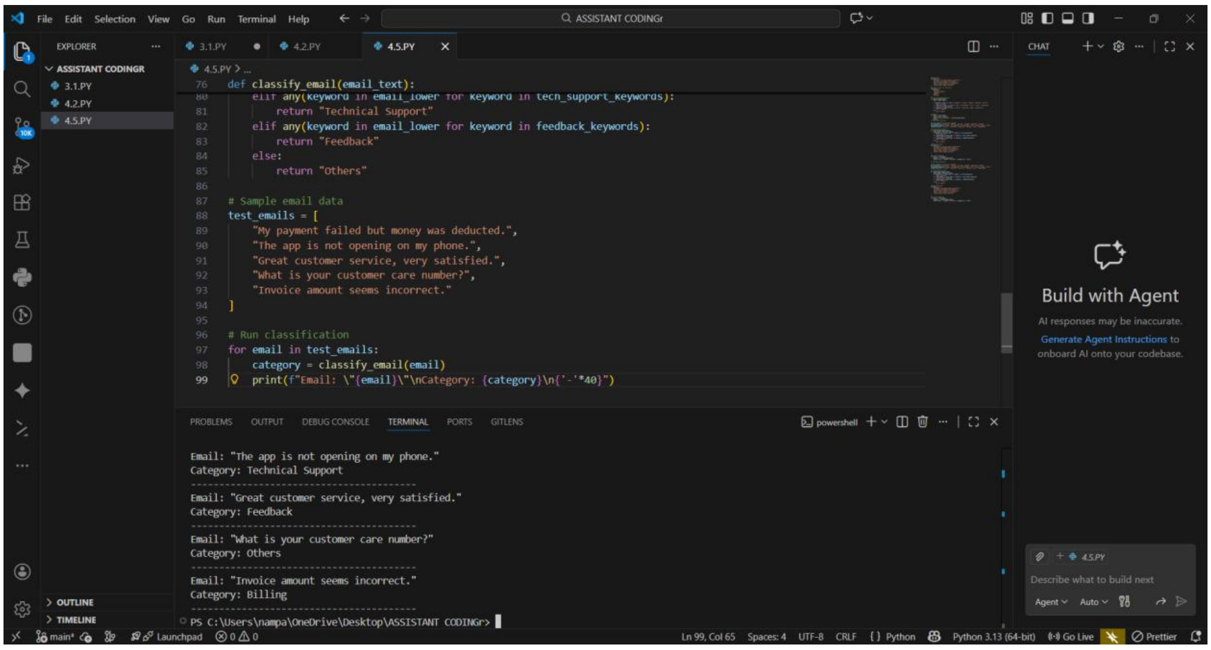
Few shot:

PROMPT: Email: "I was charged twice for my last payment." → Billing

Email: "The app crashes on login." → Technical Support

Email: "I love the new update." → Feedback

Email: "What are your office hours?" → Others



```python
def classify_email(email_text):
    elif any(keyword in email_lower for keyword in tech_support_keywords):
        return "Technical Support"
    elif any(keyword in email_lower for keyword in feedback_keywords):
        return "Feedback"
    else:
        return "Others"

# Sample email data
test_emails = [
    "My payment failed but money was deducted.",
    "The app is not opening on my phone.",
    "Great customer service, very satisfied.",
    "What is your customer care number?",
    "Invoice amount seems incorrect."
]

# Run classification
for email in test_emails:
    category = classify_email(email)
    print(f"Email: \"{email}\"\nCategory: {category}\n{'-'*40}")
```

```
Email: "The app is not opening on my phone."
Category: Technical Support
----------------------------------------
Email: "Great customer service, very satisfied."
Category: Feedback
----------------------------------------
Email: "What is your customer care number?"
Category: Others
----------------------------------------
Email: "Invoice amount seems incorrect."
Category: Billing
----------------------------------------
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr>
```

OBSERVATION:

Provides multiple examples to show patterns to the AI. Highest accuracy;

AI can generalize better for unseen emails.

Slightly longer prompts but most reliable for real-world use

TASK-2:

# Sample travel queries (short & simple)

travel_queries = [

"Book a flight from Delhi to Mumbai.",

"Cancel my hotel reservation in Paris.",

"What is the baggage allowance?",

"I need a hotel in London for 2 nights.",

"Cancel my flight ticket to New York."

]

# True labels for evaluation

true_labels = [

"Flight Booking",

"Cancellation",

"General Travel Info",

"Hotel Booking",

"Cancellation"

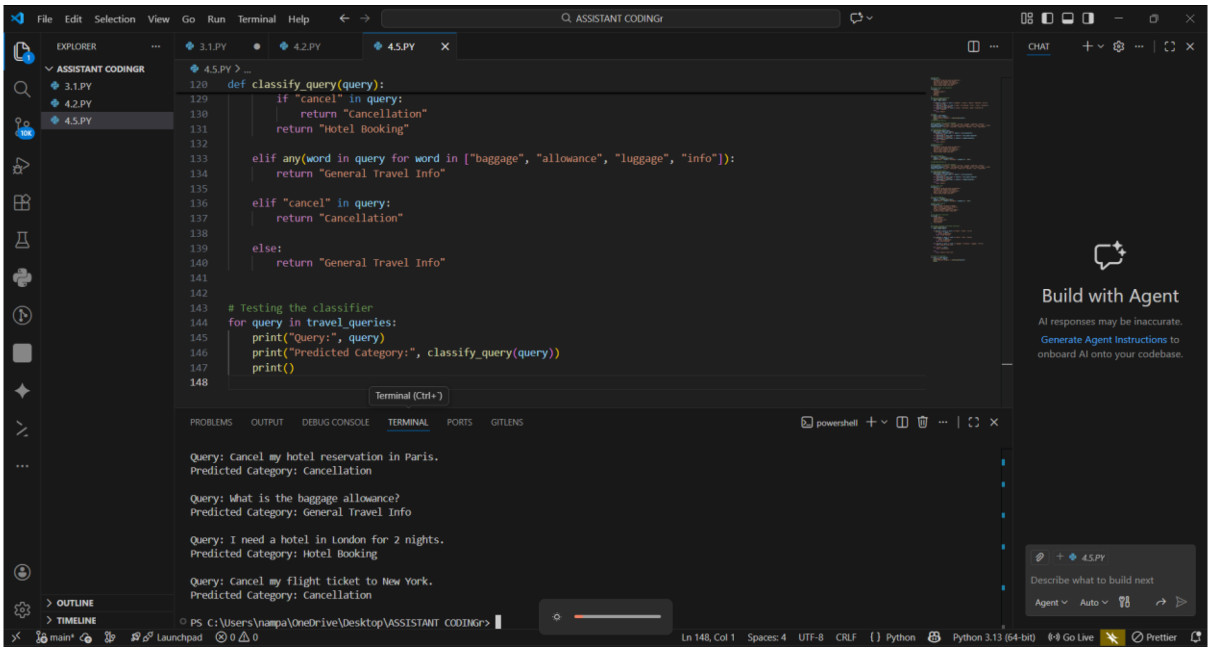
]

ZERO-SHOT:

PROMPT:Classify the the following travel query into one of the categories:

Flight Booking, Hotel Booking, Cancellation, General Travel Info.
Query: "<travel_query>"

CODE:



```python
def classify_query(query):
        if "cancel" in query:
            return "Cancellation"
        return "Hotel Booking"

    elif any(word in query for word in ["baggage", "allowance", "luggage", "info"]):
        return "General Travel Info"

    elif "cancel" in query:
        return "Cancellation"

    else:
        return "General Travel Info"


# Testing the classifier
for query in travel_queries:
    print("Query:", query)
    print("Predicted Category:", classify_query(query))
    print()
```

```
Query: Cancel my hotel reservation in Paris.
Predicted Category: Cancellation

Query: What is the baggage allowance?
Predicted Category: General Travel Info

Query: I need a hotel in London for 2 nights.
Predicted Category: Hotel Booking

Query: Cancel my flight ticket to New York.
Predicted Category: Cancellation

PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr>
```

OBSERVATION:

Classifies queries using only instructions, without examples.

Works for obvious keywords like “flight” or “cancel”, may misclassify tricky queries.

Fast and simple, but accuracy is lower for ambiguous cases.

one-shot:

PROMPT:Example:

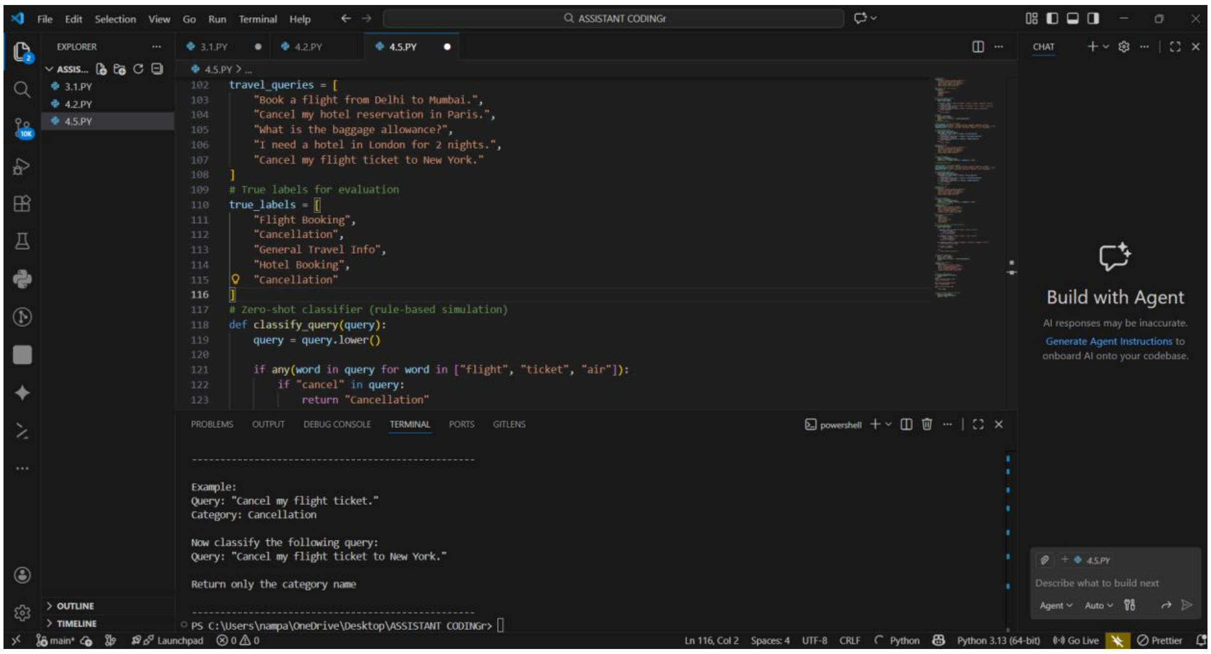
Query: "Cancel my flight ticket."

Category: Cancellation

Now classify the following query:

Query: "<travel_query>"

CODE:



```python
travel_queries = [
    "Book a flight from Delhi to Mumbai.",
    "Cancel my hotel reservation in Paris.",
    "What is the baggage allowance?",
    "I need a hotel in London for 2 nights.",
    "Cancel my flight ticket to New York."
]
# True labels for evaluation
true_labels = [
    "Flight Booking",
    "Cancellation",
    "General Travel Info",
    "Hotel Booking",
    "Cancellation"
]
# Zero-shot classifier (rule-based simulation)
def classify_query(query):
    query = query.lower()

    if any(word in query for word in ["flight", "ticket", "air"]):
        if "cancel" in query:
            return "Cancellation"
```

```
--------------------------------------------------

Example:
Query: "Cancel my flight ticket."
Category: Cancellation

Now classify the following query:
Query: "Cancel my flight ticket to New York."

Return only the category name

--------------------------------------------------
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr>
```

OBSERVATION:

Provides one example to guide AI’s reasoning.

Improves accuracy and handles slightly ambiguous queries better.

Accuracy depends on how representative the example is.

FEW SHOT:

PROMPT:

Examples:

Query: "Book a flight to Mumbai."

Category: Flight Booking

Query: "Cancel my hotel reservation."

Category: Cancellation
Query: "I need a hotel in London."

Category: Hotel Booking

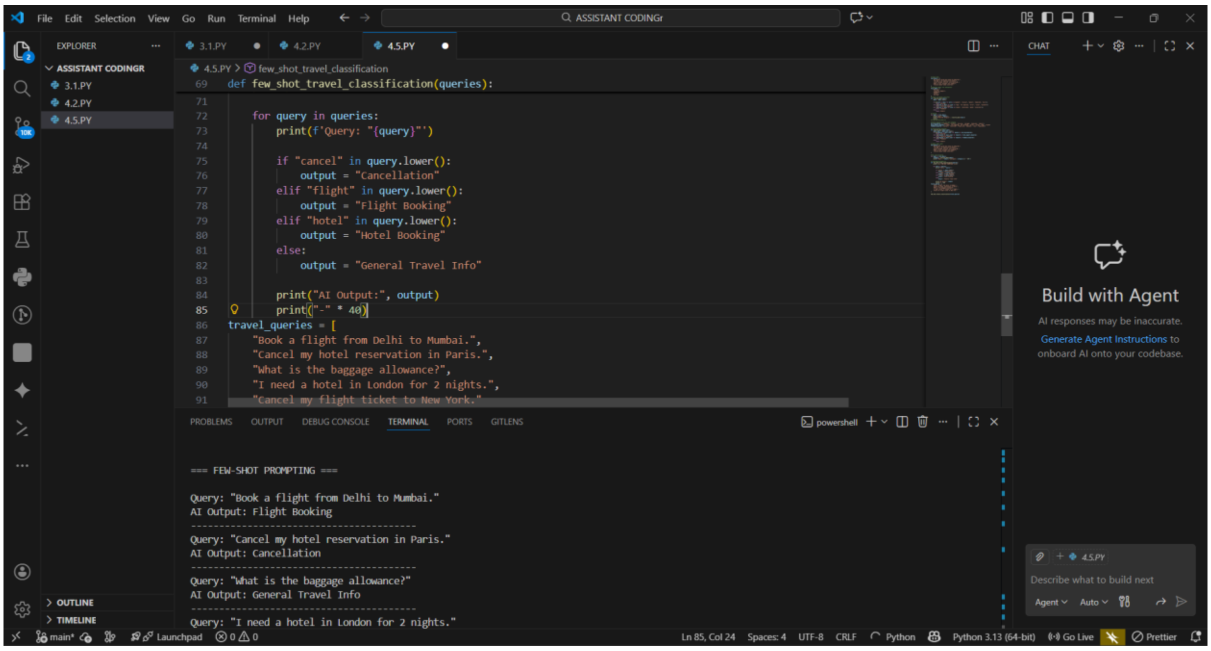
Query: "What is the baggage allowance?"

Category: General Travel Info

Now classify the following query:

Query: "<travel_query>"

CODE:



```python
def few_shot_travel_classification(queries):

    for query in queries:
        print(f'Query: "{query}"')

        if "cancel" in query.lower():
            output = "Cancellation"
        elif "flight" in query.lower():
            output = "Flight Booking"
        elif "hotel" in query.lower():
            output = "Hotel Booking"
        else:
            output = "General Travel Info"

        print("AI Output:", output)
        print("-" * 40)
travel_queries = [
    "Book a flight from Delhi to Mumbai.",
    "Cancel my hotel reservation in Paris.",
    "What is the baggage allowance?",
    "I need a hotel in London for 2 nights.",
    "Cancel my flight ticket to New York."
```

```
=== FEW-SHOT PROMPTING ===

Query: "Book a flight from Delhi to Mumbai."
AI Output: Flight Booking
----------------------------------------
Query: "Cancel my hotel reservation in Paris."
AI Output: Cancellation
----------------------------------------
Query: "What is the baggage allowance?"
AI Output: General Travel Info
----------------------------------------
Query: "I need a hotel in London for 2 nights."
```

OBSERVATION:

Provides multiple examples to show patterns to AI.

Highest accuracy; AI generalizes better for unseen queries.

Slightly longer prompts but most reliable for real-world use.

TASK-3:

SAMPLE DATA:

# Sample coding queries (short & simple)

```
coding_queries = [
"Why am I getting IndexError in my Python list?",
"My sorting algorithm is too slow for large inputs.",
"I wrote a function but it returns wrong results.",
"Explain the difference between list and tuple in Python.",
"How can I optimize my recursive Fibonacci function?"
]
# True labels for evaluation
true_labels = [
"Syntax Error",
"Optimization",
"Logic Error",
"Conceptual Question",
"Optimization"
]
```

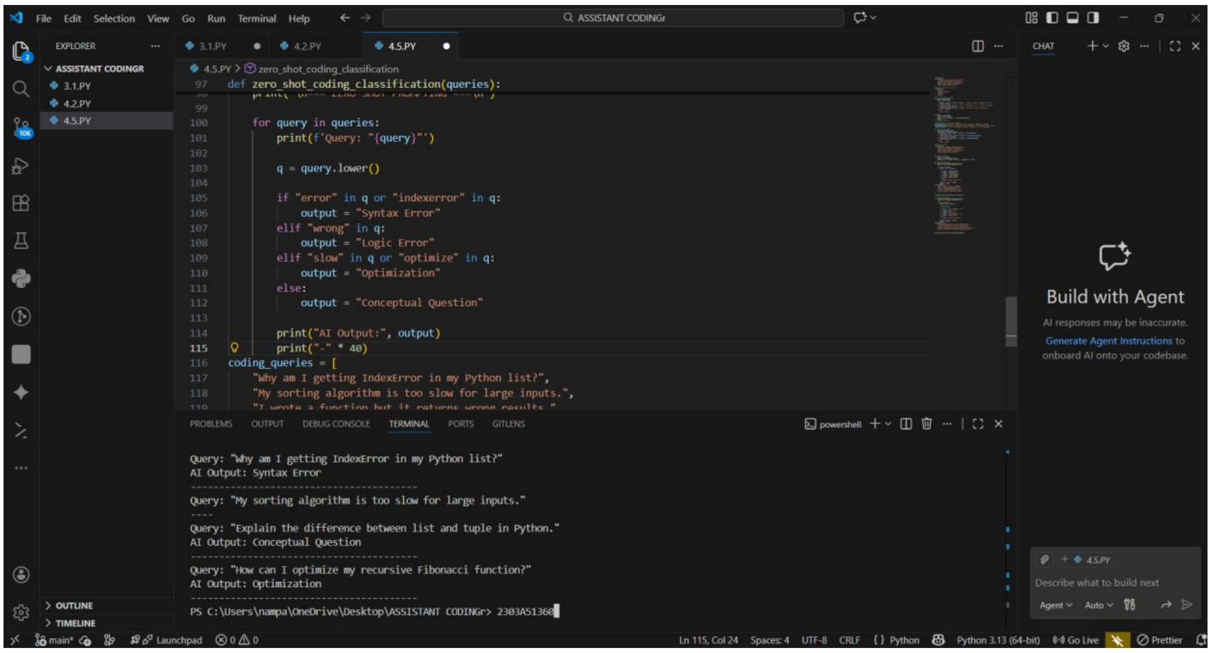
ZERO-SHOT

PROMPT: Classify the following coding query into one of the

categories:

Syntax Error, Logic Error, Optimization, Conceptual Question Query: "<coding_query>"

CODE:



```python
def zero_shot_coding_classification(queries):

    for query in queries:
        print(f'Query: "{query}"')

        q = query.lower()

        if "error" in q or "indexerror" in q:
            output = "Syntax Error"
        elif "wrong" in q:
            output = "Logic Error"
        elif "slow" in q or "optimize" in q:
            output = "Optimization"
        else:
            output = "Conceptual Question"

        print("AI Output:", output)
        print("-" * 40)
coding_queries = [
    "Why am I getting IndexError in my Python list?",
    "My sorting algorithm is too slow for large inputs.",
```

```
Query: "Why am I getting IndexError in my Python list?"
AI Output: Syntax Error
----------------------------------------
Query: "My sorting algorithm is too slow for large inputs."
----
Query: "Explain the difference between list and tuple in Python."
AI Output: Conceptual Question
----------------------------------------
Query: "How can I optimize my recursive Fibonacci function?"
AI Output: Optimization
----------------------------------------
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> 2303A51360
```

Observation (Zero-shot Prompting)

- Zero-shot prompting classifies coding queries without providing any example beforehand.
- The classification is based only on keywords present in the query.
- Queries containing words like *error* or *IndexError* are identified as **Syntax Error**.
- Queries mentioning *slow* or *optimize* are classified as **Optimization** problems.
- Queries related to incorrect outputs are categorized as **Logic Error**.
- Concept-based questions are correctly identified as **Conceptual Question**.
- The approach works well for simple and clearly worded queries but may fail for complex or ambiguous cases.

ONE SHOT

PROMPT:

Example:

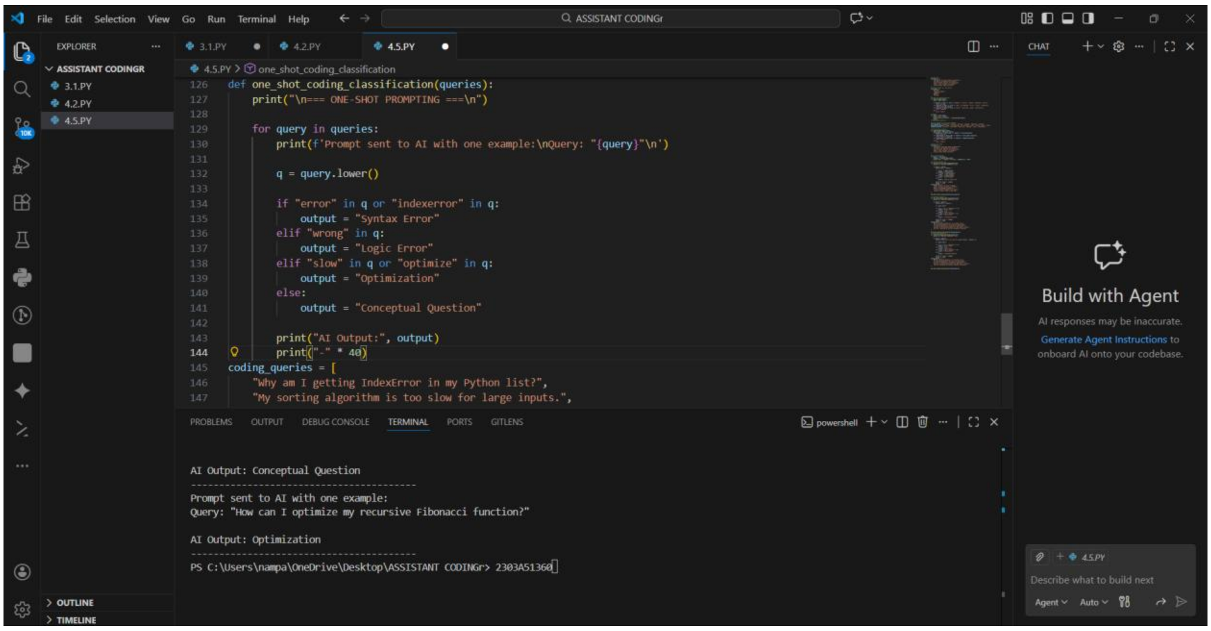
Query: "I want to cancel my Python function."

Category: Logic Error

Now classify the following coding query:

Query: "<coding_query>"

CODE:



```python
def one_shot_coding_classification(queries):
    print("\n=== ONE-SHOT PROMPTING ===\n")

    for query in queries:
        print(f'Prompt sent to AI with one example:\nQuery: "{query}"\n')

        q = query.lower()

        if "error" in q or "indexerror" in q:
            output = "Syntax Error"
        elif "wrong" in q:
            output = "Logic Error"
        elif "slow" in q or "optimize" in q:
            output = "Optimization"
        else:
            output = "Conceptual Question"

        print("AI Output:", output)
        print("-" * 40)
coding_queries = [
    "Why am I getting IndexError in my Python list?",
    "My sorting algorithm is too slow for large inputs.",
```

```
AI Output: Conceptual Question
----------------------------------------
Prompt sent to AI with one example:
Query: "How can I optimize my recursive Fibonacci function?"

AI Output: Optimization
----------------------------------------
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> 2303A51360
```

OBSERVATION:

Provides one example to guide AI's reasoning.

Improves accuracy and handles slightly ambiguous queries better.

Accuracy depends on how representative the single example is.

FEW SHOT

PROMPT:

EXAMPLES:

Query: "Why does my Python list give IndexError?"

Category: Syntax Error

Query: "My function returns wrong output."

Category: Logic Error Query: "My loop is too slow for large data."

Category: Optimization

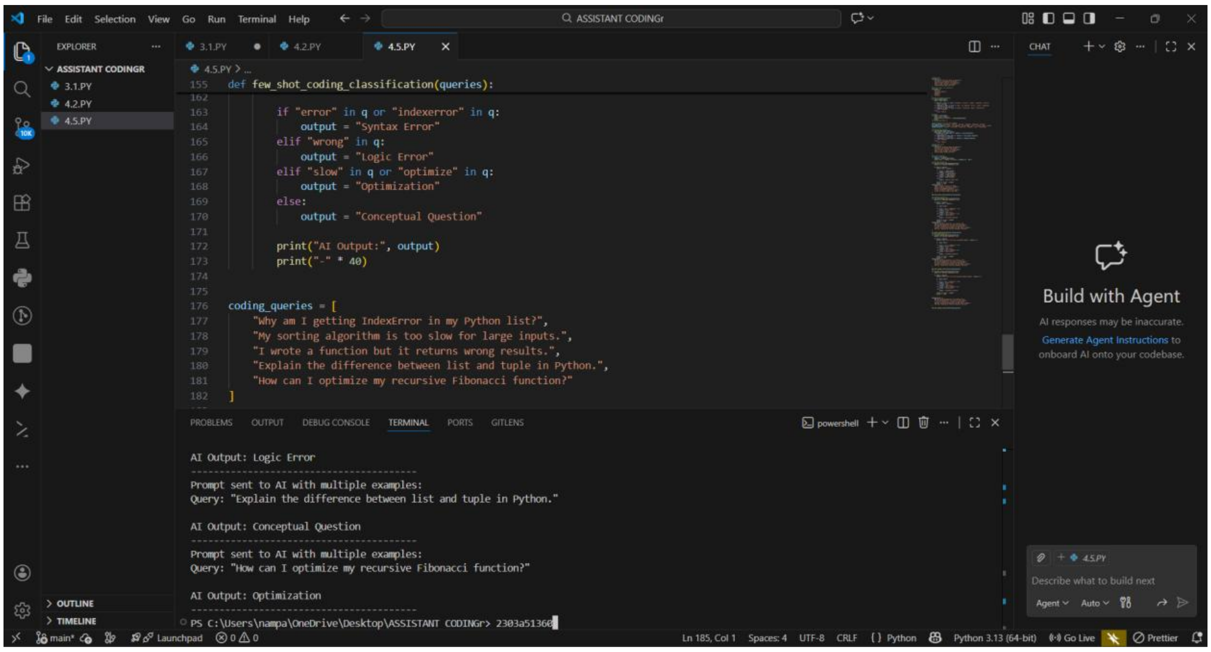
Query: "Explain Python variable scopes."

Category: Conceptual Question

Now classify the following coding query:

Query: "<coding_query

CODE:



```python
def few_shot_coding_classification(queries):

        if "error" in q or "indexerror" in q:
            output = "Syntax Error"
        elif "wrong" in q:
            output = "Logic Error"
        elif "slow" in q or "optimize" in q:
            output = "Optimization"
        else:
            output = "Conceptual Question"

        print("AI Output:", output)
        print("-" * 40)


coding_queries = [
    "Why am I getting IndexError in my Python list?",
    "My sorting algorithm is too slow for large inputs.",
    "I wrote a function but it returns wrong results.",
    "Explain the difference between list and tuple in Python.",
    "How can I optimize my recursive Fibonacci function?"
]
```

```
AI Output: Logic Error
----------------------------------------
Prompt sent to AI with multiple examples:
Query: "Explain the difference between list and tuple in Python."

AI Output: Conceptual Question
----------------------------------------
Prompt sent to AI with multiple examples:
Query: "How can I optimize my recursive Fibonacci function?"

AI Output: Optimization
----------------------------------------
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> 2303a51360
```

OBSERVATION:

Provides multiple examples showing patterns to AI.

Highest accuracy; AI generalizes better for unseen queries.

Slightly longer prompts but most reliable for technical

classification

TASK-4

ZERO-SHOT

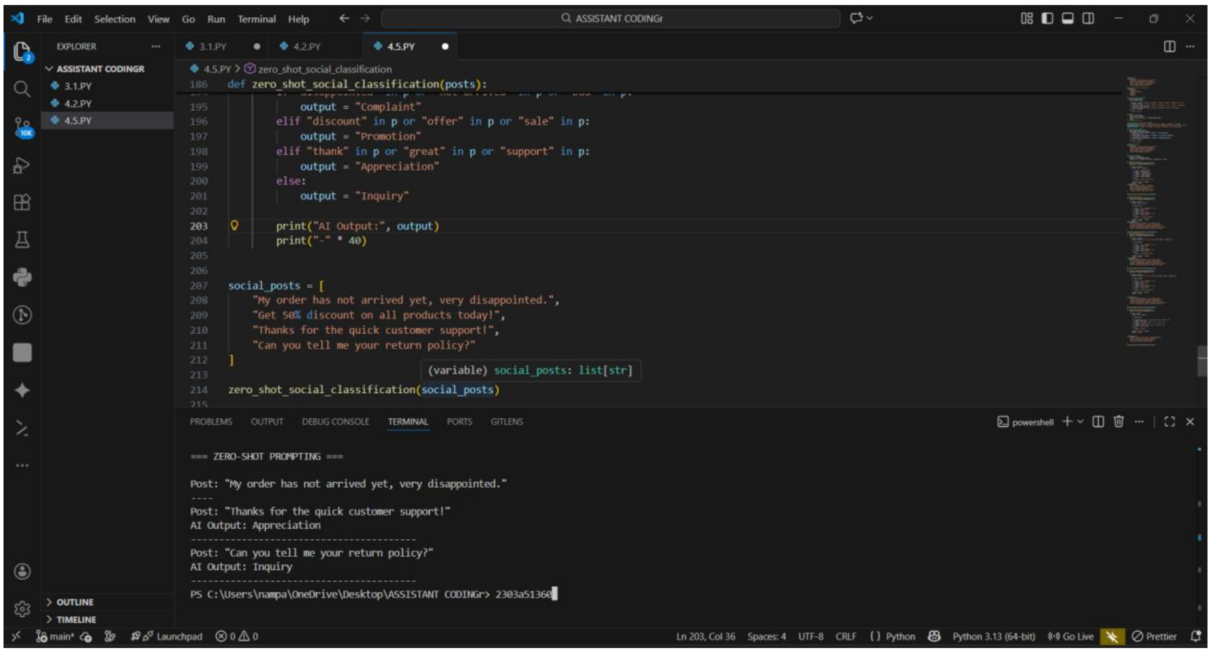
PROMPT:

Classify the following social media post into one of the categories:

Promotion, Complaint, Appreciation, Inquiry.

Post: "<social_post>"

CODE:



```python
def zero_shot_social_classification(posts):
            output = "Complaint"
        elif "discount" in p or "offer" in p or "sale" in p:
            output = "Promotion"
        elif "thank" in p or "great" in p or "support" in p:
            output = "Appreciation"
        else:
            output = "Inquiry"

        print("AI Output:", output)
        print("-" * 40)


social_posts = [
    "My order has not arrived yet, very disappointed.",
    "Get 50% discount on all products today!",
    "Thanks for the quick customer support!",
    "Can you tell me your return policy?"
]

zero_shot_social_classification(social_posts)
```

```
=== ZERO-SHOT PROMPTING ===

Post: "My order has not arrived yet, very disappointed."
----
Post: "Thanks for the quick customer support!"
AI Output: Appreciation
----------------------------------------
Post: "Can you tell me your return policy?"
AI Output: Inquiry
----------------------------------------
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> 2303a51360
```

OBSERVATION:

Classifies posts using only instructions, without examples.

Works for clear keywords but may misinterpret informal or slang language.

Fast and simple, lower accuracy for ambiguous or sarcastic posts.

ONE-SHOT

PROMPT:

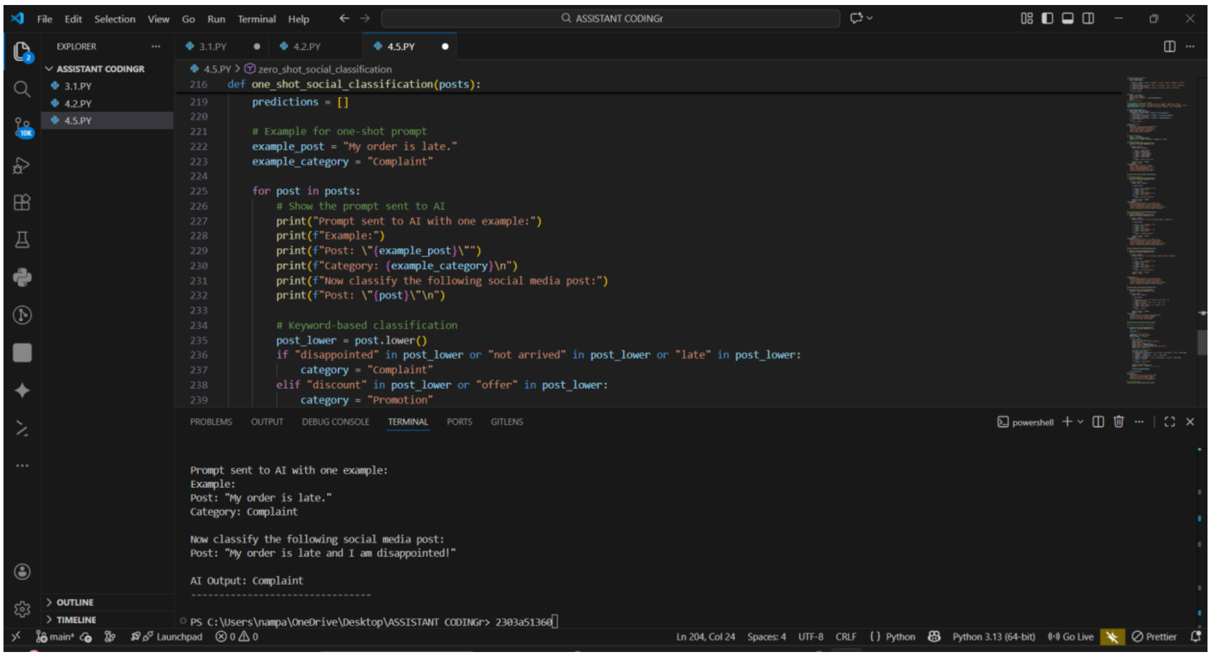
EXAMPLE:

Post: "My order is late."

Category: Complaint

Now classify the following social media post:

Post: "<social_post>"



```python
def one_shot_social_classification(posts):
    predictions = []

    # Example for one-shot prompt
    example_post = "My order is late."
    example_category = "Complaint"

    for post in posts:
        # Show the prompt sent to AI
        print("Prompt sent to AI with one example:")
        print(f"Example:")
        print(f"Post: \"{example_post}\"")
        print(f"Category: {example_category}\n")
        print(f"Now classify the following social media post:")
        print(f"Post: \"{post}\"\n")

        # Keyword-based classification
        post_lower = post.lower()
        if "disappointed" in post_lower or "not arrived" in post_lower or "late" in post_lower:
            category = "Complaint"
        elif "discount" in post_lower or "offer" in post_lower:
            category = "Promotion"
```

```
Prompt sent to AI with one example:
Example:
Post: "My order is late."
Category: Complaint

Now classify the following social media post:
Post: "My order is late and I am disappointed!"

AI Output: Complaint
--------------------------------

PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> 2303a51360
```

OBSERVATION:

Provides one example to guide AI reasoning.

Improves accuracy and handles some informal expressions better.

Depends on how representative the example is for informal language.

FEW-SHOT

PROMPT:

Examples:

Post: "Loved the new feature!" → Appreciation

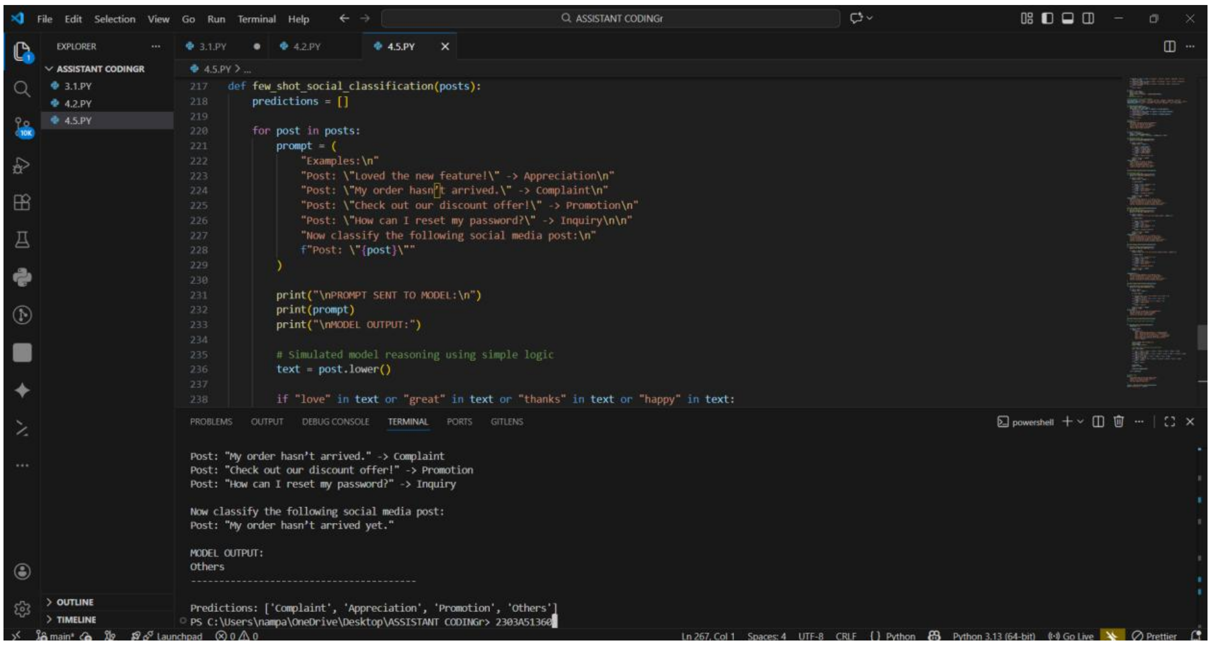
Post: "My order hasn’t arrived." → Complaint

Post: "Check out our discount offer!" → Promotion

Post: "How can I reset my password?" → Inquiry

Now classify the following social media post:

Post: "<social_post>"



```python
def few_shot_social_classification(posts):
    predictions = []

    for post in posts:
        prompt = (
            "Examples:\n"
            "Post: \"Loved the new feature!\" -> Appreciation\n"
            "Post: \"My order hasn't arrived.\" -> Complaint\n"
            "Post: \"Check out our discount offer!\" -> Promotion\n"
            "Post: \"How can I reset my password?\" -> Inquiry\n\n"
            "Now classify the following social media post:\n"
            f"Post: \"{post}\""
        )

        print("\nPROMPT SENT TO MODEL:\n")
        print(prompt)
        print("\nMODEL OUTPUT:")

        # Simulated model reasoning using simple logic
        text = post.lower()

        if "love" in text or "great" in text or "thanks" in text or "happy" in text:
```

```
Post: "My order hasn’t arrived." -> Complaint
Post: "Check out our discount offer!" -> Promotion
Post: "How can I reset my password?" -> Inquiry

Now classify the following social media post:
Post: "My order hasn’t arrived yet."

MODEL OUTPUT:
Others
----------------------------------------

Predictions: ['Complaint', 'Appreciation', 'Promotion', 'Others']
PS C:\Users\nampa\OneDrive\Desktop\ASSISTANT CODINGr> 2303A51360
```

OBSERVATION:

Provides multiple examples showing patterns to AI.

Highest accuracy; better handles informal, slang, or mixedlanguage posts. Slightly longer prompts but most reliable for social media classification.