

## Assignment-4.5

NAME: SRAVANI

BAtch-30

Roll-No:2303A510G7

Advanced Prompt Engineering: Zero-shot, one-shot, and few-shot  
Techniques

TASK-1:

ZERO-SHOT:

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:

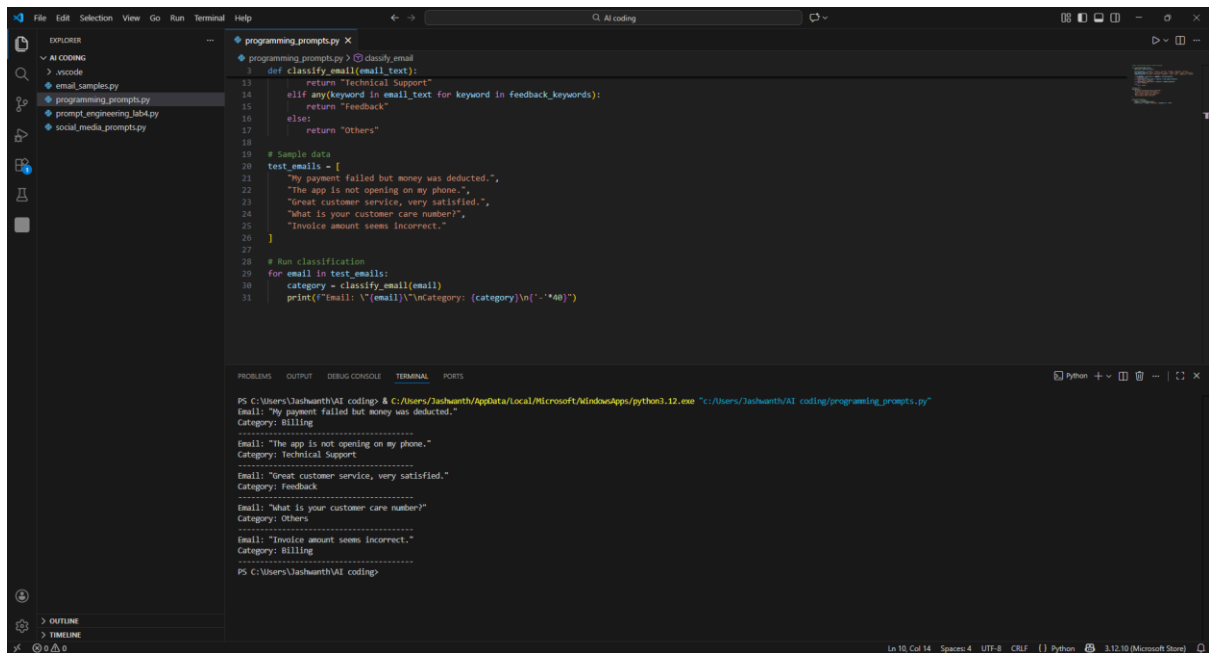
Classify the following email into one of the categories:

Billing, Technical Support, Feedback, Others.

Email: "<email\_text>"

Return only the category name.

## CODE:



```
1 def classify_email(email_text):
2     return "Technical Support"
3
4 elif any(keyword in email_text for keyword in feedback_keywords):
5     return "Feedback"
6 else:
7     return "Others"
8
9 # Sample data
10 test_emails = [
11     "My payment failed but money was deducted.",
12     "The app is not opening on my phone.",
13     "Great customer service, very satisfied.",
14     "What is your customer care number?",
15     "Invoice amount seems incorrect."
16 ]
17
18 # Run classification
19 for email in test_emails:
20     category = classify_email(email)
21     print(f"Email: '{email}'\nCategory: {category}\n{'-'*40}")
```

PS C:\Users\Jashwanth\AI coding> C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe "C:\Users\Jashwanth\AI coding\programming\_prompts.py"

Email: "My payment failed but money was deducted."  
Category: Billing

-----

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\Jashwanth\AI coding>

## OBSERVATION:

Uses instructions alone to classify emails—no examples provided. Effective for clear keywords, but prone to errors with vague or nuanced content. Fast and straightforward, though less reliable for complex or ambiguous cases.

## ONE-SHOT :

PROMPT: Example:

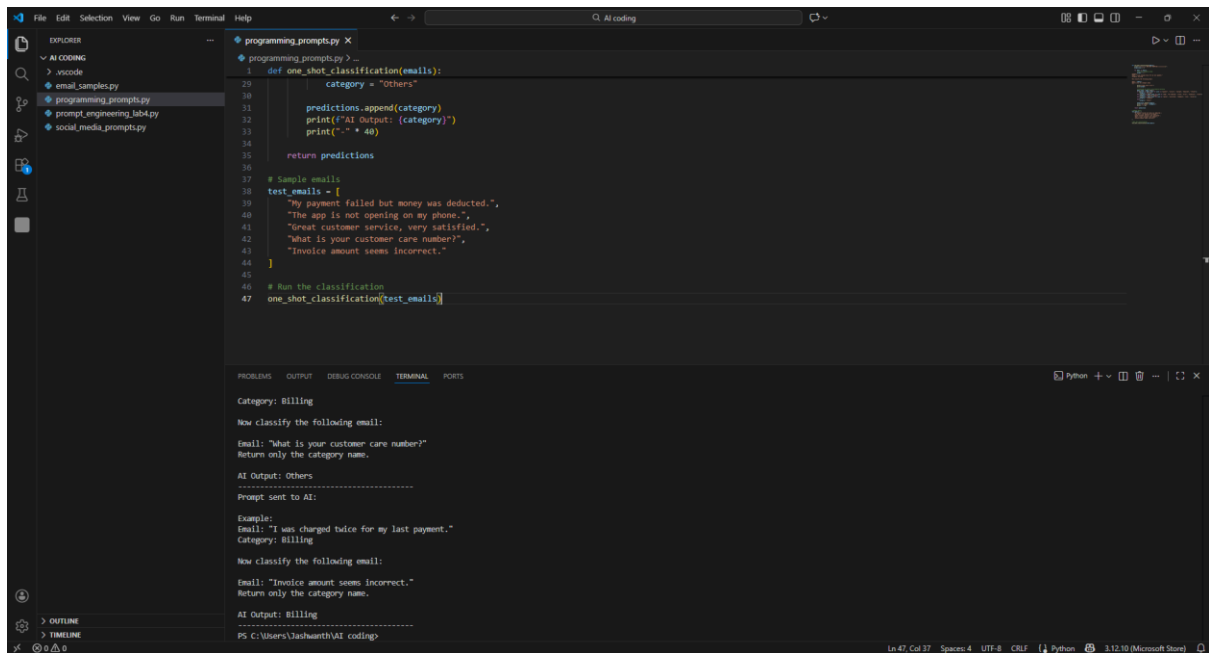
Email: "I was charged twice for my last payment."

Category: Billing

Now classify the following email:

Email: "<email\_text>"

## CODE:



```
1 def one_shot_classification(emails):
2     category = "Others"
3
4     predictions.append(category)
5     print(f"AI Output: {category}")
6     print("-" * 40)
7
8     return predictions
9
10 # Sample emails
11 test_emails = [
12     "My payment failed but money was deducted.",
13     "The app is not opening on my phone.",
14     "Great customer service, very satisfied.",
15     "What is your customer care number?",
16     "Invoice amount seems incorrect."
17 ]
18
19 # Run the classification
20 one_shot_classification(test_emails)
```

Category: Billing

Now classify the following email:

Email: "What is your customer care number?"  
Return only the category name.

AI Output: Others

-----

Prompt sent to AI:

Example:

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

Now classify the following email:

Email: "Invoice amount seems incorrect."  
Return only the category name.

AI Output: Billing

-----

PS C:\Users\Jashwanth\AI\_coding>

## OBSERVATION:

Uses a single example to guide classification. Accuracy improves compared to zero-shot, especially for similar cases, and the AI better handles mildly ambiguous emails. However, performance is limited and depends heavily on how representative that one example is.

## FEW-SHOT:

### PROMPT:

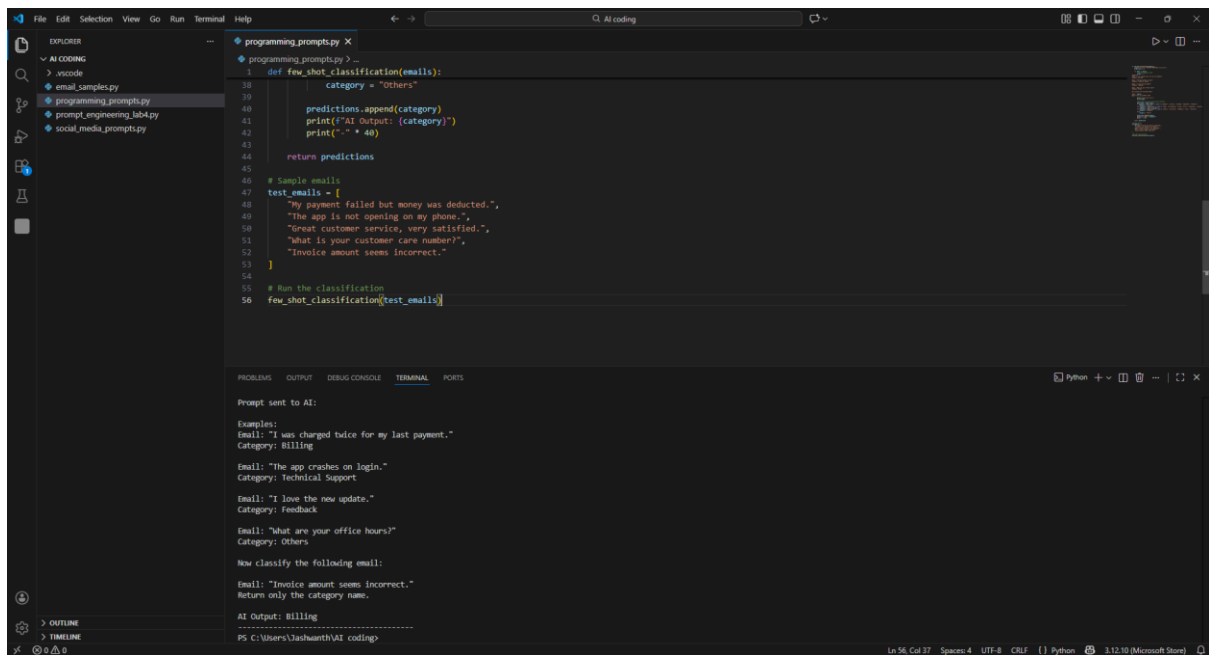
Email: "I was charged twice for my last payment." → Billing

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

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

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

CODE:



```
1 def few_shot_classification(emails):
2     category = "Others"
3
4     predictions.append(category)
5     print(f"AI Output: {category}")
6     print("-" * 40)
7
8     return predictions
9
10 # Sample emails
11 test_emails = [
12     "My payment failed but money was deducted.",
13     "The app is not opening on my phone.",
14     "Great customer service, very satisfied.",
15     "What is your customer care number?",
16     "Invoice amount seems incorrect."
17 ]
18
19 # Run the classification
20 few_shot_classification(test_emails)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Prompt sent to AI:

Examples:

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

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

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

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

Now classify the following email:

Email: "Invoice amount seems incorrect."  
Return only the category name.

AI Output: Billing

PS C:\Users\Jashwanth\AI coding>

OBSERVATION:

Uses multiple examples to establish clear patterns, enabling the AI to generalize effectively to new emails. Delivers the highest accuracy among prompting styles, though prompts are longer. Most dependable for real-world applications.

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"

]

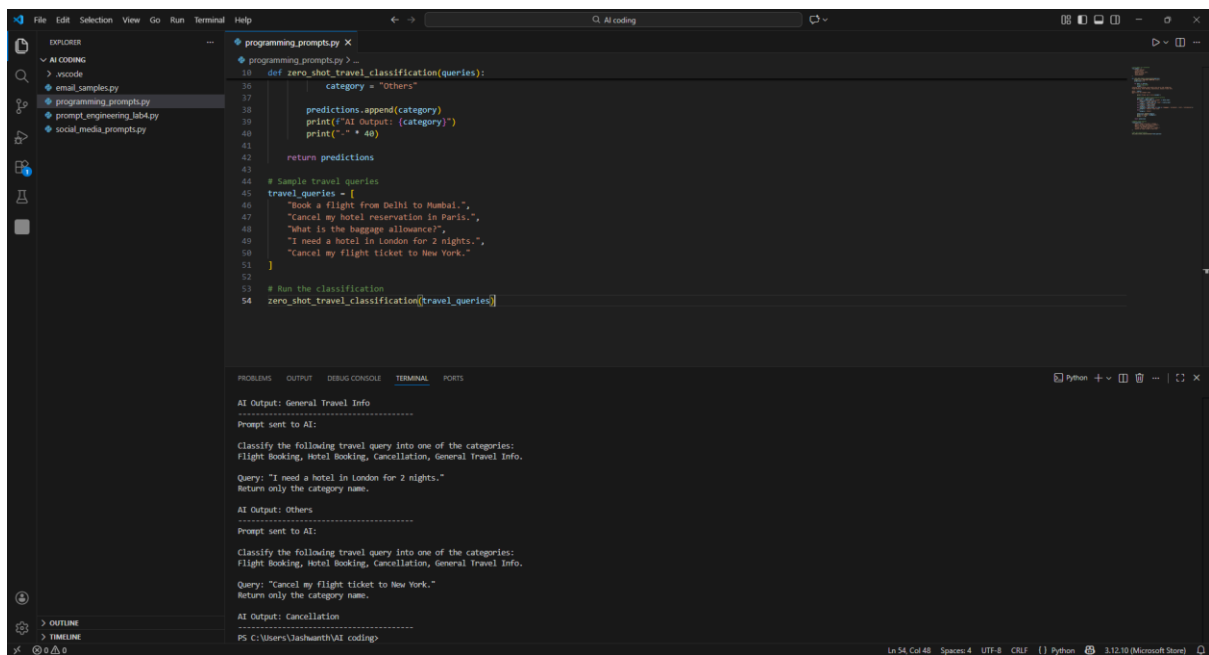
ZERO-SHOT:

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

Flight Booking, Hotel Booking, Cancellation, General Travel Info.

Query: "<travel\_query>"

CODE:



```
def zero_shot_travel_classification(queries):
    category = "Others"
    predictions.append(category)
    print(f"AI Output: {category}")
    print("-" * 40)
    return predictions

# Sample travel queries
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."
]

# Run the classification
zero_shot_travel_classification(travel_queries)
```

AI Output: General Travel Info

Prompt sent to AI:

Classify the following travel query into one of the categories:  
Flight Booking, Hotel Booking, Cancellation, General Travel Info.

Query: "I need a hotel in London for 2 nights."  
Return only the category name.

AI Output: Others

Prompt sent to AI:

Classify the following travel query into one of the categories:  
Flight Booking, Hotel Booking, Cancellation, General Travel Info.

Query: "Cancel my flight ticket to New York."  
Return only the category name.

AI Output: Cancellation

PS C:\Users\Jashwanth\AI coding>

## OBSERVATION:

Relies solely on instructions without examples. Performs well for clear keywords like “flight” or “cancel,” but struggles with nuanced or ambiguous queries. Quick and straightforward, though less reliable when context is subtle.

## ONE-SHOT:

PROMPT: Example:

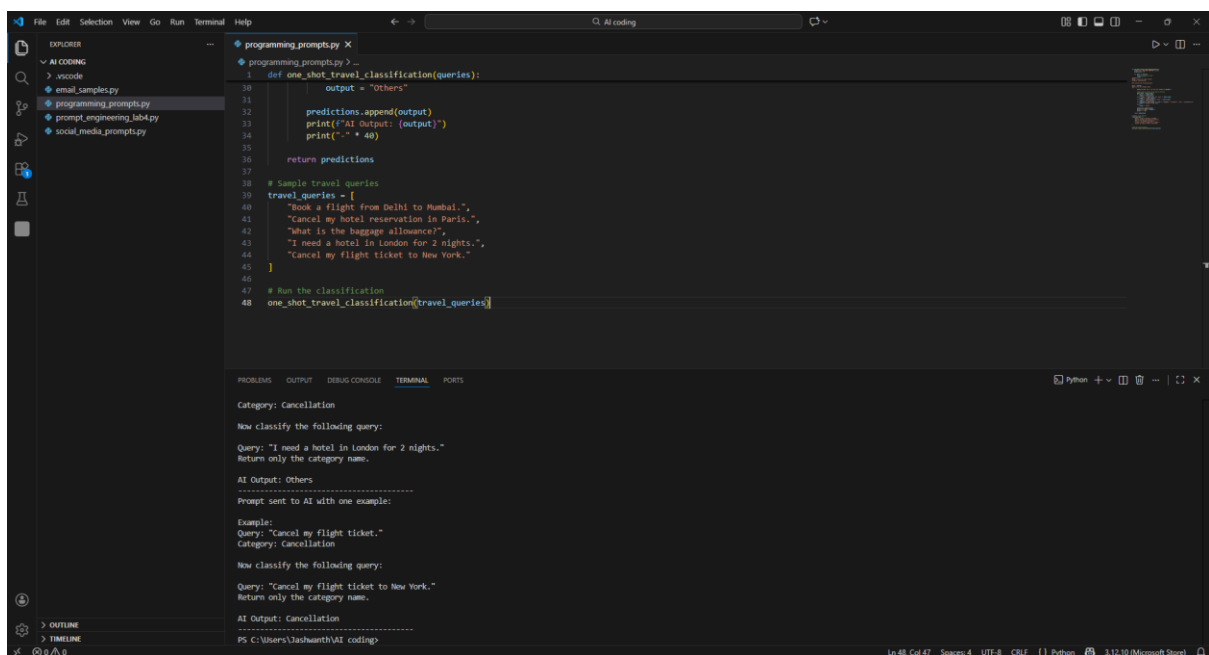
Query: "Cancel my flight ticket."

Category: Cancellation

Now classify the following query:

Query: "<travel\_query>"

## CODE:



```
File Edit Selection View Go Run Terminal Help
programming_prompts.py X
programming_prompts.py
1 def one_shot_travel_classification(queries):
2     output = "Others"
3
4     predictions.append(output)
5     print(f"AI Output: {output}")
6     print("-" * 40)
7
8     return predictions
9
10 # Sample travel queries
11 travel_queries = [
12     "Book a flight from Delhi to Mumbai.",
13     "Cancel my hotel reservation in Paris.",
14     "What is the baggage allowance?",
15     "I need a hotel in London for 2 nights.",
16     "Cancel my flight ticket to New York."
17 ]
18
19 # Run the classification
20 one_shot_travel_classification(travel_queries)
```

Category: Cancellation

Now classify the following query:

Query: "I need a hotel in London for 2 nights."

Return only the category name.

AI Output: Others

-----

Prompt sent to AI with one example:

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.

AI Output: Cancellation

-----

PS C:\Users\Jashwanth\AI\_coding>

## OBSERVATION:

Uses a single example to guide reasoning, which improves accuracy compared to zero-shot and helps with mildly ambiguous queries. Effectiveness depends largely on how representative the chosen 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

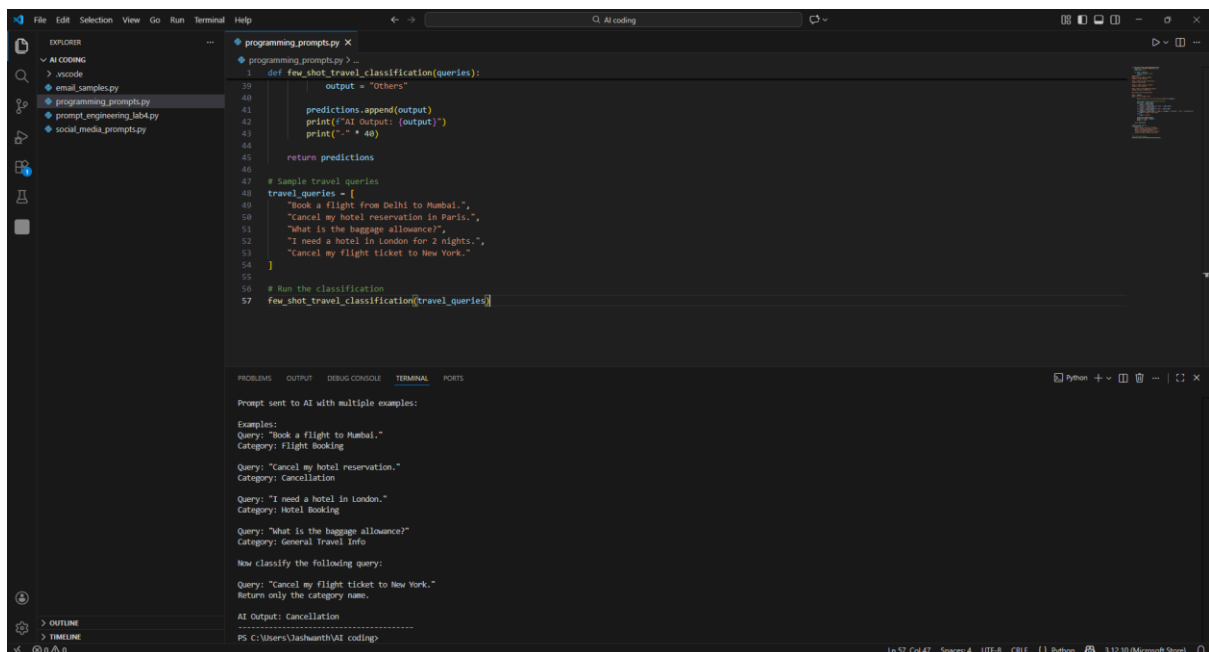
Query: "What is the baggage allowance?"

Category: General Travel Info

Now classify the following query:

Query: "<travel\_query>"

CODE:



```
1 def few_shot_travel_classification(queries):
2     output = "Others"
3     predictions.append(output)
4     print(f"AI Output: {output}")
5     print("-" * 40)
6     return predictions
7
8 # Sample travel queries
9 travel_queries = [
10     "Book a flight from Delhi to Mumbai.",
11     "Cancel my hotel reservation in Paris.",
12     "What is the baggage allowance?",
13     "I need a hotel in London for 2 nights.",
14     "Cancel my flight ticket to New York."
15 ]
16
17 # Run the classification
18 few_shot_travel_classification(travel_queries)
```

Prompt sent to AI with multiple examples:

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: "Cancel my flight ticket to New York."  
Return only the category name.

AI Output: Cancellation

OBSERVATION:

Offers multiple examples to establish clear patterns, enabling stronger generalization to new queries. Achieves the highest accuracy among prompting styles, with longer prompts but the most dependable performance for real-world scenarios.

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"

]

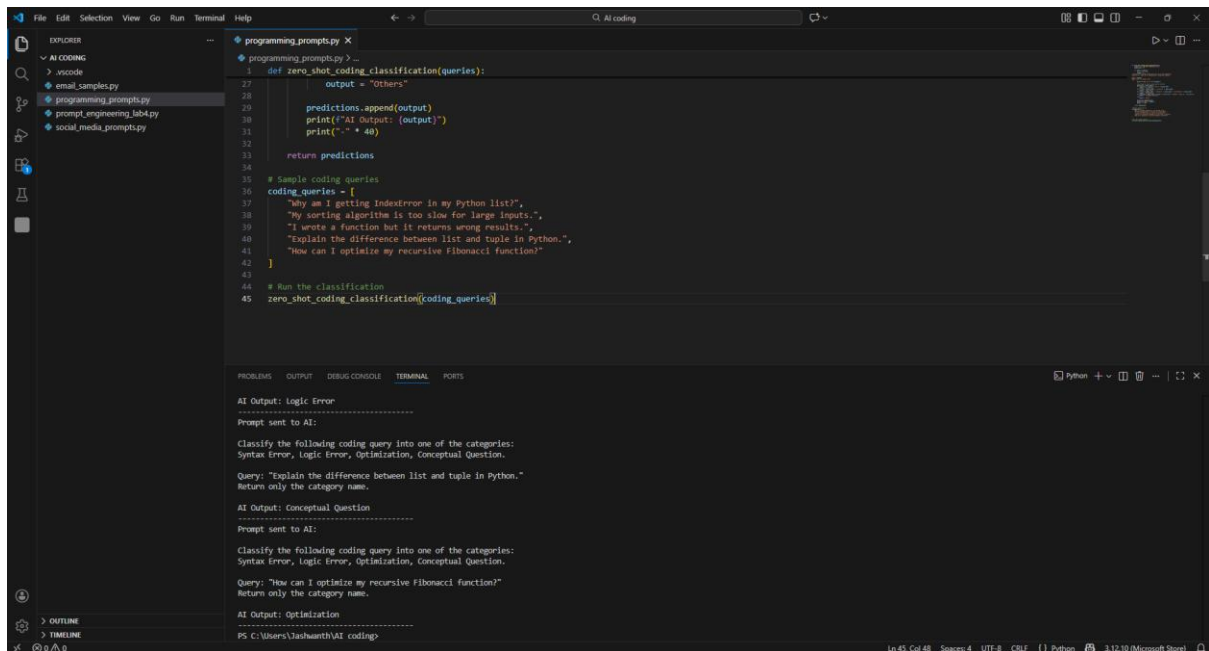
ZERO-SHOT:

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

Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: "<coding\_query>"

## CODE:



```
1 def zero_shot_coding_classification(queries):
2     output = "Others"
3
4     predictions.append(output)
5     print(f"AI Output: {output}")
6     print("-" * 40)
7
8     return predictions
9
10 # Sample coding queries
11 coding_queries = [
12     "Why am I getting IndexError in my Python list?",
13     "My sorting algorithm is too slow for large inputs.",
14     "I wrote a function but it returns wrong results.",
15     "Explain the difference between list and tuple in Python.",
16     "How can I optimize my recursive Fibonacci function?"
17 ]
18
19 # Run the classification
20 zero_shot_coding_classification(coding_queries)
```

AI Output: Logic Error

Prompt sent to AI:

Classify the following coding query into one of the categories:  
Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: "Explain the difference between list and tuple in Python."  
Return only the category name.

AI Output: Conceptual Question

Prompt sent to AI:

Classify the following coding query into one of the categories:  
Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: "How can I optimize my recursive Fibonacci function?"  
Return only the category name.

AI Output: Optimization

## OBSERVATION:

Classifies programming queries using predefined categories based on keyword cues. Works well for clear technical terms like “IndexError” or “optimize,” but may misclassify conceptual or ambiguous queries. Fast and rule-based, but lacks deeper semantic understanding.

## ONE-SHOT:

### PROMPT:

### Example:

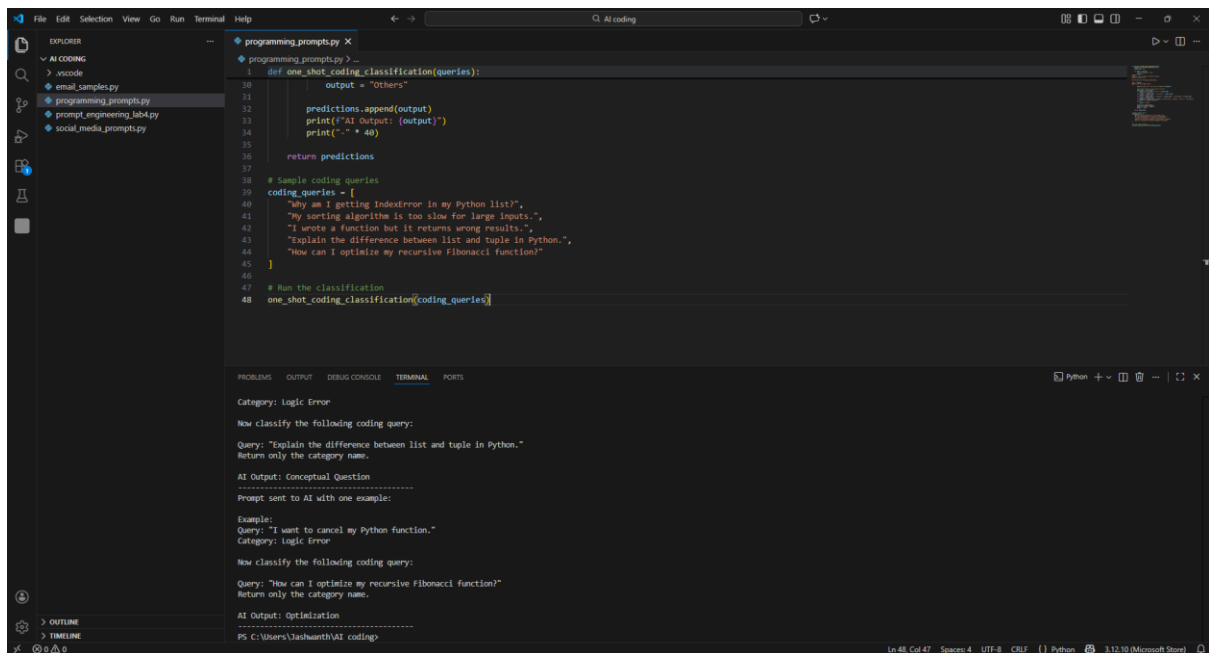
Query: "I want to cancel my Python function."

Category: Logic Error

Now classify the following coding query:

Query: "<coding\_query>"

CODE:



```
1 def one_shot_coding_classification(queries):
2     output = "Others"
3     predictions.append(output)
4     print(f"AI Output: {output}")
5     print(f"--" * 40)
6
7     return predictions
8
9 # Sample coding queries
10 coding_queries = [
11     "Why am I getting IndexError in my Python list?",
12     "My sorting algorithm is too slow for large inputs.",
13     "I wrote a function but it returns wrong results.",
14     "Explain the difference between list and tuple in Python.",
15     "How can I optimize my recursive Fibonacci function?"
16 ]
17
18 # Run the classification
19 one_shot_coding_classification(coding_queries)
```

Category: Logic Error

Now classify the following coding query:

Query: "Explain the difference between list and tuple in Python."  
Return only the category name.

AI Output: Conceptual Question

-----  
Prompt sent to AI with one example:

Example:  
Query: "I want to cancel my Python function."  
Category: Logic Error

Now classify the following coding query:

Query: "How can I optimize my recursive Fibonacci function?"  
Return only the category name.

AI Output: Optimization

-----

PS C:\Users\Jashwanth\AI coding>

OBSERVATION:

Uses a single example to guide the AI's reasoning, improving accuracy over zero-shot and helping with mildly ambiguous queries. Effectiveness depends on how well the example matches the new query.

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

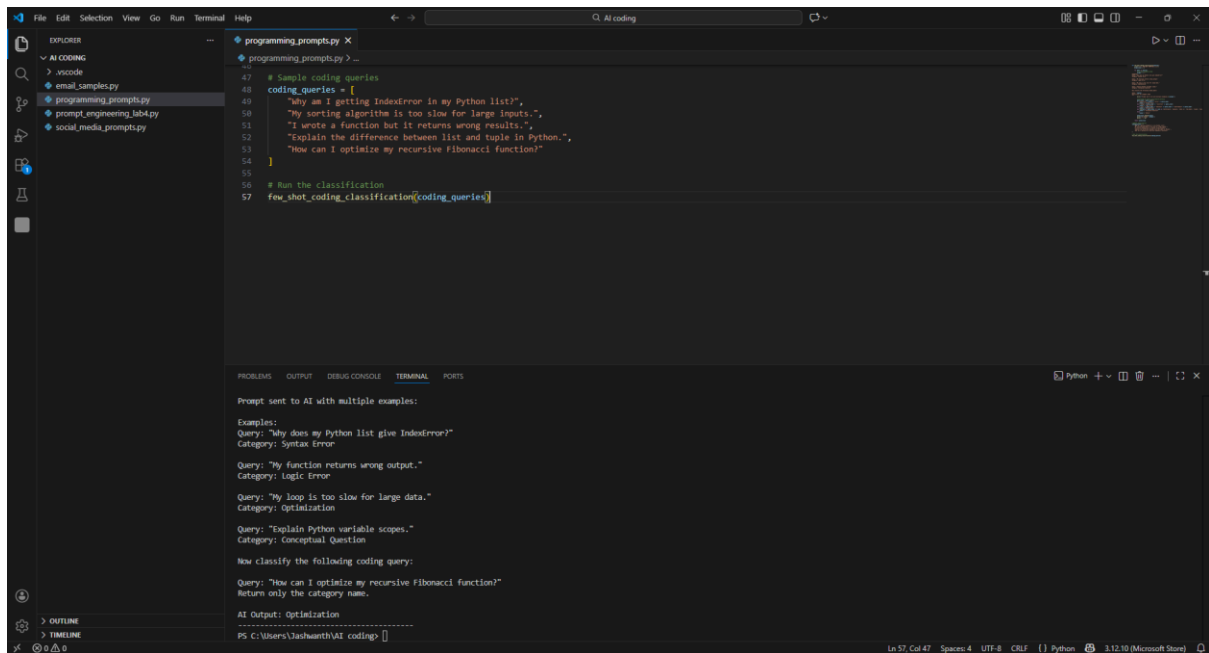
Query: "Explain Python variable scopes."

Category: Conceptual Question

Now classify the following coding query:

Query: "<coding\_query>"

CODE:



```
47 # Sample coding queries
48 coding_queries = [
49     "Why am I getting IndexError in my Python list?",
50     "My sorting algorithm is too slow for large inputs.",
51     "I wrote a function but it returns wrong results.",
52     "Explain the difference between list and tuple in Python.",
53     "How can I optimize my recursive Fibonacci function?"
54 ]
55
56 # Run the classification
57 few_shot_classification(coding_queries)
```

Prompt sent to AI with multiple examples:

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: "How can I optimize my recursive Fibonacci function?"  
Return only the category name.

AI Output: Optimization

OBSERVATION:

Provides multiple examples to highlight patterns, enabling the AI to generalize well to unseen queries. Achieves the highest accuracy among prompting styles, with longer prompts but the most reliable performance 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:

The screenshot shows a VS Code editor with a Python file named `programming_prompts.py`. The script defines a function `zero_shot_social_classification(posts)` that takes a list of social media posts and returns a list of predicted categories. The function uses a simple list of prompts and a corresponding list of categories to make predictions. The script is executed, and the output is displayed in the terminal window.

```

1 def zero_shot_social_classification(posts):
2     predictions = []
3     for post in posts:
4         prompt = f"Classify the following social media post into one of the categories: Promotion, Complaint, Appreciation, Inquiry. Return only the category name."
5         output = ai(prompt)
6         predictions.append(output)
7     return predictions
8
9 # Sample social media posts
10 social_posts = [
11     "Thanks for the quick customer support!",
12     "My order hasn't arrived yet.",
13     "Check out our discount offer!",
14     "How can I reset my password?",
15     "Absolutely love the new update!"
16 ]
17
18 # Run the classification
19 zero_shot_social_classification(social_posts)

```

The terminal output shows the results of the classification:

```

Return only the category name.
AI Output: Promotion
-----
Prompt sent to AI:
Classify the following social media post into one of the categories:
Promotion, Complaint, Appreciation, Inquiry.
Post: "Thanks for the quick customer support?"
Return only the category name.
AI Output: Inquiry
-----
Prompt sent to AI:
Classify the following social media post into one of the categories:
Promotion, Complaint, Appreciation, Inquiry.
Post: "My order hasn't arrived yet."
Return only the category name.
AI Output: Complaint
-----
Prompt sent to AI:
Classify the following social media post into one of the categories:
Promotion, Complaint, Appreciation, Inquiry.
Post: "Check out our discount offer!"
Return only the category name.
AI Output: Promotion
-----
Prompt sent to AI:
Classify the following social media post into one of the categories:
Promotion, Complaint, Appreciation, Inquiry.
Post: "How can I reset my password?"
Return only the category name.
AI Output: Inquiry
-----
Prompt sent to AI:
Classify the following social media post into one of the categories:
Promotion, Complaint, Appreciation, Inquiry.
Post: "Absolutely love the new update!"
Return only the category name.
AI Output: Appreciation

```

OBSERVATION:

Relies solely on instructions without examples. Performs well with clear keywords but struggles with informal, slang, or sarcastic language. Quick and straightforward, though accuracy drops for ambiguous posts.

one-shot

PROMPT:

Example:

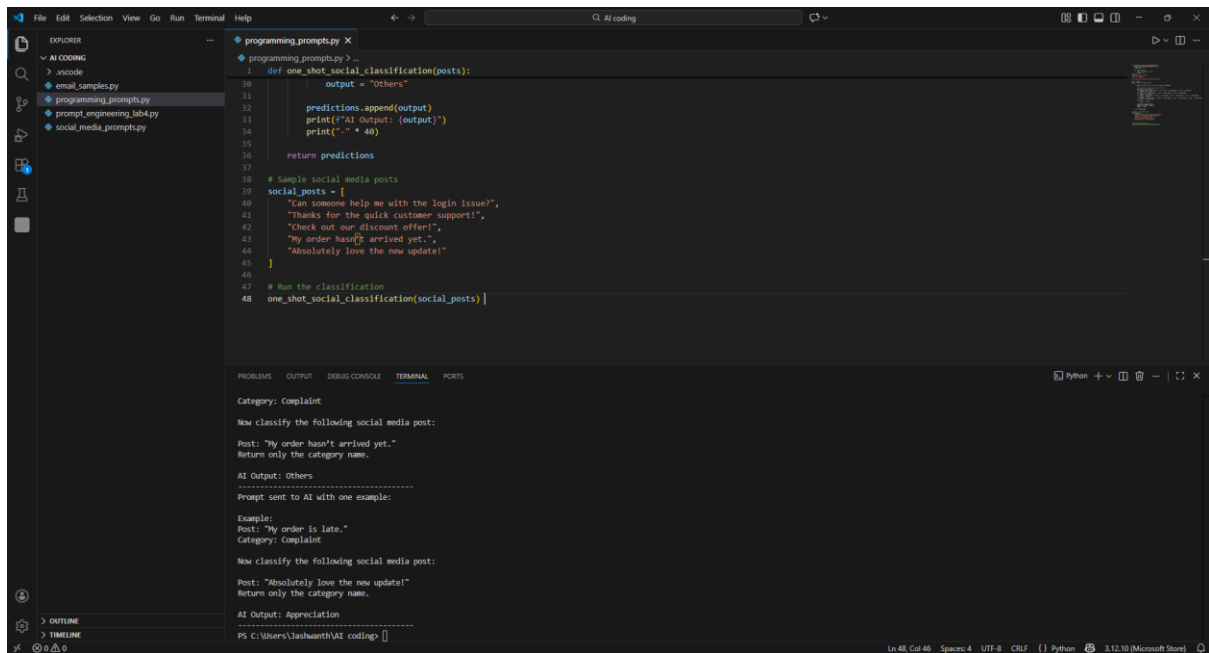
Post: "My order is late."

Category: Complaint

Now classify the following social media post:

Post: "&lt;social\_post&gt;"

CODE:



```
def one_shot_social_classification(posts):
    output = "Others"
    predictions.append(output)
    print(f"AI Output: {output}")
    print("-" * 40)
    return predictions

# Sample social media posts
social_posts = [
    "Can someone help me with the login issue?",
    "Thanks for the quick customer support!",
    "Check out our discount offer!",
    "My order hasn't arrived yet.",
    "Absolutely love the new update!"
]

# Run the classification
one_shot_social_classification(social_posts)
```

Category: Complaint

Now classify the following social media post:

Post: "My order hasn't arrived yet."

Return only the category name.

AI Output: Others

-----

Prompt sent to AI with one example:

Example:

Post: "My order is late."

Category: Complaint

Now classify the following social media post:

Post: "Absolutely love the new update!"

Return only the category name.

AI Output: Appreciation

-----

OBSERVATION:

Uses a single example to guide AI reasoning, improving accuracy and handling some informal expressions more effectively. The quality of results depends on how well the chosen example reflects informal language in new posts.

FEW-SHOT:

PROMPT:

Examples:

Post: "Loved the new feature!" → Appreciation

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

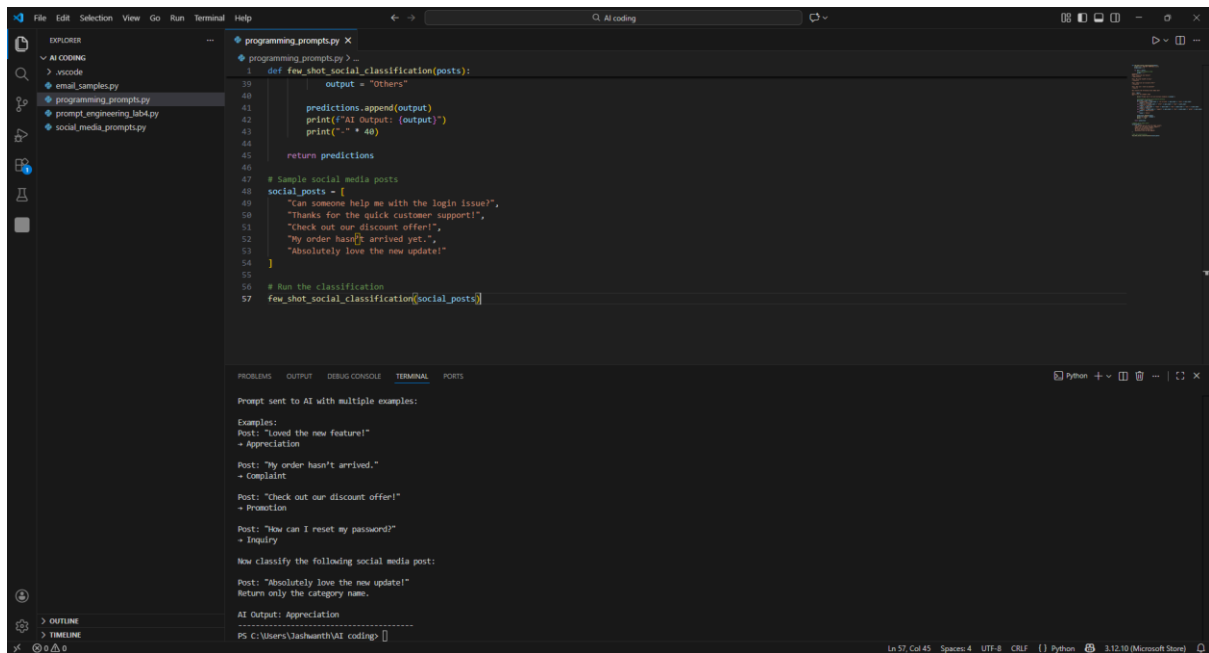
Post: "Check out our discount offer!" → Promotion

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

Now classify the following social media post:

Post: "<social\_post>"

CODE:



```
1 def few_shot_social_classification(posts):
2     output = "Others"
3
4     predictions.append(output)
5     print(f"AI Output: {output}")
6     print("-" * 40)
7
8     return predictions
9
10 # Sample social media posts
11 social_posts = [
12     "Can someone help me with the login issue?",
13     "Thanks for the quick customer support!",
14     "Check out our discount offer!",
15     "My order hasn't arrived yet.",
16     "Absolutely love the new update!"
17 ]
18
19 # Run the classification
20 few_shot_social_classification(social_posts)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Prompt sent to AI with multiple examples:

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: "Absolutely love the new update!"  
Return only the category name.

AI Output: Appreciation

PS C:\Users\Jashwanth\AI coding> |

OBSERVATION:

Uses multiple examples to illustrate patterns, enabling the AI to generalize more effectively. Delivers the highest accuracy, handling informal, slang, and mixed-language posts better. Prompts are longer, but this approach is the most reliable for social media classification.