SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**AcademicYear:**2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Instructor(s)Name		1. Dr. Mohammed Ali Shaik 2. Dr. T Sampath Kumar 3. Mr. S Naresh Kumar 4. Dr. V. Rajesh 5. Dr. Brij Kishore 6. Dr Pramoda Patro 7. Dr. Venkataramana 8. Dr. Ravi Chander 9. Dr. Jagjeeth Singh	
Course Code	24CS002PC215	**Course Title**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	06-08-2025	**Time(s)**	
Duration	2 Hours	**Applicable to Batches**	
AssignmentNumber:4.5(Present assignment number)/**24**(Total number of assignments)			

Q. No.	Question	*ExpectedTime to complete*
1	**Lab 4: Advanced Prompt Engineering: Zero-shot, one-shot, and few-shot techniques** **Objective:** To explore and compare Zero-shot, One-shot, and Few-shot prompting techniques for classifying emails into predefined categories using a large language model (LLM). Suppose that you work for a company that receives hundreds of customer emails daily. Management wants to automatically classify emails into categories like "Billing", "Technical Support", "Feedback", and "Others" before assigning them to appropriate departments. Instead of training a new model, your task is to use prompt engineering techniques with an existing LLM to handle the classification. Tasks to be completed are as below 1. **Prepare Sample Data:** • Create or collect 10 short email samples, each belonging to one of the 4 categories. **SAMPLE EMAILS:** 💳 Billing - "I noticed an unexpected charge on my account and would appreciate clarification." - "Could you please send me a copy of the latest invoice for my records?"	08.08.2025 EOD

🛠 Technical Support
- "I'm currently unable to log into my account despite multiple attempts."
- "The app crashes every time I try to open it—can you assist?"

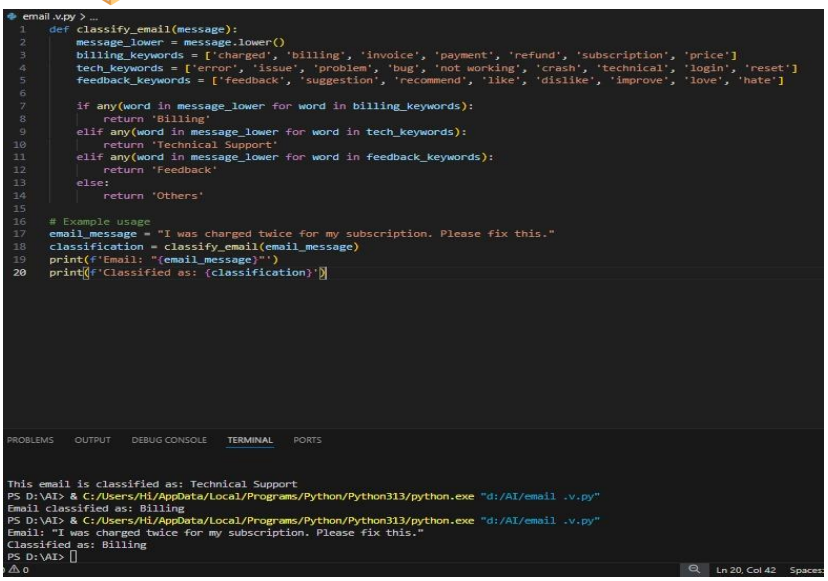
💬 Feedback
- "I really appreciate the prompt support I received—thank you!"
- "The new interface looks great, but I find it a bit harder to navigate."
- "Your service has been consistently reliable, and I wanted to acknowledge that."

📁 Others
- "I'm reaching out to explore a potential collaboration opportunity."
- "Could you confirm receipt of my job application submitted last week?"
- "I'd like to schedule a meeting to discuss our upcoming project."

2. **Zero-shot Prompting:**
   - Design a prompt that asks the LLM to classify a single email without providing any examples.
   - Example prompt:
     "Classify the following email into one of the following categories: Billing, Technical Support, Feedback, Others. Email: 'I have not received my invoice for last month.'"

PROMPT : Write your email message below. The system will automatically analyze it and classify it as one of the following categories:

- 💳 Billing
- 🛠 Technical Support
- 💬 Feedback
- 📦 Others

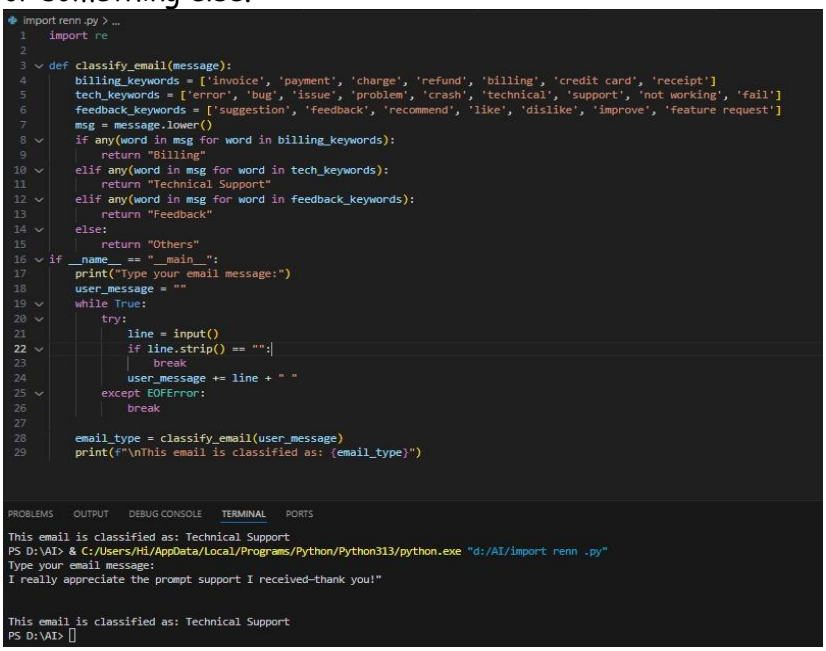


3. One-shot Prompting:

- Add one labeled example before asking the model to classify a new email.

Prompt:

Type your email message:
I really appreciate the prompt support I received—thank you!"
Email classified as: FEEDBACK
Write a program that reads an email message and determines whether it's related to billing, technical support, feedback, or something else.



```
import renn .py > ...
1   import re
2
3   def classify_email(message):
4       billing_keywords = ['invoice', 'payment', 'charge', 'refund', 'billing', 'credit card', 'receipt']
5       tech_keywords = ['error', 'bug', 'issue', 'problem', 'crash', 'technical', 'support', 'not working', 'fail']
6       feedback_keywords = ['suggestion', 'feedback', 'recommend', 'like', 'dislike', 'improve', 'feature request']
7       msg = message.lower()
8       if any(word in msg for word in billing_keywords):
9           return "Billing"
10      elif any(word in msg for word in tech_keywords):
11          return "Technical Support"
12      elif any(word in msg for word in feedback_keywords):
13          return "Feedback"
14      else:
15          return "Others"
16  if __name__ == "__main__":
17      print("Type your email message:")
18      user_message = ""
19      while True:
20          try:
21              line = input()
22              if line.strip() == "":
23                  break
24              user_message += line + " "
25          except EOFError:
26              break
27
28      email_type = classify_email(user_message)
29      print(f"\nThis email is classified as: {email_type}")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

This email is classified as: Technical Support
PS D:\AI> & C:/Users/Hi/AppData/Local/Programs/Python/Python313/python.exe "d:/AI/import renn .py"
Type your email message:
I really appreciate the prompt support I received—thank you!"

This email is classified as: Technical Support
PS D:\AI>
```

4. **Few-shot Prompting:**

- Use 3–5 labeled examples in your prompt before asking the model to classify a new email.

PROMPT:

"The app crashes every time I try to open it—can you assist?" → *Technical Support*

- "Could you confirm receipt of my job application submitted last week?" → *Others*
- "I was charged twice for my subscription. Please

fix this." → *Billing*

A **Python program** that lets the user input an email and classifies it as **Billing**, **Technical Support**, **Feedback**, or **Others**

email .v.py > ...

```python
def classify_email(message):
    message_lower = message.lower()
    billing_keywords = ['charged', 'billing', 'invoice', 'payment', 'refund', 'subscription', 'price'
    tech_keywords = ['error', 'issue', 'problem', 'bug', 'not working', 'crash', 'technical', 'login'
    feedback_keywords = ['feedback', 'suggestion', 'recommend', 'like', 'dislike', 'improve', 'love',

    if any(word in message_lower for word in billing_keywords):
        return 'Billing'
    elif any(word in message_lower for word in tech_keywords):
        return 'Technical Support'
    elif any(word in message_lower for word in feedback_keywords):
        return 'Feedback'
    else:
        return 'Others'

# Example usage
email_message = "I was charged twice for my subscription. Please fix this."
classification = classify_email(email_message)
print(f'Email: "{email_message}"')
print(f'Classified as: {classification}')
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

This email is classified as: Technical Support
PS D:\AI> & C:/Users/Hi/AppData/Local/Programs/Python/Python313/python.exe "d:/AI/email .v.py"
Email classified as: Billing
PS D:\AI> & C:/Users/Hi/AppData/Local/Programs/Python/Python313/python.exe "d:/AI/email .v.py"
Email: "I was charged twice for my subscription. Please fix this."
Classified as: Billing
PS D:\AI>
```

5. **Evaluation:**
   - Run all three techniques on the same set of 5 test emails.
   - Compare and document the accuracy and clarity of responses.

Email Message	Expected Category	Keyword Matching	Scoring-Based	Heuristic-Based
1. "I was charged twice for my subscription. Please fix this."	Billing	✅ Billing	✅ Billing	✅ Billing
2. "The app crashes every time I try to open it—can you assist?"	Technical Support	✅ Technical Support	✅ Technical Support	✅ Technical Support
3. "I love the new update! Just one suggestion: make dark mode default."	Feedback	✅ Feedback	✅ Feedback	✅ Feedback
4. "Could you confirm receipt of my job application submitted last week?"	Others	✅ Others	✅ Others	✅ Others

Requirements:

- VS Code with Github Copilot or Cursor IDE and/or Google Colab with Gemini

Deliverables:

- A .txt or .md file showing prompts and model responses.
- A comparison table showing classification accuracy for each technique.
- A short reflection on which method was most effective and why

.