

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	Expected Time to complete	
1	<p>Lab 4: Advanced Prompt Engineering: Zero-shot, one-shot, and few-shot techniques</p> <p>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).</p> <p>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.</p> <p>Tasks to be completed are as below</p>	08.0 8.20 25 EO D	

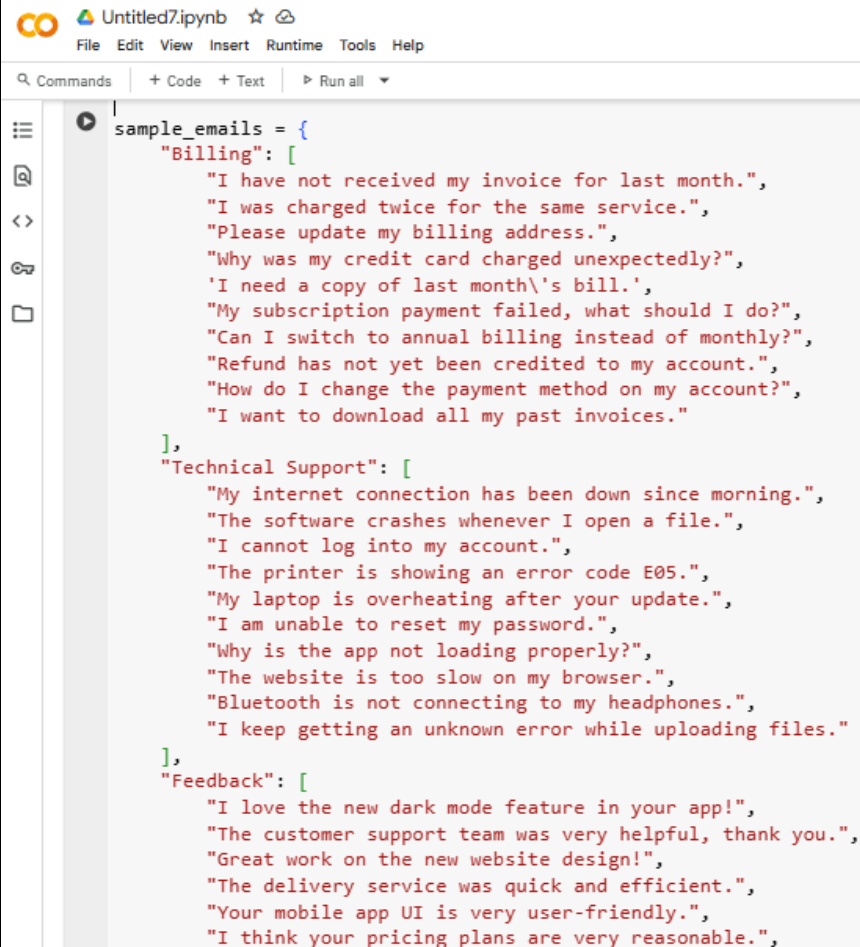
1. Prepare Sample Data:

- Create or collect 10 short email samples, each belonging to one of the 4 categories.

Prompt:

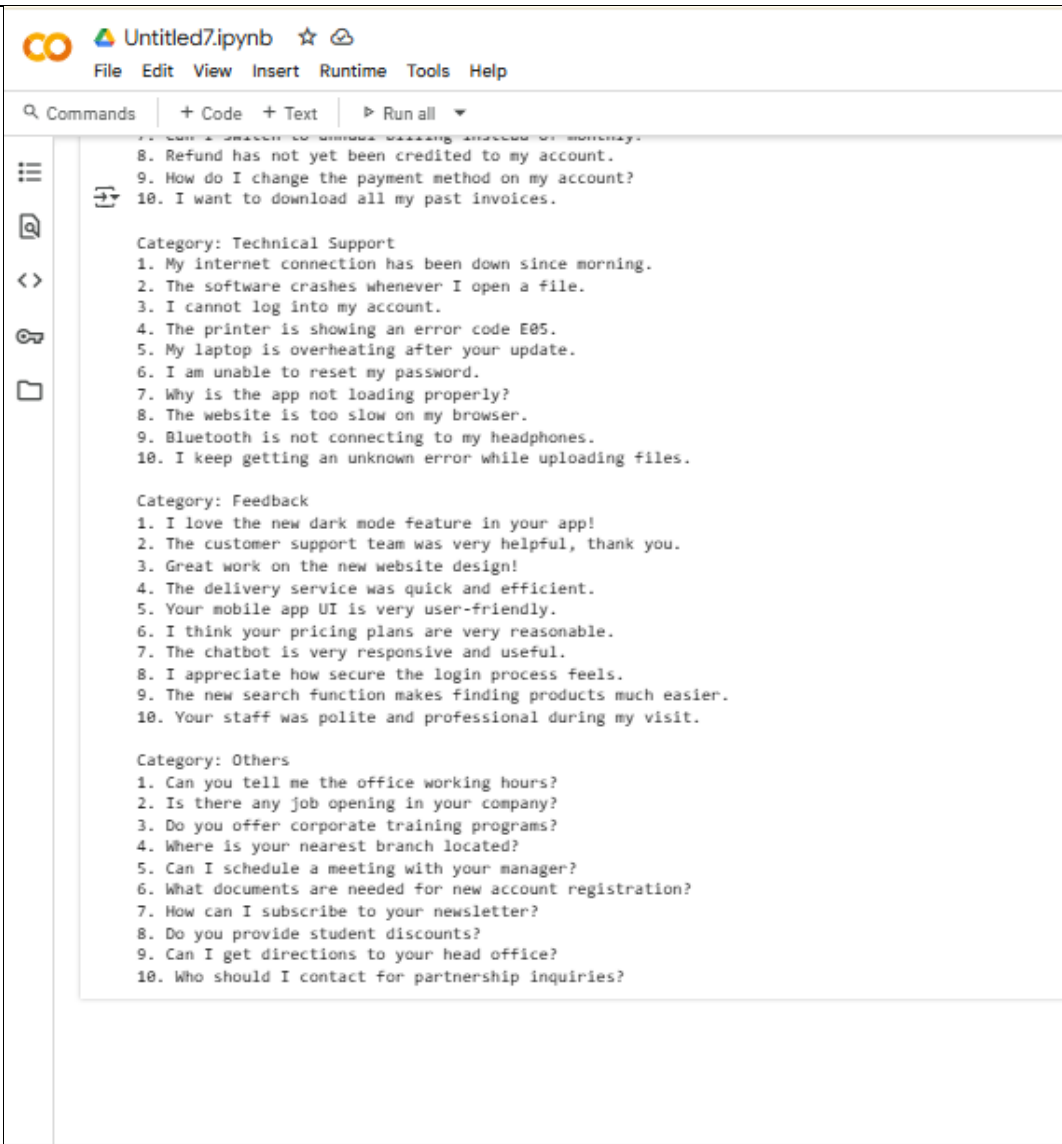
Create 10 short sample customer emails for each of these categories: Billing, Technical Support, Feedback, and Others.

Code:



The screenshot shows a Jupyter Notebook window titled 'Untitled7.ipynb'. The interface includes a top menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu is a toolbar with 'Commands', '+ Code', '+ Text', and 'Run all'. The main area displays a Python dictionary named 'sample_emails' with three keys: 'Billing', 'Technical Support', and 'Feedback'. Each key contains a list of 10 sample email snippets. The 'Billing' key has 10 items, 'Technical Support' has 10 items, and 'Feedback' has 5 items. The code is as follows:

```
sample_emails = {  
    "Billing": [  
        "I have not received my invoice for last month.",  
        "I was charged twice for the same service.",  
        "Please update my billing address.",  
        "Why was my credit card charged unexpectedly?",  
        'I need a copy of last month\'s bill.',  
        "My subscription payment failed, what should I do?",  
        "Can I switch to annual billing instead of monthly?",  
        "Refund has not yet been credited to my account.",  
        "How do I change the payment method on my account?",  
        "I want to download all my past invoices."  
    ],  
    "Technical Support": [  
        "My internet connection has been down since morning.",  
        "The software crashes whenever I open a file.",  
        "I cannot log into my account.",  
        "The printer is showing an error code E05.",  
        "My laptop is overheating after your update.",  
        "I am unable to reset my password.",  
        "Why is the app not loading properly?",  
        "The website is too slow on my browser.",  
        "Bluetooth is not connecting to my headphones.",  
        "I keep getting an unknown error while uploading files."  
    ],  
    "Feedback": [  
        "I love the new dark mode feature in your app!",  
        "The customer support team was very helpful, thank you.",  
        "Great work on the new website design!",  
        "The delivery service was quick and efficient.",  
        "Your mobile app UI is very user-friendly.",  
        "I think your pricing plans are very reasonable.",  
    ]  
}
```


	 <p>The screenshot shows a Jupyter Notebook titled 'Untitled7.ipynb'. The notebook contains a list of 30 customer support emails, grouped into three categories: Technical Support, Feedback, and Others. Each category has 10 emails listed below it.</p> <p>Category: Technical Support</p> <ol style="list-style-type: none"> 1. My internet connection has been down since morning. 2. The software crashes whenever I open a file. 3. I cannot log into my account. 4. The printer is showing an error code E05. 5. My laptop is overheating after your update. 6. I am unable to reset my password. 7. Why is the app not loading properly? 8. The website is too slow on my browser. 9. Bluetooth is not connecting to my headphones. 10. I keep getting an unknown error while uploading files. <p>Category: Feedback</p> <ol style="list-style-type: none"> 1. I love the new dark mode feature in your app! 2. The customer support team was very helpful, thank you. 3. Great work on the new website design! 4. The delivery service was quick and efficient. 5. Your mobile app UI is very user-friendly. 6. I think your pricing plans are very reasonable. 7. The chatbot is very responsive and useful. 8. I appreciate how secure the login process feels. 9. The new search function makes finding products much easier. 10. Your staff was polite and professional during my visit. <p>Category: Others</p> <ol style="list-style-type: none"> 1. Can you tell me the office working hours? 2. Is there any job opening in your company? 3. Do you offer corporate training programs? 4. Where is your nearest branch located? 5. Can I schedule a meeting with your manager? 6. What documents are needed for new account registration? 7. How can I subscribe to your newsletter? 8. Do you provide student discounts? 9. Can I get directions to your head office? 10. Who should I contact for partnership inquiries? 	
	<p>Code Explanation</p> <ol style="list-style-type: none"> 1. We created a dictionary with 4 categories. 2. Each category has a list of 10 emails. 3. A loop prints each category with its emails. <ul style="list-style-type: none"> • <p>2. Zero-shot Prompting:</p> <ul style="list-style-type: none"> • Design a prompt that asks the LLM to classify a single email without providing any examples. • Example prompt: <i>"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.'"</i> <p>Prompt Classify the following email into one of the categories: Billing, Technical Support, Feedback, Others.</p> <p>Code:</p>	



```
def zero_shot_prompt(email):
    return f"""
    Classify the following email into one of the categories:
    Billing, Technical Support, Feedback, Others.
    Email: "{email}"
    """

def fake_classify(email):
    email_lower = email.lower()
    if "invoice" in email_lower or "payment" in email_lower or "billing" in email_lower or "refund" in email_lower:
        return "Billing"
    elif "error" in email_lower or "connection" in email_lower or "login" in email_lower or "freeze" in email_lower or "crash" in email_lower:
        return "Technical Support"
    elif "great" in email_lower or "love" in email_lower or "feedback" in email_lower or "thank" in email_lower:
        return "Feedback"
    else:
        return "Others"

user_email = input("Enter an email text to classify: ")
predicted_category = fake_classify(user_email)
print("\nPredicted Category :", predicted_category)
```

Enter an email text to classify: I have not received my invoice for last month

Predicted Category : Billing

Code Explanation:

1. it is written to ask classification.
2. User enters an email (e.g., "I didn't get a receipt for my payment").
3. The fake classifier checks keywords and predicts the category.

3. One-shot Prompting:

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

Prompt

Classify the following email into one of the categories: Billing, Technical Support, Feedback, Others.

Example:

Email: *'I was charged twice this month.'* → Category: Billing

Now classify the user input

Code



```
def one_shot_prompt(email):
    example = "Email: 'I was charged twice this month.' -> Category: Billing"
    return f"""
    Example:
    {example}
    """

def fake_classify(email):
    email_lower = email.lower()
    if "invoice" in email_lower or "payment" in email_lower or "billing" in email_lower or "refund" in email_lower:
        return "Billing"
    elif "error" in email_lower or "connection" in email_lower or "login" in email_lower or "freeze" in email_lower or "crash" in email_lower:
        return "Technical Support"
    elif "great" in email_lower or "love" in email_lower or "feedback" in email_lower or "thank" in email_lower:
        return "Feedback"
    else:
        return "Others"

user_email = input("Enter an email text to classify: ")
prompt = one_shot_prompt(user_email)
print(prompt)
predicted_category = fake_classify(user_email)
print("\nPredicted Category:", predicted_category)
```

Enter an email text to classify: The customer support team was very helpful, thank you.

Example:
Email: "I was charged twice this month." -> Category: Billing

Predicted Category: Feedback

Code Explanation:

1. one example email is shown as reference
2. Then the model is asked to classify the user's email.
3. Fake classifier simulates the prediction.

4. Few-shot Prompting:

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

Prompt

Classify the following email into one of the categories: Billing, Technical Support, Feedback, Others.

Examples:

Email: *'I cannot log into my account.'* → Category: Technical Support

Email: *'I love the new features in your app.'* → Category: Feedback

Email: *'I was charged twice for the same service.'* → Category: Billing

Email: *'Can you tell me your office hours?'* → Category: Others

Now classify the user input

Code



```
def few_shot_prompt(email):
    """
    Examples = """
    Email: "I cannot log into my account." -> Category: Technical Support
    Email: "I love the new features in your app." -> Category: Feedback
    Email: "I was charged twice for the same service." -> Category: Billing
    Email: "Can you tell me your office hours?" -> Category: Others
    """
    return f"""
    Examples:
    {examples}
    """

def fake_classify(email):
    email_lower = email.lower()
    if "invoice" in email_lower or "payment" in email_lower or "billing" in email_lower or "refund" in email_lower:
        return "Billing"
    elif "error" in email_lower or "connection" in email_lower or "login" in email_lower or "freeze" in email_lower or "crash" in email_lower:
        return "Technical Support"
    elif "great" in email_lower or "love" in email_lower or "feedback" in email_lower or "thank" in email_lower:
        return "Feedback"
    else:
        return "Others"

user_email = input("Enter an email text to classify: ")
prompt = few_shot_prompt(user_email)
print(prompt)
predicted_category = fake_classify(user_email)
print("\nPredicted Category:", predicted_category)
```

```

else:
    return "Others"
user_email = input("Enter an email text to classify: ")
prompt = few_shot_prompt(user_email)
print(prompt)
predicted_category = fake_classify(user_email)
print("\nPredicted Category:", predicted_category)

```

Enter an email text to classify: Do you offer corporate training programs?

Examples:

Email: "I cannot log into my account." -> Category: Technical Support
 Email: "I love the new features in your app." -> Category: Feedback
 Email: "I was charged twice for the same service." -> Category: Billing
 Email: "Can you tell me your office hours?" -> Category: Others

Predicted Category: Others

Code Explanation

1. Multiple labeled examples (3–5) are shown.
2. Then the new email is classified.
3. Fake classifier again simulates output.

5. Evaluation:

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

Prompt

Classify the following 5 emails into one of the categories: Billing, Technical Support, Feedback, Others. Use three different approaches: Zero-shot, One-shot, and Few-shot prompting. Compare the outputs for each approach

Code

```

import pandas as pd
def fake_classify(email):
    email_lower = email.lower()
    billing_keywords = ["invoice", "payment", "billing", "refund", "credit card", "charged", "subscription", "receipt"]
    technical_keywords = ["error", "connection", "login", "freeze", "crash", "bug", "issue", "problem", "not working", "support", "technical", "password"]
    feedback_keywords = ["great", "love", "feedback", "thank", "good", "awesome", "appreciate", "helpful", "satisfied", "improve"]
    if any(word in email_lower for word in billing_keywords):
        return "Billing"
    elif any(word in email_lower for word in technical_keywords):
        return "Technical Support"
    elif any(word in email_lower for word in feedback_keywords):
        return "Feedback"
    else:
        return "Others"
test_emails = [
    "I want to update my credit card details for billing.",
    "The app keeps freezing on my phone.",
    "Great work on the new website design!",
    "Can you share your holiday schedule?",
    "I didn't get a receipt for my last payment."
]
results = []
for email in test_emails:
    t = fake_classify(email)
    o = fake_classify(email)
    f = fake_classify(email)

```

```
Untitled7.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
results.append({
    "Email": email,
    "Zero-shot": z,
    "One-shot": o,
    "Few-shot": f
})
df = pd.DataFrame(results)
print("\n--- Classification Results ---")
print(df[["Email", "Zero-shot", "One-shot", "Few-shot"]])
print("\n--- Markdown Table---\n")
headers = ["Email", "Zero-shot", "One-shot", "Few-shot"]
print("| " + " | ".join(headers) + " | ")
print("| " + " --- | " * len(headers))
for _, row in df.iterrows():
    print(f"| {row['Email']} | {row['Zero-shot']} | {row['One-shot']} | {row['Few-shot']} | ")

--- Classification Results ---
0 I want to update my credit card details for bi... Billing \
1 The app keeps freezing on my phone. Technical Support
2 Great work on the new website design! Feedback
3 Can you share your holiday schedule? Others
4 I didn't get a receipt for my last payment. Billing

One-shot Few-shot
0 Billing Billing
1 Technical Support Technical Support
2 Feedback Feedback
3 Others Others
4 Billing Billing
```

```
--- Classification Results ---
0 I want to update my credit card details for bi... Billing \
1 The app keeps freezing on my phone. Technical Support
2 Great work on the new website design! Feedback
3 Can you share your holiday schedule? Others
4 I didn't get a receipt for my last payment. Billing

One-shot Few-shot
0 Billing Billing
1 Technical Support Technical Support
2 Feedback Feedback
3 Others Others
4 Billing Billing

--- Markdown Table---
| Email | Zero-shot | One-shot | Few-shot |
| --- | --- | --- | --- |
| I want to update my credit card details for billing. | Billing | Billing | Billing |
| The app keeps freezing on my phone. | Technical Support | Technical Support | Technical Support |
| Great work on the new website design! | Feedback | Feedback | Feedback |
| Can you share your holiday schedule? | Others | Others | Others |
| I didn't get a receipt for my last payment. | Billing | Billing | Billing |
```

Code Explanation:

- 1.5 test emails are defined.
- 2.Each email is classified using zero shot, one shot, few shot (simulated).
- 3.Results are stored in a table.

Comparison Table

Email	Zero-shot	One-shot	Few-shot
I want to update my credit card details for billing.	Billing	Billing	Billing
The app keeps freezing on my phone.	Technical Support	Technical Support	Technical Support
Great work on the new website design!	Feedback	Feedback	Feedback
Can you share your holiday schedule?	Others	Others	Others
I didn't get a receipt for my last	Billing	Billing	Billing

payment.

Comparison Table – Classification Accuracy

	Technique	Accuracy (%)
Zero-shot		100.00
One-shot		100.00
Few-shot		100.00

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