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. 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*	08.08.2025 EOD

for last month.'"

3. One-shot Prompting:

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

4. **Few-shot Prompting:**

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

5. **Evaluation:**

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

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

CODE :-

```
# Sample Emails Dataset (10 total)
emails = [
    {"email": "I have not received my invoice for last month.", "label": "Billing"},
    {"email": "There is an extra charge on my account.", "label": "Billing"},
    {"email": "The app crashes when I open it.", "label": "Technical Support"},
    {"email": "I can't reset my password.", "label": "Technical Support"},
    {"email": "I love the new features in your update!", "label": "Feedback"},
    {"email": "The UI looks much better now.", "label": "Feedback"},
    {"email": "Do you offer student discounts?", "label": "Others"},
    {"email": "Are your services available in the UK?", "label": "Others"},
    {"email": "App keeps freezing after update.", "label": "Technical Support"},
    {"email": "Thanks for resolving my issue quickly!", "label": "Feedback"},
]

# Select 5 for testing
test_emails = emails[5:]

# Simulated predictions (using keyword rules)
def zero_shot(email):
    if "invoice" in email or "charge" in email:
        return "Billing"
    elif "crash" in email or "reset" in email or "freeze" in email:
        return "Technical Support"
    elif "love" in email or "thanks" in email or "better" in email:
        return "Feedback"
    else:
        return "Others"

def one_shot(email):
    if any(k in email for k in ["reset", "freeze", "can't", "crash"]):
        return "Technical Support"
    elif "invoice" in email or "charge" in email:
        return "Billing"
    elif "thanks" in email or "update" in email:
        return "Feedback"
    else:
        return "Others"

def few_shot(email):
    if any(k in email for k in ["invoice", "charge"]):
        return "Billing"
    elif any(k in email for k in ["freeze", "reset", "crash", "can't"]):
        return "Technical Support"
    elif any(k in email for k in ["love", "thanks", "better", "update"]):
        return "Feedback"
    else:
        return "Others"
```

```python
# Evaluate and collect results
results = []
for sample in test_emails:
    e = sample["email"]
    expected = sample["label"]
    results.append({
        "email": e,
        "expected": expected,
        "zero": zero_shot(e),
        "one": one_shot(e),
        "few": few_shot(e)
    })

# Print results and accuracy summary
correct_zero = sum(1 for r in results if r["expected"] == r["zero"])
correct_one = sum(1 for r in results if r["expected"] == r["one"])
correct_few = sum(1 for r in results if r["expected"] == r["few"])
total = len(results)

print("\n--- Prompting Results ---")
for r in results:
    print(f"\nEmail: {r['email']}")
    print(f"Expected: {r['expected']}")
    print(f"Zero-shot: {r['zero']}")
    print(f"One-shot: {r['one']}")
    print(f"Few-shot: {r['few']}")

print("\n--- Accuracy Comparison ---")
print(f"Zero-shot Accuracy: {correct_zero}/{total} ({correct_zero/total:.0%})")
print(f"One-shot Accuracy : {correct_one}/{total} ({correct_one/total:.0%})")
print(f"Few-shot Accuracy : {correct_few}/{total} ({correct_few/total:.0%})")
```

OUTPUT :-

```
PS C:\AI CODEING>  & 'c:\Users\kbhuv\AppData\Local\Programs\Pyth
  '65116' '--' 'c:\AI CODEING\lab ass 1.1.py'

 --- Prompting Results ---

 Email: The UI looks much better now.
 Expected: Feedback
 Zero-shot: Feedback
 One-shot: Others
 Few-shot: Feedback

Email: Do you offer student discounts?
Expected: Others
Zero-shot: Others
One-shot: Others
Few-shot: Others

Email: Are your services available in the UK?
Expected: Others
Zero-shot: Others
One-shot: Others
```

.