SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name	Dr. Rishabh Mittal		
Instructor(s) Name	Mr. S Naresh Kumar Ms. B. Swathi Dr. Sasanko Shekhar Gantayat Mr. Md Sallauddin Dr. Mathivanan Mr. Y Srikanth Ms. N Shilpa Dr. Rishabh Mittal (Coordinator) Dr. R. Prashant Kumar Mr. Ankushavali MD Mr. B Viswanath Ms. Sujitha Reddy Ms. A. Anitha Ms. M.Madhuri Ms. Katherashala Swetha Ms. Velpula sumalatha Mr. Bingi Raju		
CourseCode	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week2 – Monday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Duration	2 Hours	**Applicable to Batches**	All batches
Assignment Number: 4.1(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*Expected Time to complete*
1	**Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques**	Week2 - Monday

Lab Objectives:

- To explore and apply different levels of prompt examples in AI-assisted code generation.
- To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality.
- To evaluate the impact of context richness and example quantity on AI performance.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Use zero-shot prompting to instruct AI with minimal context.
- Use one-shot prompting with a single example to guide AI code generation.
- Apply few-shot prompting using multiple examples to improve AI responses.
- Compare AI outputs across the three prompting strategies.

Analyze the sample example problem and complete the given problem statement 1,2

Advanced Prompt Engineering – Topic Classification of News Headlines

Sample Example Problem:

Problem Statement 0:

A news aggregation platform wants to automatically categorize headlines into Politics, Sports, Technology, and Entertainment without training a machine learning model.

Tasks to be Completed

1. Prepare Sample Data

   Collect 10 news headlines, each belonging to one of the four categories.

2. Zero-shot Prompting

   Write a prompt asking the LLM to classify a headline into a category without examples.

3. One-shot Prompting

   Add one labeled headline example before classifying a new headline.

4. Few-shot Prompting

   Use 3–5 labeled headlines in the prompt before requesting classification.

5. Evaluation

   Compare outputs from all three prompting methods using the same test headlines and document observation

Sample Solution for problem statement 0:

1. Sample News Headlines

No.	News Headline	Categ
H1	Government announces new education policy	Politics
H2	Parliament passes new tax reform bill	Politics
H3	India wins the T20 cricket series	Sports
H4	Football club signs a new international player	Sports
H5	Tech company launches a new AI-powered smartphone	Technolo
H6	Cybersecurity firm reports major data breach	Technolo
H7	Upcoming movie breaks box office records	Entertair
H8	Popular actor announces next film project	Entertair

2. Zero-shot Prompting

Prompt Used:

Classify the following news headline into one of these categories:

Politics, Sports, Technology, Entertainment.

Headline: “India wins the T20 cricket series.”

Output:

Sports

Observation:

	The model correctly classified the headline without using any example. --- **3. One-shot Prompting** Prompt Used: Example: Headline: “Government announces new education policy” Category: Politics Now classify the following headline into Politics, Sports, Technology, or Entertainment. Headline: “Tech company launches a new AI-powered smartphone.” **Output:** Technology **Observation:** Providing one example improved clarity and consistency in classification. --- **4. Few-shot Prompting** Prompt Used: Example 1: Headline: “Parliament passes new tax reform bill” Category: Politics Example 2: Headline: “Football club signs a new international player” Category: Sports Example 3: Headline: “Cybersecurity firm reports major data breach” Category: Technology	

	Example 4: Headline: “Upcoming movie breaks box office records” Category: Entertainment Now classify the following headline into Politics, Sports, Technology, or Entertainment. Headline: “Popular actor announces next film project.” Output: Entertainment Observation: Few-shot prompting produced the most accurate and confident response	
Problem Statement1	**Customer Email Classification** A company receives a large number of customer emails every day and wants to automatically classify them into the following categories: • Billing • Technical Support • Feedback • Others Instead of training a new machine learning model, the company decides to use prompt engineering techniques with an existing large language model. **Tasks** 1. Prepare five short sample emails, each belonging to one of the above categories. 2. Write a zero-shot prompt to classify a given email into one of the categories without providing any examples.	

3. Write a one-shot prompt by including one labeled email example and ask the model to classify a new email.
4. Write a few-shot prompt by including two or three labeled email examples and ask the model to classify a new email.
5. Compare the outputs obtained using zero-shot, one-shot, and few-shot prompting techniques and briefly comment on their effectiveness

Prompt

```
create data of 5 emails belonging to Biling,
Technical Support, Feedback Othes,
write a python program to Classify the emails
into one of the categories
```

Code:

```
# Sample email data
emails = [
    "Subject: Invoice for your recent
purchase\nDear Customer, please find attached
the invoice for your recent purchase. If you
have any questions, feel free to contact our
billing department.",
    "Subject: Technical Support Needed\nHello,
I am experiencing issues with my account login.
Can you please assist me in resolving this
problem?",
    "Subject: Feedback on your service\nI
wanted to provide some feedback regarding the
excellent service I received from your team
last week. Keep up the great work!",
    "Subject: Inquiry about product
features\nCould you provide more information
about the features of your new product? I am
considering making a purchase.",
    "Subject: Complaint about delayed
delivery\nI am writing to express my
dissatisfaction with the delayed delivery of my
order. Please address this issue promptly."
```

```
]
# Function to classify emails
def classify_email(email):
    email_lower = email.lower()
    if "invoice" in email_lower or "billing" in
email_lower:
        return "Billing"
    elif "technical support" in email_lower or
"issues" in email_lower or "assist" in
email_lower:
        return "Technical Support"
    elif "feedback" in email_lower or "service"
in email_lower:
        return "Feedback"
    else:
        return "Others"
# Classify and print the category of each email
for email in emails:
    category = classify_email(email)
    print(f"Email categorized as:
{category}\n")
```

Output

```
Email categorized as: Billing

Email categorized as: Technical Support

Email categorized as: Feedback

Email categorized as: Others

Email categorized as: Others
```

Problem Statement 2

Intent Classification for Chatbot Queries

A company wants to deploy a chatbot to handle customer queries. Each query must be classified into one of the following intents: Account Issue, Order Status, Product Inquiry, or General Question using prompt engineering techniques.

Tasks to be Completed

1. Prepare Sample Data

   Create 6 short chatbot user queries, each mapped to one of

the four intents.

2. Zero-shot Prompting

   Design a prompt that asks the LLM to classify a user query into the given intent categories without examples.

3. One-shot Prompting

   Provide one labeled query in the prompt before classifying a new query.

4. Few-shot Prompting

   Include 3–5 labeled intent examples to guide the LLM before classifying a new query.

5. Evaluation

   Apply all three techniques to the same set of test queries and document differences in performance.

Prompt

```
create a ai chatbot to handle customer queries for
a retail website,
the chatbot bot should be able to respond to
questions about product availability,
order status, Account Issues, product inquiries or
general questions.
```

Code :

```
# Sample customer queries
queries = [
    "Is the new smartphone model available in
stock?",
    "What is the status of my order #12345?",
    "I am having trouble logging into my account.
Can you help?",
    "Can you provide more details about the features
of the laptop?",
    "What are your store hours and return policy?"
]
# Function to classify customer queries
def classify_query(query):
```

```
    query_lower = query.lower()
    if "available" in query_lower or "in stock" in
query_lower:
        return "Product Availability"
    elif "status" in query_lower or "order" in
query_lower:
        return "Order Status"
    elif "trouble logging in" in query_lower or
"account" in query_lower:
        return "Account Issues"
    elif "details" in query_lower or "features" in
query_lower:
        return "Product Inquiries"
    else:
        return "General Questions"
# Classify and print the category of each query
for query in queries:
    category = classify_query(query)
    print(f"Query categorized as: {category}\n")
```

Output :

```
Query categorized as: Product Availability

Query categorized as: Order Status

Query categorized as: Account Issues

Query categorized as: Product Inquiries

Query categorized as: General Questions
```

Problem Statement 3

Student Feedback Analysis

A university collects student feedback and wants to categorize comments as Positive, Negative, or Neutral.

Questions:

a) Write a Zero-shot prompt to classify feedback sentiment.

b) Provide a One-shot prompt with one feedback example.

c) Create a Few-shot prompt using multiple labeled feedback samples.

d) Explain how examples improve sentiment classification accuracy.

Prompt

```
create a feedback analysis system for university
feedback recieved from students,
it should be categorized into positive, negative,
neutral feedback.
```

Code

```
'''create a feedback analysis system for university
feedback recieved from students,
it should be categorized into positive, negative,
neutral feedback.'''
# Sample feedback data
feedbacks = [
    "The course was very informative and the
professor was excellent!",
    "I found the lectures to be boring and
unengaging.",
    "The assignments were okay, nothing special.",
    "Great learning experience, I enjoyed every
part of it.",
    "The material was outdated and not relevant to
current industry standards.",
    "It was an average course, I learned some new
things but nothing extraordinary."
]
# Function to classify feedback
def classify_feedback(feedback):
    feedback_lower = feedback.lower()
    positive_keywords = ["excellent", "great",
"enjoyed", "informative", "good", "fantastic",
"amazing"]
    negative_keywords = ["boring", "unengaging",
"outdated", "dissatisfied", "poor", "bad",
"terrible"]

    if any(word in feedback_lower for word in
```

```
positive_keywords):
        return "Positive"
    elif any(word in feedback_lower for word in
negative_keywords):
        return "Negative"
    else:
        return "Neutral"
# Classify and print the category of each feedback
for feedback in feedbacks:
    category = classify_feedback(feedback)
    print(f"Feedback categorized as: {category}\n")


'''create a feedback analysis system for university
feedback recieved from students,
it should be categorized into positive, negative,
neutral feedback.
Feedback: “The course content was boring and
confusing.”
Sentiment: Negative
'''
feedback = "The course content was boring and
confusing."
sentiment = classify_feedback(feedback)
print(f"Feedback: \"{feedback}\"\nSentiment:
{sentiment}")

'''create a feedback analysis system for university
feedback recieved from students,
it should be categorized into positive, negative,
neutral feedback.

Feedback: “I really enjoyed the practical sessions
and found them very helpful.”
Sentiment: Positive

Feedback: “The assignments helped me learn a lot.”
Sentiment: Positive

Feedback: “The syllabus was okay, nothing special.”
```

```
Sentiment: Neutral

Feedback: “The exams were unfair and too
difficult.”
Sentiment: Negative
'''
feedback_list = [
    "I really enjoyed the practical sessions and
found them very helpful.",
    "The assignments helped me learn a lot.",
    "The syllabus was okay, nothing special.",
    "The exams were unfair and too difficult."
]
for feedback in feedback_list:
    sentiment = classify_feedback(feedback)
    print(f"Feedback: \"{feedback}\"\nSentiment:
{sentiment}\n")
```

Output

```
Feedback: "The course content was boring and confusing."
Sentiment: Negative
Feedback: "I really enjoyed the practical sessions and found them very helpful."
Sentiment: Positive

Feedback: "The assignments helped me learn a lot."
Sentiment: Neutral

Feedback: "The syllabus was okay, nothing special."
Sentiment: Neutral

Feedback: "The exams were unfair and too difficult."
Sentiment: Neutral
```

Problem Statement 4

Course Recommendation System

An online learning platform wants to recommend courses by classifying learner queries into Beginner, Intermediate, or Advanced levels.

Questions:

a) Write a Zero-shot prompt to classify learner queries.

b) Create a One-shot prompt with one example query.

c) Develop a Few-shot prompt with multiple labeled queries.

d) Discuss how Few-shot prompting improves recommendation

quality.

Prompt

```
create a online learning platform to recommend courses based on user preferences.
like Display only beginner, intermediate, advanced levels.
```

Code

```
'''create a online learning platform to recommend courses based on user preferences.
like Display only beginner, intermediate, advanced levels.'''
# Sample course data
courses = [
      {"title": "Python for Beginners", "level": "Beginner"},
      {"title": "Data Structures and Algorithms", "level": "Intermediate"},
    {"title": "Advanced Machine Learning", "level": "Advanced"},
     {"title": "Web Development Basics", "level": "Beginner"},
        {"title": "Database Management Systems", "level": "Intermediate"},
     {"title": "Deep Learning Techniques", "level": "Advanced"}
]
# Function to recommend courses based on user preference
def recommend_courses(level):
     recommended = [course for course in courses if course["level"].lower() == level.lower()]
    return recommended
# Get user preference
user_level = input("Enter your preferred course level (Beginner/Intermediate/Advanced): ")
recommended_courses = recommend_courses(user_level)
# Display recommended courses
```

```
if recommended_courses:
    print(f"Recommended {user_level} courses:")
    for course in recommended_courses:
        print(f"- {course['title']}")
else:
        print(f"No courses found for the level:
{user_level}")
'''create a online learning platform to recommend
courses based on user preferences.
like Display only beginner, intermediate, advanced
levels.
User Preference: Intermediate
Recommended Intermediate courses:
- Data Structures and Algorithms
- Database Management Systems
'''
user_level = "Intermediate"
recommended_courses = recommend_courses(user_level)
print(f"Recommended {user_level} courses:")
for course in recommended_courses:
    print(f"- {course['title']}")
'''create a online learning platform to recommend
courses based on user preferences.
like Display only beginner, intermediate, advanced
levels.
User Preference: Beginner
Recommended Beginner courses:
- Python for Beginners
- Web Development Basics
'''
user_level = "Beginner"
recommended_courses = recommend_courses(user_level)
print(f"Recommended {user_level} courses:")
for course in recommended_courses:
    print(f"- {course['title']}")
```

Output

```
Enter your preferred course level (Beginner/Intermediate/Advanced): Beginner
Recommended Beginner courses:
- Python for Beginners
- Web Development Basics
Recommended Intermediate courses:
- Data Structures and Algorithms
- Database Management Systems
Recommended Beginner courses:
- Python for Beginners
- Web Development Basics
```

Problem Statement 5

Social Media Post Moderation

A social media platform wants to classify posts into Acceptable, Offensive, or Spam.

Questions:

a) Write a Zero-shot prompt for post moderation.

b) Convert it into a One-shot prompt.

c) Design a Few-shot prompt using multiple examples.

d) Explain the challenges of Zero-shot prompting in content moderation.

Prompt

```
classify the post posted on social media into
acceptable or offensive or spam.
```

Code

```
'''classify the post posted on social media into
acceptable or offensive or spam.'''
posts = [
    "Check out this amazing deal on electronics!",
    "You are so stupid and ugly!",
    "Win a free iPhone now!!! Click here!",
    "Had a great day at the park with friends.",
    "This is the worst product I have ever bought.",
    "Congratulations! You've been selected for a
prize."
]
# Function to classify posts
def classify_post(post):
    post_lower = post.lower()
    offensive_keywords = ["stupid", "ugly", "idiot",
```

```
"hate", "dumb"]
    spam_keywords = ["win", "free", "click here",
"selected", "prize", "deal"]

    if any(word in post_lower for word in
offensive_keywords):
        return "Offensive"
    elif any(word in post_lower for word in
spam_keywords):
        return "Spam"
    else:
        return "Acceptable"
# Classify and print the category of each post
for post in posts:
    category = classify_post(post)
    print(f"Post categorized as: {category}\n")

'''classify the post posted on social media into
acceptable or offensive or spam.
Post: “You are an idiot and I hate you.”
Category: Offensive
'''
post = "You are an idiot and I hate you."
category = classify_post(post)

print(f"Post: \"{post}\"\nCategory: {category}")

'''classify the post posted on social media into
acceptable or offensive or spam.
Post: “Congratulations! You've won a free vacation.”
Category: Spam
Post: “Had a wonderful time with family.”
Category: Acceptable
Post: “This is a dumb idea.”
Category: Offensive
'''
post_list = [
    "Congratulations! You've won a free vacation.",
    "Had a wonderful time with family.",
    "This is a dumb idea."
```

```
]
for post in post_list:
    category = classify_post(post)
    print(f"Post: \"{post}\"\nCategory:
{category}\n")
```

Output

```
Post: "You are an idiot and I hate you."
Category: Offensive
Post: "Congratulations! You've won a free vacation."
Category: Spam

Post: "Had a wonderful time with family."
Category: Acceptable

Post: "This is a dumb idea."
Category: Offensive
```