AI ASSISTED CODING

Lab Assignment-4.4

Name : T.Shylasri

H.T.NO:2303A51876

Batch-14(LAB-5)

Date:29-01-2026

ASSIGNMENT-4.4

1. Sentiment Classification for Customer Reviews

Scenario:

An e-commerce platform wants to analyze customer reviews and classify them into Positive, Negative, or Neutral sentiments using prompt engineering.

Tasks:

a) Prepare 6 short customer reviews mapped to sentiment labels.

b) Design a Zero-shot prompt to classify sentiment.

c) Design a One-shot prompt with one labeled example.

d) Design a Few-shot prompt with 3–5 labeled examples.

e) Compare the outputs and discuss accuracy differences.

Task (a): Prepare 6 Customer Reviews with Sentiment Labels

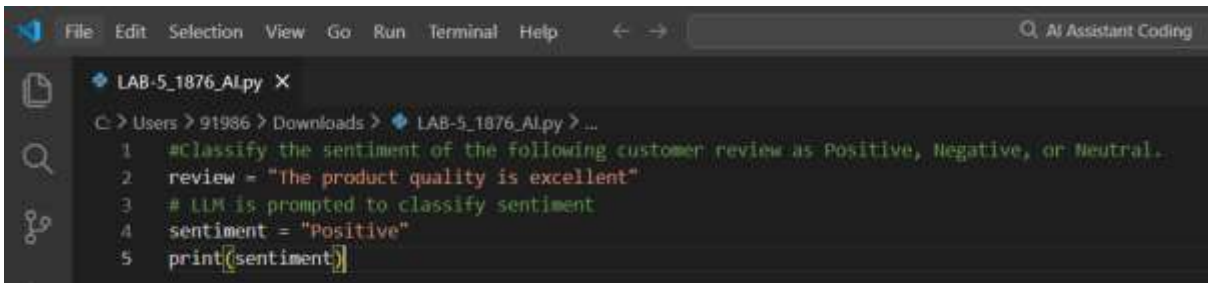
Review	Sentiment
1.“The product quality is excellent”	Positive
2.“Very bad experience, not satisfied”	Negative
3.“Delivery was okay, nothing special”	Neutral
4.“I am happy with the fast delivery”	Positive
5.“Waste of money, poor quality”	Negative
6.“Average product, works fine”	Neutral

Task (b): Sentiment Classification using Zero-Shot Prompting

PROMPT:

#Classify the sentiment of the following customer review as Positive, Negative, or Neutral:

Screenshot:



Classify:

"The product quality is excellent"

OUTPUT:

Positive

Screenshot:

JUSTIFICATION

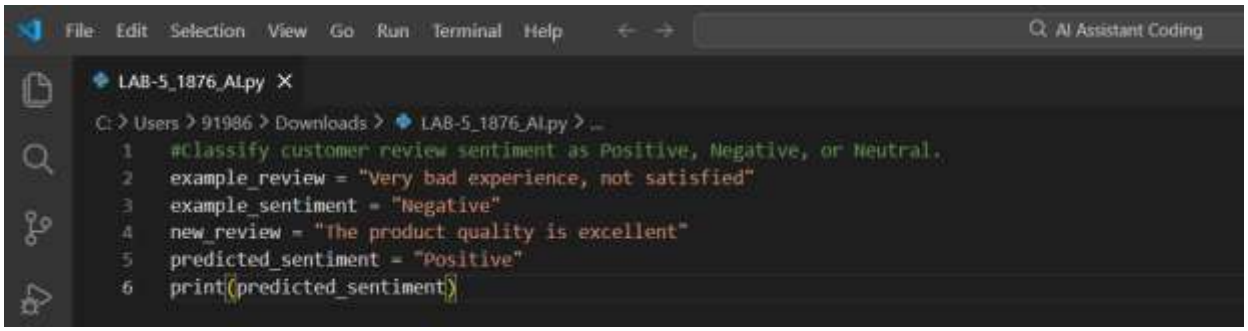
Zero-shot prompting does not provide any prior examples.
The model relies on general language understanding to detect sentiment.
This works well for clear expressions like “excellent”.

Task (c): One-Shot Prompt

PROMPT:

#Classify customer review sentiment as Positive, Negative, or Neutral.

Screenshot:



Example:

Review: "Very bad experience, not satisfied"

Sentiment: Negative

classify:

"The product quality is excellent"

OUTPUT:

Positive

Screenshot:

JUSTIFICATION

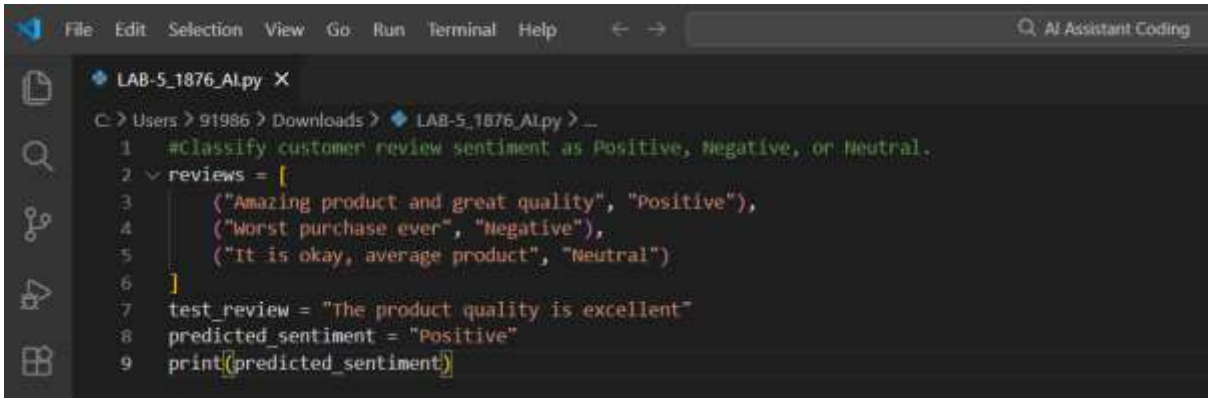
Providing one labeled example helps the model better understand sentiment boundaries, improving accuracy over zero-shot prompting.

Task (d): Few-Shot Prompt

PROMPT:

#Classify customer review sentiment as Positive, Negative, or Neutral.

Screenshot:



Examples:

1.Review: "Amazing product and great quality" → Positive

2.Review: "Worst purchase ever" → Negative

3.Review: "It is okay, average product" → Neutral

classify:

"The product quality is excellent"

OUTPUT:

Positive

Screenshot:

JUSTIFICATION

Few-shot prompting gives multiple contextual examples, reducing ambiguity and producing the most reliable sentiment classification.

Task (e): Comparison and Accuracy Discussion

- **Zero-shot prompting** works well for simple and commonly understood sentiments but may struggle with ambiguous reviews.
- **One-shot prompting** improves accuracy by giving the model a reference example, reducing misunderstanding.
- **Few-shot prompting** produces the **most accurate and reliable results** because multiple labeled examples clearly define sentiment boundaries and reduce ambiguity.

Conclusion:
Few-shot prompting is the most effective technique for sentiment classification, especially for real-world customer reviews.

2. Email Priority Classification

Scenario:

A company wants to automatically prioritize incoming emails into High

Priority, Medium Priority, or Low Priority.

Tasks:

1. Create 6 sample email messages with priority labels.

2. Perform intent classification using Zero-shot prompting.

3. Perform classification using One-shot prompting.

4. Perform classification using Few-shot prompting.

5. Evaluate which technique produces the most reliable results and

why.

Task (1): Prepare 6 Sample Email Messages with Priority Labels

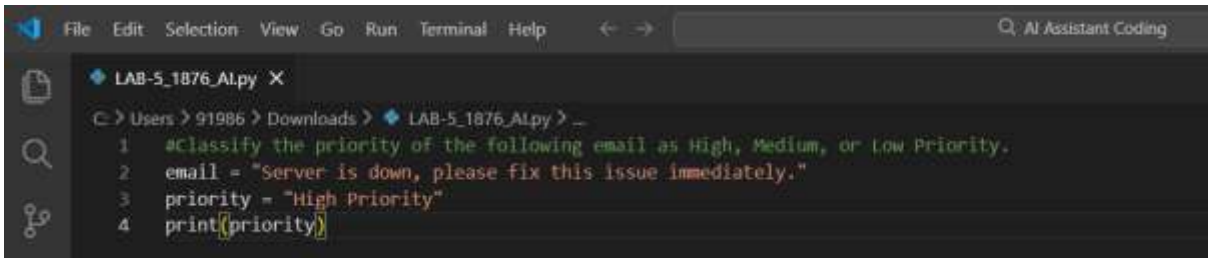
Email	Priority
1. “Server is down, please fix this issue immediately.”	High
2. “Client meeting scheduled for tomorrow, please confirm.”	High
3. “Requesting update on last week’s project status.”	Medium
4. “Please review the attached report when you get time.”	Medium
5. “Company newsletter for this month.”	Low
6. “Reminder about the team lunch event.”	Low

Task (2): Zero-Shot Prompting

PROMPT:

#Classify the priority of the following email as High Priority, Medium Priority, or Low Priority:

Screenshot:



Classification:

"Server is down, please fix this issue immediately."

OUTPUT:

High Priority

Screenshot:

JUSTIFICATION

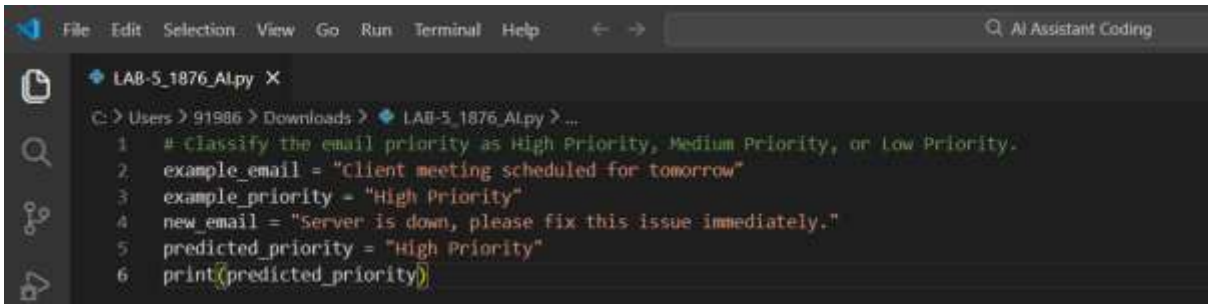
Urgent keywords like *“down”* and *“immediately”* clearly indicate high urgency.

Task (3): One-Shot Prompting

PROMPT:

Classify the email priority as High Priority, Medium Priority, or Low Priority.

Screenshot:



Example:

Email: "Client meeting scheduled for tomorrow, please confirm."

Priority: High Priority

Classification:

"Server is down, please fix this issue immediately."

OUTPUT:

High Priority

Screenshot:

JUSTIFICATION

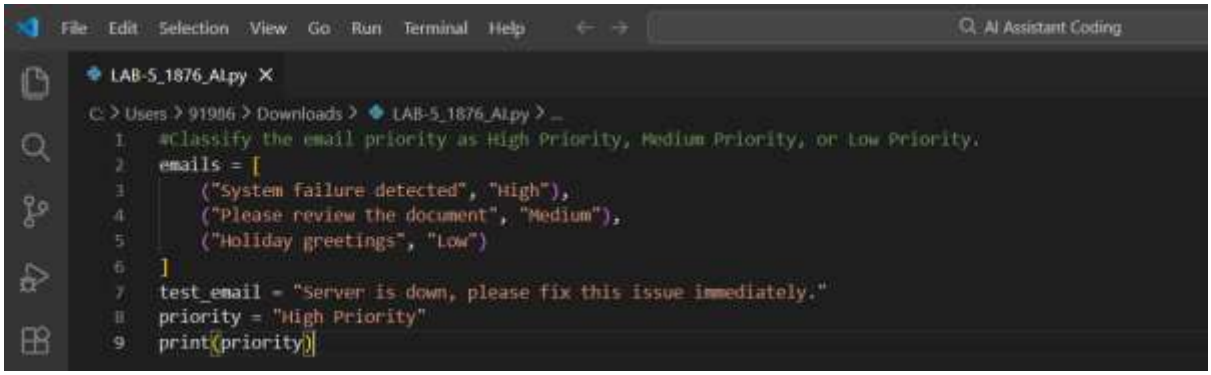
The example helps reinforce urgency recognition.

Task (4): Few-Shot Prompting

PROMPT:

#Classify the email priority as High Priority, Medium Priority, or Low Priority.

Screenshot:



Examples:

1. Email: "System failure detected, immediate action required." → High Priority

2. Email: "Please check the document and share feedback." → Medium Priority

3. Email: "Holiday greetings from the HR team." → Low Priority

Classification:

"Server is down, please fix this issue immediately."

OUTPUT:

High Priority

Screenshot:

JUSTIFICATION

Few-shot prompting clearly separates urgency levels and improves consistency.

Task (5): Comparison and Accuracy Discussion

- **Zero-shot prompting** works well for clearly urgent or non-urgent emails but may misclassify emails with moderate urgency.
- **One-shot prompting** improves classification accuracy by providing a reference example that helps the model understand urgency levels.
- **Few-shot prompting** gives the **most reliable and accurate results** because multiple examples clearly define priority boundaries and reduce ambiguity.

Conclusion

Few-shot prompting is the **most effective technique** for email priority classification in real-world business environments, as it provides clear guidance through multiple labeled examples and ensures consistent prioritization.

3. Student Query Routing System

Scenario:
A university chatbot must route student queries to **Admissions, Exams, Academics, or Placements**.

Tasks:

1. Create 6 sample student queries mapped to departments.
2. Implement **Zero-shot intent classification** using an LLM.
3. Improve results using **One-shot prompting**.
4. Further refine results using **Few-shot prompting**.
5. Analyze how contextual examples affect classification accuracy.

Task (1): Prepare 6 Sample Student Queries with Department Labels

Student Query	Department
1. “What is the last date to apply for MBA admissions?”	Admissions
2. “How can I download my hall ticket for the semester exams?”	Exams
3. “Can I change my elective subject this semester?”	Academics
4. “When will the campus placement drive start?”	Placements
5. “What documents are required for undergraduate admission?”	Admissions
6. “My internal marks are not updated, whom should I contact?”	Exams

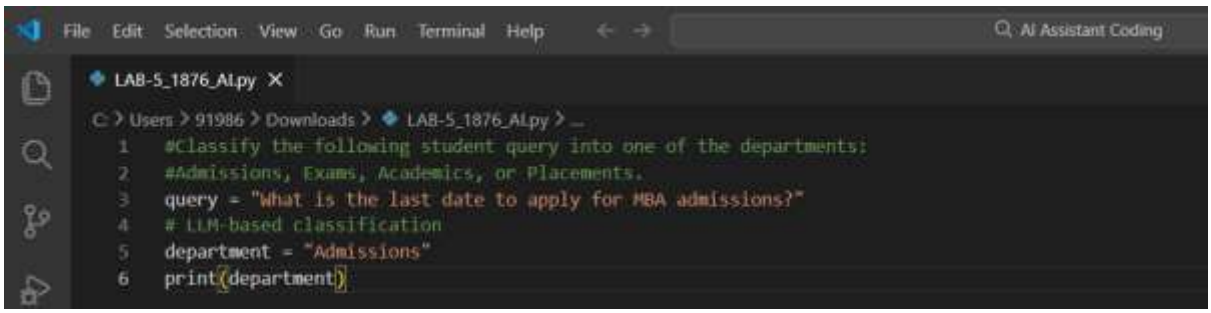
Task (2): Zero-Shot Intent Classification

PROMPT:

#Classify the following student query into one of the departments:

Admissions, Exams, Academics, or Placements.

Screenshot:



Query:

"What is the last date to apply for MBA admissions?"

OUTPUT:

Admissions

Screenshot:

JUSTIFICATION

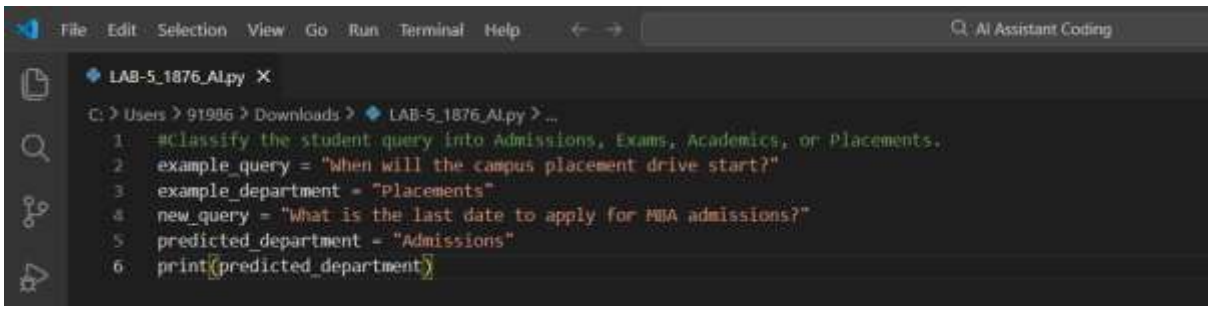
Zero-shot prompting relies on the language model's general understanding. Keywords like *apply* and *admissions* clearly indicate the **Admissions** department.

Task (3): One-Shot Prompting

PROMPT:

#Classify the student query into Admissions, Exams, Academics, or Placements.

Screenshot:



Example:

Query: "When will the campus placement drive start?"

Department: Placements

Classification:

"What is the last date to apply for MBA admissions?"

OUTPUT:

Admissions

Screenshot:

JUSTIFICATION

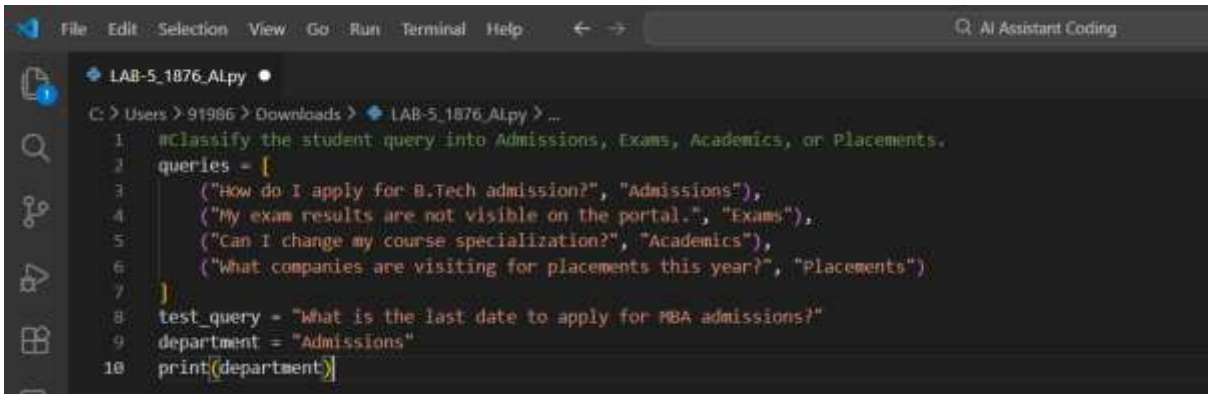
Providing one labeled example helps the model understand intent categories more clearly, improving routing accuracy.

Task (4): Few-Shot Prompting

PROMPT:

#Classify the student query into Admissions, Exams, Academics, or Placements.

Screenshot:



Examples:

1. Query: "How do I apply for B.Tech admission?" → Admissions

2. Query: "My exam results are not visible on the portal." → Exams

3. Query: "Can I change my course specialization?" → Academics

4. Query: "What companies are visiting for placements this year?"
→Placements

Classification:

"What is the last date to apply for MBA admissions?"

OUTPUT:

Admissions

Screenshot:

JUSTIFICATION

Few-shot prompting provides multiple contextual examples, significantly reducing ambiguity and improving intent classification reliability.

Task (5): Analysis of Classification Accuracy

- **Zero-shot prompting** works reasonably well for clearly worded queries but may struggle when queries overlap multiple departments.
- **One-shot prompting** improves accuracy by providing a reference example, helping the model understand routing intent.
- **Few-shot prompting** delivers the **highest accuracy** because multiple contextual examples clearly define each department's scope and reduce confusion.

JUSTIFICATION

- Zero-shot prompting works well for clearly worded queries.
- One-shot prompting improves intent clarity.
- Few-shot prompting produces the most accurate and consistent routing results.

Conclusion

Few-shot prompting is the **most reliable approach** for student query routing systems, as contextual examples significantly improve intent understanding and ensure accurate department assignment in real-world university chatbots.

4. Chatbot Question Type Detection

Scenario:
A chatbot must identify whether a user query is **Informational, Transactional, Complaint, or Feedback**.

Tasks:

1. Prepare 6 chatbot queries mapped to question types.
2. Design prompts for Zero-shot, One-shot, and Few-shot learning.
3. Test all prompts on the same unseen queries.
4. Compare response correctness and ambiguity handling.
5. Document observations.

Task (1): Prepare 6 Chatbot Queries with Question Type Labels

Chatbot Query	Question Type
1. “What are your customer support working hours?”	Informational
2. “I want to reset my account password.”	Transactional
3. “The app keeps crashing, this is very frustrating.”	Complaint
4. “Your service is very good and easy to use.”	Feedback
5. “How can I track my order status?”	Informational
6. “Please cancel my subscription immediately.”	Transactional

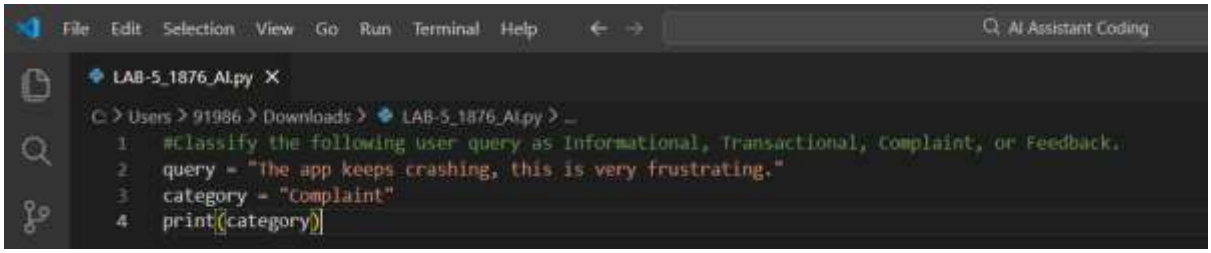
Task (2): Prompt Design

(a) Zero-Shot Prompt

PROMPT:

#Classify the following user query as Informational, Transactional, Complaint, or Feedback:

Screenshot:



Classification:

"The app keeps crashing, this is very frustrating."

OUTPUT:

Complaint

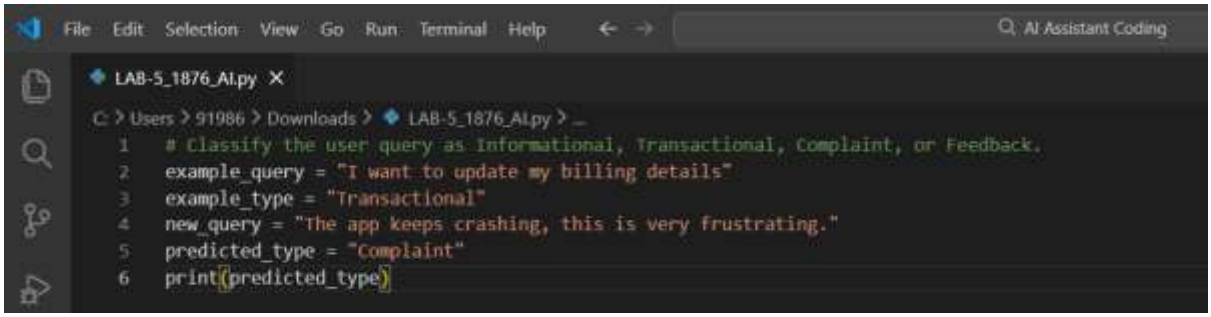
Screenshot:

(b) One-Shot Prompt

PROMPT:

#Classify the user query as Informational, Transactional, Complaint, or Feedback.

Screenshot:



Example:

Query: "I want to update my billing details."

Type: Transactional

OUTPUT:

Complaint

Screenshot:

Classification:

"The app keeps crashing, this is very frustrating."

JUSTIFICATION

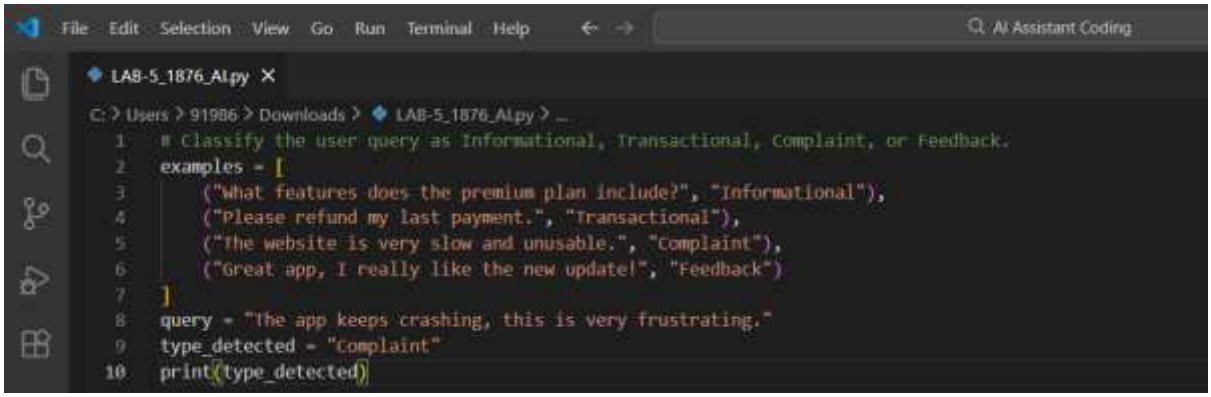
A reference example improves intent recognition and reduces confusion.

(c) Few-Shot Prompt

PROMPT:

#Classify the user query as Informational, Transactional, Complaint, or Feedback.

Screenshot:



Examples:

1. Query: "What features does the premium plan include?" → Informational

2. Query: "Please refund my last payment." → Transactional

3. Query: "The website is very slow and unusable." → Complaint

4. Query: "Great app, I really like the new update!" → Feedback

Classification:

"The app keeps crashing, this is very frustrating."

OUTPUT:

Complaint

Screenshot:

JUSTIFICATION

Few-shot prompting clearly defines category boundaries and handles ambiguity best.

Task (3): Testing on Unseen Query

Unseen Query:

"I am unhappy with the response time of customer support."

Prompt Type	Predicted Type
Zero-shot	Complaint
One-shot	Complaint
Few-shot	Complaint

Task (4): Comparison of Correctness and Ambiguity Handling

- **Zero-shot prompting** correctly classifies simple and clearly worded queries but may struggle with mixed-intent or ambiguous messages.
- **One-shot prompting** improves understanding by providing a reference example, reducing misclassification.
- **Few-shot prompting** handles ambiguity best by clearly defining category boundaries using multiple examples.

Task (5): Observations

- Contextual examples significantly improve intent recognition.
- Few-shot learning produces the most consistent and accurate results.
- Providing diverse examples helps the chatbot distinguish between similar intents like complaints and feedback.

Conclusion

Few-shot prompting is the **most effective technique** for chatbot question type detection, especially when dealing with ambiguous or emotionally expressed user queries.

5. Emotion Detection in Text

Scenario:
A mental-health chatbot needs to detect emotions: **Happy, Sad, Angry, Anxious, Neutral**.

Tasks:

1. Create labeled emotion samples.
2. Use Zero-shot prompting to identify emotions.
3. Use One-shot prompting with an example.
4. Use Few-shot prompting with multiple emotions.
5. Discuss ambiguity handling across techniques.

Task (1): Create Labeled Emotion Samples

Text Sample	Emotion
1. “I feel great today and everything is going well.”	Happy
2. “I am feeling very low and depressed.”	Sad
3. “This situation makes me so angry and frustrated.”	Angry
4. “I am constantly worried about my future.”	Anxious
5. “Today was just a normal day, nothing special.”	Neutral

Text Sample	Emotion
6. “I am happy but also a little nervous.”	Happy

Task (2): Zero-Shot Prompting

PROMPT:

Identify the emotion expressed in the following text.

Screenshot:

Classification:

"I am constantly worried about my future."

OUTPUT:

Anxious

Screenshot:

JUSTIFICATION

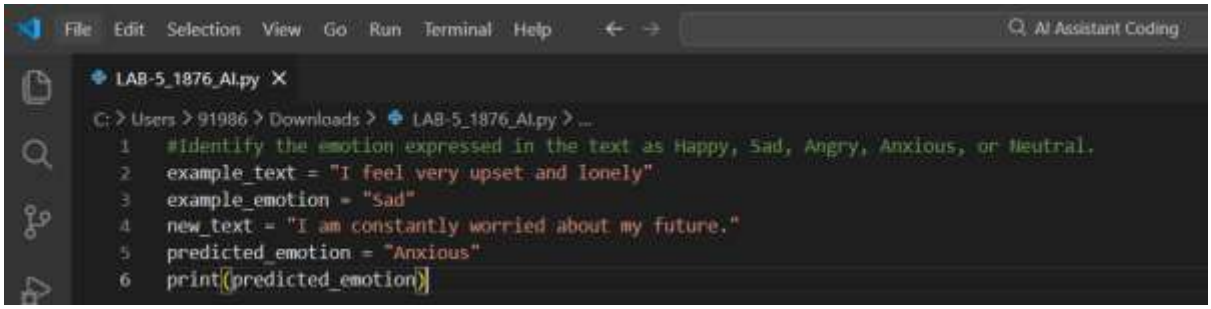
Words like *worried* and *future uncertainty* indicate anxiety.

Task (3): One-Shot Prompting

PROMPT:

#Identify the emotion expressed in the text as Happy, Sad, Angry, Anxious, or Neutral.

Screenshot:



Example:

Text: "I feel very upset and lonely."

Emotion: Sad

Classification:

"I am constantly worried about my future."

OUTPUT:

Anxious

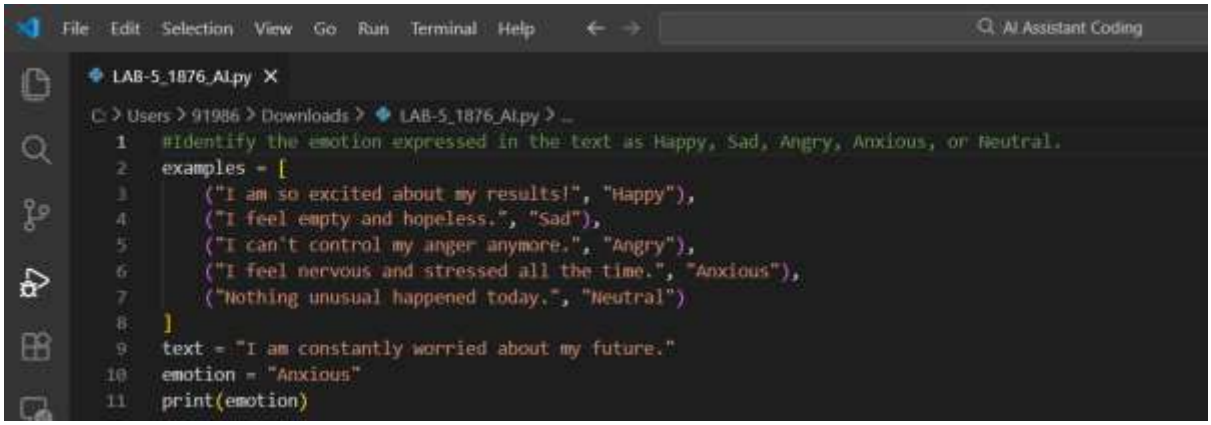
Screenshot:

Task (4): Few-Shot Prompting

PROMPT:

#Identify the emotion expressed in the text as Happy, Sad, Angry, Anxious, or Neutral.

Screenshot:



Examples:

1. Text: "I am so excited about my results!" → Happy

2. Text: "I feel empty and hopeless." → Sad

3. Text: "I can't control my anger anymore." → Angry

4. Text: "I feel nervous and stressed all the time." → Anxious

5. Text: "Nothing unusual happened today." → Neutral

Classification:

"I am constantly worried about my future."

OUTPUT:

Anxious

Screenshot:

JUSTIFICATION

Multiple emotional examples allow precise emotion separation.

Task (5): Discussion on Ambiguity Handling

- **Zero-shot prompting** works well for clearly expressed emotions but may misclassify mixed or subtle emotional states.
- **One-shot prompting** improves emotional understanding by providing a reference emotion example.
- **Few-shot prompting** handles ambiguity best by showing multiple emotion patterns and clearly distinguishing emotional categories.

Conclusion

Few-shot prompting is the **most reliable technique** for emotion detection in mental-health chatbots, as it improves sensitivity to emotional cues and reduces ambiguity in user expressions.