

AI ASSISTED LAB EXAM

K.Bharath

2403A52065

BATCH – 03

QUESTION – 1

Q1. Zero-shot Prompting in Healthcare [5M]

Scenario: A doctor uses an AI tool to quickly triage patient symptoms into “Mild,” “Moderate,” or “Severe.”

- Task 1: Write a zero-shot prompt that classifies the severity of symptoms without giving any examples.

PROMPT –

You are a symptom severity classifier. Given: age,

symptoms, duration/onset, history/comorbidities, meds/allergies, and vitals (HR, BP, RR, Temp, SpO2), output exactly one label: Mild, Moderate, or Severe.

Criteria:

- Mild: self-limited, stable vitals, no red flags.
- Moderate: noticeable impairment or mildly abnormal vitals/risk; needs timely outpatient care.
- Severe: any red flag/instability—cardiac chest pain, dyspnea at rest, SpO2<92%, SBP<90, RR>30, new confusion/focal deficit, uncontrolled bleeding, anaphylaxis, Temp>40° C with toxicity, severe dehydration, rapid decline, severe uncontrolled pain.

Return only:

severity: <Mild |Moderate |Severe>

reasoning: 1–2 brief lines citing key findings

next _step: ED now | Urgent clinic | Routine care

CODE SCREEN SHOTS –

```

1 def classify_severity(age, symptoms, duration_onset, history, meds_allergies, hr=None, bp=None, rr=None,
2     """
3     Classify symptom severity as Mild, Moderate, or Severe based on provided criteria.
4
5     Returns a string with exactly three lines:
6     severity: <Mild|Moderate|Severe>
7     reasoning: <1-2 brief lines>
8     next_step: <ED now | Urgent clinic | Routine care>
9     """
10
11     def _contains(text, keywords):
12         t = (text or "").lower()
13         return any(k in t for k in keywords)
14
15     severity = None
16     next_step = None
17     reasons = []
18
19     # --- Severe (red flags / instability) ---
20     if spo2 is not None and spo2 < 92:
21         reasons.append("SpO2 < 92%")
22     if hr is not None and hr > 100:
23         reasons.append("HR > 100")
24     if bp is not None and (sbp < 90 or dbp > 160):
25         reasons.append("BP < 90/160")
26     if rr is not None and rr > 30:
27         reasons.append("RR > 30")
28     if temp is not None and temp > 40:
29         reasons.append("Temp > 40°C")
30     if _contains(symptoms, [
31         "cardiac chest pain", "chest pain", "dyspnea at rest", "dyspnea", "shortness of breath",
32         "new confusion", "focal deficit", "stroke", "weakness one side", "bleeding", "anaphylaxis",
33         "rapidly worsening", "severe pain"
34     ]):
35         reasons.append("Red-flag symptoms")
36
37     if len(reasons) == 0:
38         severity = "Mild"
39         next_step = "Routine care"
40     elif len(reasons) < 3:
41         severity = "Moderate"
42         next_step = "Urgent clinic"
43     else:
44         severity = "Severe"
45         next_step = "ED now"
46
47     return f"severity: {severity}\nreasoning: {', '.join(reasons)}\nnext_step: {next_step}"

```

```

9     # --- Severe (red flags / instability) ---
10     if spo2 is not None and spo2 < 92:
11         reasons.append("SpO2 < 92%")
12     if bp:
13         try:
14             sbp = int(str(bp).split("/")[0])
15             if sbp < 90:
16                 reasons.append("SBP < 90")
17         except Exception:
18             pass
19     if rr is not None and rr > 30:
20         reasons.append("RR > 30")
21     if temp is not None and temp > 40:
22         reasons.append("Temp > 40°C")
23     if _contains(symptoms, [
24         "cardiac chest pain", "chest pain", "dyspnea at rest", "dyspnea", "shortness of breath",
25         "new confusion", "focal deficit", "stroke", "weakness one side", "bleeding", "anaphylaxis",
26         "rapidly worsening", "severe pain"
27     ]):
28         reasons.append("Red-flag symptoms")
29
30     if len(reasons) == 0:
31         severity = "Mild"
32         next_step = "Routine care"
33     elif len(reasons) < 3:
34         severity = "Moderate"
35         next_step = "Urgent clinic"
36     else:
37         severity = "Severe"
38         next_step = "ED now"
39
40     return f"severity: {severity}\nreasoning: {', '.join(reasons)}\nnext_step: {next_step}"

```

[Review next file >](#)

```

if len(reasons) > 0:
    severity = "Severe"
    next_step = "ED now"
    reasoning = ". ".join(reasons) + "."
    return f"severity: {severity}\nreasoning: {reasoning}\nnext_step: {next_step}"

# --- Moderate: impairment or mildly abnormal vitals / risk ---
mod_reasons = []
if _contains(symptoms, ["impairment", "discomfort", "limiting", "can't work", "can't work", "reduced acti
    mod_reasons.append("Impairment present")
if hr is not None and (hr < 60 or hr > 100):
    mod_reasons.append("Abnormal HR")
if (temp is not None) and (37.8 <= temp <= 40):
    mod_reasons.append("Fever without toxicity")
if (history or "").strip():
    mod_reasons.append("Risk history/comorbidities")

if len(mod_reasons) > 0:
    severity = "Moderate"
    next_step = "Urgent clinic"
    reasoning = ". ".join(mod_reasons[:2]) + "See brief
    return f"severity: {severity}\nreasoning: {reasoning}\nnext_step: {next_step}"

```

```

# --- Mild ---
severity = "Mild"
next_step = "Routine care"
reasoning = "Self-limited symptoms, stable vitals, no red flags."
return f"severity: {severity}\nreasoning: {reasoning}\nnext_step: {next_step}"

def _to_int(value):
    try:
        return int(value) if value != "" else None
    except Exception:
        return None

def _to_float(value):
    try:
        return float(value) if value != "" else None
    except Exception:
        return None

```

```

34 if __name__ == "__main__":
35     # Minimal CLI for quick testing in the exact output format
36     print("Enter patient details (press Enter to skip optional vitals):")
37     age = _to_int(input("Age: ").strip()) or 0
38     symptoms = input("Symptoms: ").strip()
39     duration_onset = input("Duration/Onset: ").strip()
40     history = input("History/Comorbidities: ").strip()
41     meds_allergies = input("Meds/Allergies: ").strip()
42
43     hr = _to_int(input("HR (bpm): ").strip())
44     bp = input("BP (e.g., 120/80): ").strip() or None
45     rr = _to_int(input("RR (breaths/min): ").strip())
46     temp = _to_float(input("Temp (°C): ").strip())
47     spo2 = _to_int(input("SpO2 (%): ").strip())
48
49     result = classify_severity(
50         age=age,
51         symptoms=symptoms,
52         duration_onset=duration_onset,
53         history=history,
54         meds_allergies=meds_allergies,
55         hr=hr,
56         bp=bp,
57         rr=rr,
58         temp=temp,
59         spo2=spo2,
60     )
61
62     print("\n" + result)

```

[Review next file >](#)

OUTPUT –

```

PS C:\Users\karthik\IdeaProjects\AI ASSISTED> & C:/Users/karthi
sers/karthik/IdeaProjects/AI ASSISTED/assignment6.4/K/AI.PY"
● Enter patient details (press Enter to skip optional vitals):
Age: 19
Symptoms: FEVER
Duration/Onset: 1
History/Comorbidities: NO
Meds/Allergies: TYNOL TABLET / NO
HR (bpm): NORMAL
BP (e.g., 120/80): NORMAL
RR (breaths/min): 60
Temp (°C): 60
SpO2 (%): 70

severity: Severe
reasoning: SpO2 < 92%. RR > 30. Temp > 40°C.
next_step: ED now
○ PS C:\Users\karthik\IdeaProjects\AI ASSISTED> █

```

OBSERVATION –

Code observations

- Severe first: Red flags (SpO2<92, SBP<90, RR>30, Temp>40°C, critical symptoms) trigger immediate "ED now" classification.
- Moderate second: Impairment, abnormal HR, fever 37.8–40°C, or any history bumps to "Urgent clinic" with brief reasoning.
- Mild default: If no red flags or moderate criteria, defaults to "Routine care" with generic reasoning

about stable vitals.

Output observations

- Format compliance: Output matches exact spec with severity/reasoning/next _ step on separate lines.
- Reasoning brevity: Keeps to 1–2 lines as requested, using key findings to justify classification.
- Next step alignment: "ED now" for Severe, "Urgent clinic" for Moderate, "Routine care" for Mild—appropriate escalation.

Task 2 - : Create a scenario where an AI assistant needs to guide a patient about diet. Write two

prompts: one without context and one with detailed context (e.g., age, health condition, dietary

restrictions.

PROMPT 1

WITHOUT CONTEXT

You are a diet guidance assistant. Provide a simple 7-day healthy eating plan with breakfast, lunch, dinner, and 1–2 snacks per day. Keep it affordable, easy to follow, and culturally neutral. Include hydration tips and 3 safety

cautions. Avoid medical claims.

PROPMT – 2

WITH CONTEXT

You are a diet guidance assistant. Create a 7- day meal plan tailored to this patient:

- Age: 28
- Sex: Female
- Height/Weight: 160 cm, 68 kg
- Health: PCOS; borderline high LDL
- Medications: None
- Dietary restrictions: Vegetarian; lactose intolerant; no mushrooms
- Activity: Light (walks 20–30 min/day)
- Budget: Low to moderate; prefers Indian home- style foods
- Goals: Weight management and improved lipid profile

Instructions:

- 1) Give a daily kcal range and macro emphasis (higher fiber/protein, lower saturated fat).
- 2) Provide 7 days of meals (breakfast, lunch, dinner, 1–2

snacks) with portions.

3) Use low- GI carbs, legumes, vegetables, nuts/seeds; limit sugar and refined flour.

4) Add a grocery list, basic prep tips, and hydration plan.

5) Include 4 safety cautions and when to seek medical advice.

Keep it concise and practical.

OBSERVATION -

- Clear, detailed context (PCOS, LDL, vegetarian, lactose intolerant, Indian foods) enables tailored, safe advice.
- Instructions prioritize low- GI, higher protein/fiber, and lower saturated fat— appropriate for PCOS and lipid control.
- Actionable outputs (kcal range, portions, 7- day plan, grocery list, hydration, safety cautions) ensure practicality and adherence.

QUESTION 2

Q2. One-shot vs Few-shot for Customer Support [5M]

Scenario: An e-commerce company uses AI to classify support emails into "Refund," "Order Status,"

or "Technical Issue."

- Task 1: Write:

- o A one-shot prompt with 1 example of classification.

- o A few-shot prompt with 3–4 examples.

ONE SHOT PROMPT 1 EXAMPLE

You are a customer support email classifier. Categorize each email into exactly one of: "Refund", "Order Status", or "Technical Issue".

Example:

Email: "I have not received my order yet. The tracking number is 1234567890."

Classification: Order Status

Now classify this email

FEW SHORT PROMPTS

You are a customer support email classifier. Categorize each email into exactly one of: "Refund", "Order Status", or "Technical Issue".

Examples:

1. Email: "I want to return the product I bought because it is damaged."

Classification: Refund

2. Email: "Where is my order? I placed it two weeks ago."

Classification: Order Status

3. Email: "The website is not loading properly on my phone."

Classification: Technical Issue

4. Email: "I was charged twice for the same order."

Classification: Refund

OBSERVATION –

- Clear intent mapping: Examples

show refund (damage, double charge), order status (tracking, delivery), and technical (website/app issues).

- Pattern recognition: Teaches the AI to identify key phrases like "return", "where is my order", "not loading", "charged twice".
- Consistent format: Each example follows the same Email/Classification structure for easy learning.

Task 2: Use the same incoming email text for both prompts. Compare how the outputs differ and

explain why.

Test Email: "I can't log into my account and I'm worried about my recent order. The app keeps crashing when I try to check the status." One-shot Prompt Result:

- Classification: Technical Issue
- Reasoning: Focuses on primary technical problems ("can't log in", "app keeps crashing")
- Confidence: Lower, may be uncertain due to mixed content

Few-shot Prompt Result:

- Classification: Technical Issue
- Reasoning: Prioritizes technical issues over order concerns due to

pattern reinforcement

- Confidence: Higher, more consistent classification

Why Outputs Differ:

1. Pattern Learning: Few-shot examples teach the AI to recognize technical issues as priority when multiple problem types exist
1. Confidence Level: Multiple examples create stronger pattern recognition, leading to more decisive classification
1. Priority Understanding: Few-shot learns that technical problems (login, app crashes) should override order status concerns in mixed-content emails

Key Difference: While both classify as

"Technical Issue," the few-shot approach provides more confident and consistent results due to enhanced pattern recognition from multiple examples.

Observations

- Same classification result: Both one-shot and few-shot classify the mixed email as "Technical Issue" due to login/app crash keywords.
- Confidence difference: Few-shot provides higher confidence and consistency through pattern reinforcement from multiple examples.

- Learning effect: Few-shot better handles mixed-content emails by prioritizing technical problems over order concerns, while one-shot may struggle with ambiguity.

END -