**Customer Support Ticket Analyser**

# Objective:

To create a support ticket analyser agent which performs mainly 3 tasks:

* Summarize the issue so that its easily understood by excutive.
* Based on message, sentiment, revenue and customer tier, suggest urgency level of the ticket. i.e. Critical/P1, High/P2, Medium/P3, Low/P4
* Classify the issue based on the subject and message provided. i.e. bug, feature request, billing, security, login issue, UI feedback, documentation, general query with a confidence score.

# Working:

## Input:

  {

    "ticket\_id": "SUP-001",

    "customer\_tier": "free",

    "subject": "This product is completely broken!!!",

    "message": "Nothing works! I can't even log in. This is the worst software I've ever used.",

    "previous\_tickets": 0,

    "monthly\_revenue": 0,

    "account\_age\_days": 2

  }

## Process:

I have used pydantic BaseModel for data validation, parsing, and structured output formatting.

For **Analyser Agent,** I used prompt:

"You are a support ticket analyzer.\n"

"Given a ticket with subject, message, revenue, and account age, perform the following:\n"

"- Using subject and message, summarize the core issue in one short sentence explaining the issue to support.\n"

"- Using subject and message, analyze the customer's tone to determine the sentiment (positive, or neutral, or negative).\n"

"- Based on tone, content, and customer tier, monthly\_revenue, account\_age\_days, assess urgency as Critical/P1, High/P2, Medium/P3, Low/P4. Determine urgency level. Even if sentiment is negative, consider customer tier and value. \n"

"Return ONLY a JSON with: summarized\_issue,sentiment,urgency\_level"

**Classification Agent** prompt:

"You are an issue classification expert for customer support.\n"

"Classify the issue based on the subject and message provided.\n"

"Only use these categories: bug, feature\_request,billing, security, login\_issue, ui\_feedback, documentation, general\_query.\n"

"Return a JSON with exactly two fields: issue\_type (string), confidence\_score (float between 0 and 1).\n"

"Do not explain or add any other text. Output only a JSON object."

## Output:

O/P is a combination of both (Ticket info + classification details)

Sending POST request to <http://localhost:5000/analyze_ticket>...

✅ Success! Response received:

{

"customer\_details": {

"account\_age\_days": 2,

"customer\_tier": "free",

"monthly\_revenue": 0,

"previous\_tickets": 0

},

"issue\_routing": {

"confidence\_score": 0.8,

"issue\_type": "bug"

},

"ticket\_details": {

"sentiment": "negative",

"summarized\_issue": "Customer unable to log in and experiencing issues with the product",

"urgency\_level": "High/P2"

},

"ticket\_id": "SUP-001"

}

# Issues:

* **Model limitations:** Firstly, I was using small model (7b) but most of the times it not unable to provide o/p as I wanted (hallucinating). Then I used llama3-70b-8192 which works perfectly.
* **Output parsing:** Second problem I faced was parsing o/p. It was because of models inconsistent o/p which was solved by adding prompt like “Do not explain or add any other text. Output only a JSON object.” Or specifying exact keys.

# What I wanted to add:

* To use CoT like prompting and specify exactly how agent should assign urgency for each ticket. This eould be nice but 70b models are able to handle it. Maybe with CoT we can use smaller model.
* Add another RAG based agent which help user to solve general query like login issues. If a general issue not solved by bot then, issue type can be updated as bug.
* This agent can also use to ask extra information from user if issue was unclear.
* And create something like autogen swarm.