WhatsApp Expense Tracker Bot – Agentic AI

# Objective

The objective of this project is to create an intelligent WhatsApp bot that helps users track their daily expenses through simple natural language messages. It leverages Flask for backend server, Twilio for WhatsApp messaging, SQLite for storing expenses, and OpenAI GPT for understanding user inputs and classifying expenses or queries.

# Technologies and Libraries Used

1. Flask – To create a webhook for receiving WhatsApp messages.  
2. Twilio – To connect the bot with WhatsApp.  
3. SQLite – A lightweight database to store user expenses.  
4. OpenAI – Used to integrate GPT for intelligent message parsing.  
5. Pyngrok – To expose the Flask server to the internet during development.  
6. Dateparser – For flexible parsing of natural language dates.  
7. Nest\_asyncio – To allow Flask and ngrok to run inside Jupyter notebooks.

# How the Bot Works

1. The Flask app receives messages from WhatsApp via Twilio's webhook system.  
2. The message is analyzed using OpenAI GPT to determine the user's intent:  
 - If it's an expense (e.g., 'Spent ₹200 on snacks'), the bot extracts the amount, category, and date.  
 - If it's a query (e.g., 'How much did I spend on travel?'), the bot calculates relevant information.  
3. Extracted expense details are saved to an SQLite database.  
4. For queries, the bot fetches and aggregates data based on filters like category or time.  
5. The response is returned to the user over WhatsApp using Twilio’s MessagingResponse.

# Features Implemented

- Add expenses through natural language (e.g., 'Spent ₹300 on lunch yesterday').  
- Extract and categorize expenses using GPT.  
- Parse natural language dates using dateparser.  
- Store structured expense data in a local SQLite database.  
- Answer user queries like:  
 • 'How much did I spend this week?'  
 • 'What is my total expense on food?'  
- Summarize total and category-wise expenses.

Assumptions

|  |  |
| --- | --- |
| 1. User Input Format | The user will send natural language messages like "Spent 500 on pizza" or "Paid 1200 for medicines". |
| 2. Single Expense per Message | Each incoming message contains only one expense. |
| 3. Amount Format | Amounts will be in numeric format (e.g., "500", "Rs 300", "₹1000"). |
| 4. Implicit Date | If no date is mentioned, assume the current date. |
| 5. User Identity | Each user's WhatsApp number (sender ID) uniquely identifies their expenses. |
| 6. Internet & API Availability | The bot relies on internet access to call OpenAI GPT and Twilio, and assumes both are reachable. |

|  |  |
| --- | --- |
| 7. Category Inference | GPT can accurately classify the expense into one of: food, travel, medicine, entertainment, or other. |
| 8. Prompt Success | Prompting GPT with a short context is enough to yield a relevant category. |
| 9. Regex Sufficiency | Regex-based extraction is reliable for amount and category hints when applicable. |
| 10. Query Interpretation | Queries like "How much this month?" or "Total food spending" are interpretable using keywords like "month", "on food", etc. |
| 11. Local Storage | SQLite is assumed to be adequate for storing expense logs (single-user or low-volume). |
| 12. Flat Schema | All expense data can be stored in a flat table with fields: amount, user, category, description, date. |
| 13. Timestamp Accuracy | datetime.now() is accurate enough if no explicit date is parsed. |

|  |  |
| --- | --- |
| 14. Flask Webhook | The Flask server is reachable via ngrok (or deployed) and registered as Twilio's webhook endpoint. |
| 15. Twilio Sandbox Setup | You have a Twilio WhatsApp sandbox or approved number, and it’s correctly configured. |
| 16. Sequential Execution | Flask + ngrok + notebook run in one runtime (e.g., via Jupyter with nest\_asyncio). |

# Conclusion

This project showcases how AI can be used in an agentic way to understand natural user input and provide personalized financial tracking.   
By combining LLMs with structured storage and an intuitive interface like WhatsApp, it delivers a powerful and user-friendly solution.