GROUP 18

CTRL C , CTRL V

DOMAIN 3, TASK 2

ECONOMIC EMPOWERMENT THROUGH AI

📄 Technical Documentation of MEX Assistant

📝 Introduction

The MEX Assistant is a real-time, chat-based analytics platform designed for food delivery merchants. Powered by OpenAI's GPT-4o model and integrated with data pipelines and visualization tools, this assistant empowers non-technical users to make data-driven decisions. Merchants can inquire about revenue trends, peak order hours, product performance, and operational bottlenecks through a natural language interface. This project demonstrates the seamless blend of real-time simulation, AI-powered reasoning, and user-friendly visual feedback, tailored specifically for small to mid-sized F&B outlets.

🧱 System Architecture

The architecture of the **Merchant Insights Assistant** is designed to be modular, scalable, and user-friendly. It consists of the following layers and components:

🧑‍💼 1. Merchant User (Frontend Interface)

- **Interface**: A Streamlit-based web application that serves as the entry point for users.
- **Role**: Merchants interact with the system through a conversational UI, asking questions about their delivery performance and business trends.

🔄 2. Streamlit Backend Logic

- **User Authentication**: Manages merchant login and session tracking.
- **Chatbot Interface**: Facilitates communication between the merchant and the AI assistant.
- **Graph & Report Export**: Generates visual charts and downloadable reports (PDFs and PPTX).
- **Session State Handling**: Maintains user interaction history, graphs, and generated outputs for continuity during sessions.

📊 3. Analytics Engine

- **NLP-to-Code Prompting**: Converts merchant queries into executable Python code using structured prompts.
- **GPT-4o OpenAI Agent**: The LLM interprets questions and generates code to analyze data and create visual outputs.
- **Real-time Simulation**: Simulates ongoing order and delivery activity by manipulating CSV data to mimic a live environment.
- **Visualization Layer**: Uses libraries like Matplotlib and Seaborn to create insightful graphs for metrics like revenue, delay trends, and product performance.

📂 4. Data Layer

- **Input Data**: Relies on historical and real-time CSV files (e.g., `transaction_bagel.csv`, `transaction_noodle.csv`).
- **Merged DataFrames**: Consolidates data from different merchants and time periods for analysis. Includes support for date filtering, merchant-specific views, and cumulative trends.

🌐 5. External Services

- **OpenAI API**: Powers the natural language understanding and code generation through GPT-4o.
- **Twilio API**: Sends SMS alerts to merchants when critical events are detected (e.g., long wait times).
- **ReportLab**: Used to export chat summaries and performance insights into PDF documents.
- **python-pptx**: Allows exporting of conversation and graphs into PowerPoint presentations.

🔁 Flow Summary

The user inputs a question → it's processed by the backend and passed to GPT-4o → the generated code analyzes the relevant data → results are visualized and sent back to the frontend → optional reports or alerts are triggered via third-party integrations.

Design Choices:

- Frontend built with Streamlit ensures rapid development and a responsive UI.
- Backend logic is modularized for maintainability (`explore_data.py`, `visualisation.py`, etc.).
- GPT-4o is used in a Retrieval-Augmented Generation (RAG) loop to interpret natural language and convert it into executable code.
- Data pipelines combine live and static CSVs for real-time + historical analysis.

Scalability:

- Supports multiple merchants.
- Real-time simulation uses thread locking to support streaming.
- Could be containerized and deployed on cloud (e.g. Streamlit Cloud, GCP, AWS).

Performance Considerations:

- Efficient use of `pandas` operations.
- Minimal API latency with prompt engineering and caching.
- Graph generation is optimized with `matplotlib` and `seaborn`.

✅ Integration and Real-World Deployment

Integration Capabilities:

- **OpenAI API**: To convert merchant queries into actionable Python scripts.
- **Twilio API**: Sends SMS alerts to merchants for critical insights.
- **PDF/PPTX Generators**: Provides downloadable reports for offline analysis.

Real-World Deployment Readiness:

- Can be deployed as a SaaS for small-medium F&B outlets.
- Modular enough to be integrated with backend APIs or DBs.
- Simulates real-time operations based on fixed time windows.

Limitations & Workarounds:

- Currently file-based (CSV) — future work could replace with SQL or Firebase.
- API keys and merchant passwords are hardcoded (to be secured via `.env`).

🎯 2. Prototype Quality and Functionality (30%)

✅ UI/UX Evaluation

User Interface:

- Clean and intuitive, customized with CSS and Streamlit components.
- Welcome modals, merchant-specific dashboards, and dynamic feedback loops.
- Chat interface is styled with custom avatars, timestamps, and conditional graphs.

User Experience:

- Prompt-driven exploration makes the system accessible even to non-technical users.
- Automatic summary, multi-turn follow-ups, and real-time response tracking.
- Report export features enhance merchant decision-making.

✅ Scalability & Real-World Application

Scalability Design:

- Time-based simulations mimic real operational data.
- CSV locks prevent concurrency issues during real-time updates.
- Modular code allows multi-merchant expansion.

Future-Proofing:

- Can scale with cloud-hosted databases.
- Easy to integrate with POS APIs or mobile apps.
- Structured logging and real-time flagging (e.g. late deliveries) ready for expansion.

🧪 3. Test Cases and Deployment Plan

Test Coverage:

- Manual validation of visualizations for different time filters.
- Chatbot tested with multiple types of merchant questions (trend analysis, product popularity, revenue breakdown).
- PDF and PPTX report generation tested with different chat histories.

Deployment Plan:

1. Host on Streamlit Cloud or deploy to GCP (Cloud Run).
2. Secure environment variables using `.env` and `python-dotenv`.
3. Store merchant and transaction data in a database (e.g. Firebase or PostgreSQL).
4. Add authentication flow for multi-user access.
5. Use logging and monitoring (e.g. Streamlit telemetry, Sentry).

🗒️ Conclusion

The Merchant Insights Assistant combines AI-powered language understanding, real-time data simulation, and intuitive visual reporting into a single platform aimed at enhancing operational awareness for food delivery merchants. With modular architecture, seamless integration of external services, and scalable design principles, the solution is not only a functional prototype but a viable product blueprint. Future iterations may integrate with real-time databases and enterprise dashboards, but even in its current form, the assistant delivers actionable intelligence to everyday business owners with just a question.