**CUSTOMER ANALYTICS DASHBOARD PROJECT**

**INDEX:**

* **Brief Introduction**
* **Why Open api was used**

**INTRODUCTION:**

**📝Project Overview**

This project is a **customer experience analytics dashboard** that helps analyze customer feedback and visualize insights.

* Collects customer feedback (CSV → Database).
* Performs **Sentiment Analysis** (Positive / Neutral / Negative) using **scikit-learn**.
* Generates **AI-powered summaries** using the **OpenAI API**.
* Displays results on a simple **interactive dashboard** (HTML + JavaScript + Chart.js).

This project shows how **AI + Data Science + Web Development** can come together to improve customer satisfaction monitoring.

**⚙️ Tech Stack**

* **Backend**: FastAPI (Python)
* **Database**: MySQL
* **Machine Learning**: Pandas, NumPy, scikit-learn, Pickle
* **Generative AI**: OpenAI API (text summarization)
* **Frontend**: HTML, CSS, JavaScript, Chart.js
* **Version Control**: Git/GitHub

**🔑 Features**

1. **Feedback Storage**
   * Upload CSV feedback data → stored in MySQL.
2. **Sentiment Analysis**
   * Logistic Regression model (scikit-learn) classifies feedback.
3. **Dashboard Visualization**
   * Sentiment distribution (pie chart).
   * Sentiment trend over time (line chart).
4. **AI-Powered Summary**
   * Generates monthly insights using OpenAI API.

**🏗️ Architecture**

CSV Feedback → MySQL Database → FastAPI Backend →

→ Sentiment Model (scikit-learn)

→ AI Summarizer (OpenAI API)

→ Frontend (HTML + Chart.js) → Dashboard

**🚀 How to Run**

**1. Clone the repo**

git clone https://github.com/yourusername/cx-analytics-dashboard.git

cd cx-analytics-dashboard

**2. Setup environment**

pip install -r requirements.txt

**3. Configure database**

* Install **MySQL**.
* Create database:

CREATE DATABASE feedback\_db;

* Update config.py with your DB username/password.

**4. Train model**

python train\_model.py

* This trains Logistic Regression sentiment classifier.
* Model is saved as sentiment\_model.pkl.

**5. Run backend**

uvicorn app.main:app --reload

* API docs: <http://localhost:8000/docs>

**6. Open frontend**

* Open index.html in browser → dashboard loads charts + summary.

**📊 Dashboard Preview**

* **Pie Chart**: Positive vs Neutral vs Negative feedback.
* **Line Chart**: Sentiment trend over time.
* **AI Summary**: Key issues highlighted by customers.

*(Add screenshots here once ready)*

**🔬 Example Interview Explanation**

*“My project was an AI-powered dashboard that analyzed Vodafone customer feedback. I built the backend with FastAPI, stored data in MySQL, and applied machine learning (Logistic Regression) to classify sentiment. The frontend (Chart.js) visualized customer sentiment trends. Additionally, I integrated the OpenAI API to summarize customer complaints into key insights. This gave Vodafone a clear picture of customer satisfaction at scale.”*

**📚 Skills Demonstrated**

* Programming: Python, HTML, CSS, JavaScript
* Tools: FastAPI, MySQL, scikit-learn, OpenAI API, Chart.js
* ML/AI: Sentiment analysis, Text summarization
* Software Engineering: Database integration, API development, Dashboard design

**WHY OPEN API WAS USED:**

**📌 OpenAI Integration in Customer Experience Analytics Dashboard**

This project uses **OpenAI’s GPT API** to automatically summarize customer feedback into short, meaningful sentences. The integration adds a layer of **natural language understanding** beyond traditional ML models.

**🚀 Workflow of OpenAI in the Project**

**1. Customer Feedback Submission**

* A user (or dataset row) provides feedback, e.g.  
  *“The service was very poor and disappointing.”*
* This feedback is first **preprocessed** and passed into our **Logistic Regression model + TF-IDF vectorizer** to classify sentiment as **Positive / Negative / Neutral**.

**2. Calling OpenAI API**

* Once sentiment is predicted, the raw feedback is also passed to **OpenAI GPT**.
* We send a request like this:

response = openai.Completion.create(

engine="text-davinci-003",

prompt=f"Summarize this customer feedback in one line: {feedback.text}",

max\_tokens=50,

)

summary = response.choices[0].text.strip()

**3. Why OpenAI is Used**

* **Problem:** Customer feedback is often long, repetitive, or unstructured.
* **Solution:** GPT creates **concise summaries**, e.g.:
  + Input: *“The service was very poor and disappointing.”*
  + GPT Output: *“Customer is unhappy with poor service experience.”*

This makes it easy for dashboards or management reports to quickly grasp the core issue.

**4. Backend Workflow**

1. API receives customer feedback (/analyze/).
2. Preprocess → Predict sentiment (ML model).
3. Send feedback → OpenAI GPT → Get summary.
4. Return response JSON:

{

"feedback": "The service was very poor and disappointing.",

"sentiment": "Negative",

"summary": "Customer is unhappy with poor service experience."

}

**5. Environment & Security**

* API key is stored in **environment variables** (OPENAI\_API\_KEY) → never hardcoded.
* Example:
* setx OPENAI\_API\_KEY "sk-proj-xxxxx"
* Loaded in Python:
* import os, openai
* openai.api\_key = os.getenv("OPENAI\_API\_KEY")

**6. Value Addition for Business**

* Sentiment model tells **what the customer feels**.
* OpenAI summary tells **why they feel that way** in plain English.
* Together, this enables **faster decision-making** for telecom service teams (e.g., Vodafone UK scenario).

**📝 Talking Points for Interview**

* Integrated **traditional ML (Logistic Regression)** + **state-of-the-art LLM (OpenAI GPT)**.
* Demonstrated how LLMs **enhance structured ML outputs** with contextual summaries.
* Secured credentials via **environment variables**.
* Designed for scalability → multiple feedback entries processed via API endpoints.

Would you like me to also make a **diagram (workflow illustration)** that you can show in interviews (API request → ML model → OpenAI → response)?