**📌 Project Overview: Automated Quality Monitoring System**

**🎯 Objective**

To provide a user-friendly dashboard that enables automatic analysis of data quality in uploaded CSV files using **Gradio** for the interface and **Great Expectations** for data validation.

**✅ Current Functionality**

Your system includes the following features:

**1. User Interface (Gradio-based)**

* Users can upload CSV files via a web interface.
* An “Analyze Data Quality” button triggers analysis.
* Results are displayed as:
  + **Summary statistics** (e.g., total rows, missing values)
  + **Validation results** (checks from Great Expectations)
  + **Missing values visualization** (bar chart)

**2. Data Quality Checks**

Implemented via Great Expectations:

* **Non-null values** check for all columns.
* **Uniqueness check** on the first column (assumed as ID).
* **Row count** check (between 1 and 1,000,000 rows).
* **Duplicate row** detection (handled manually).

**3. Technical Stack**

* **Backend:** Python with Pandas, Great Expectations
* **Frontend:** Gradio Blocks UI
* **Visualization:** Matplotlib & Seaborn
* **Port handling:** Dynamically finds an available port from 7860–7960

**🧪 Usage Instructions**

As provided in README.md:

bash

CopyEdit

pip install -r requirements.txt

python data\_quality\_dashboard.py

**🔧 Potential Enhancements (Next Development Phases)**

**📈 Functional Improvements**

* **Support for Excel files (.xlsx)** in addition to CSV.
* **Column-type validation** (e.g., check if numeric columns are really numeric).
* **Threshold alerts** (e.g., if >20% missing values → flag as critical).
* **Automated anomaly detection** using statistical profiling or ML.

**💬 UI/UX Enhancements**

* Display **interactive visualizations** (e.g., missing heatmaps via Plotly).
* Add a **progress bar** for large datasets.
* Enable **downloadable reports** (PDF/HTML/JSON) of analysis results.

**🔐 Security & Logging**

* Secure file uploads (size/type restrictions).
* Better **exception handling** and user-friendly error messages.
* Centralized **logging output** to a file or cloud logging system.

**☁️ Deployment Options**

* Dockerize the application for deployment on cloud (e.g., AWS/GCP).
* Enable remote sharing (share=True in Gradio) for collaborative quality reviews.

**📂 Pipeline Integration**

* Add API endpoints to **trigger via scripts or CI/CD pipelines**.
* Integrate with **data versioning tools** like DVC or LakeFS for reproducibility.

**🗃️ Dependencies**

From requirements.txt:

txt

CopyEdit

gradio==4.19.2

pandas==2.2.1

great-expectations==0.18.10

matplotlib==3.8.3

seaborn==0.13.2

These provide the foundation for UI, data analysis, validation, and visualization.