**Overview:**

The system starts with **data ingestion**, where users can upload Excel or CSV files directly, or data can come through an ingestion pipeline. This data is temporarily stored either in memory using Pandas or in a designated data folder.

Once the data is ingested, it goes through a **preprocessing** step to clean, normalize, and extract patterns from both historical and current records.

The **Flask backend** acts as the API layer, offering endpoints that allow users to upload data, trigger anomaly detection, classify breaks, retrieve results along with classification details, and get resolution suggestions powered by Agentic AI. It also collects feedback from users (reconcilers) on the results.

The **ML layer** is responsible for the core logic. It detects anomalies using a combination of rule-based checks (like threshold breaches or pattern deviations) and machine learning models such as Isolation Forest or AutoEncoders. For classifying the type of anomaly or "break", the system uses a classification model that maps issues to predefined categories. If the system encounters a new pattern, it tags it as “New” and explains the reasoning using an LLM like OpenAI’s GPT.

An **Agentic AI layer** (using tools like LangChain and OpenAI) summarizes detected anomalies and suggests possible resolution actions. It uses structured prompts and metadata from the anomalies to generate context-aware responses.

The **user interface** is built using Streamlit or lightweight Flask templates. It provides a dashboard where users can view the list of detected anomalies, understand the reasons behind them, and see suggested actions. There's also an option to flag any incorrect results (false positives or negatives).

Finally, a **feedback loop** allows reconcilers to manually submit their input. This feedback is captured and used to continuously improve the accuracy and performance of the ML models over time.