Smarter Reconciliation and Anomaly Detection using Gen AI(GROK): Architectural Diagram, Overview, and Solutioning Notes

[Mahesh Srivatsavi]

March 25, 2025

1 Architectural Diagram

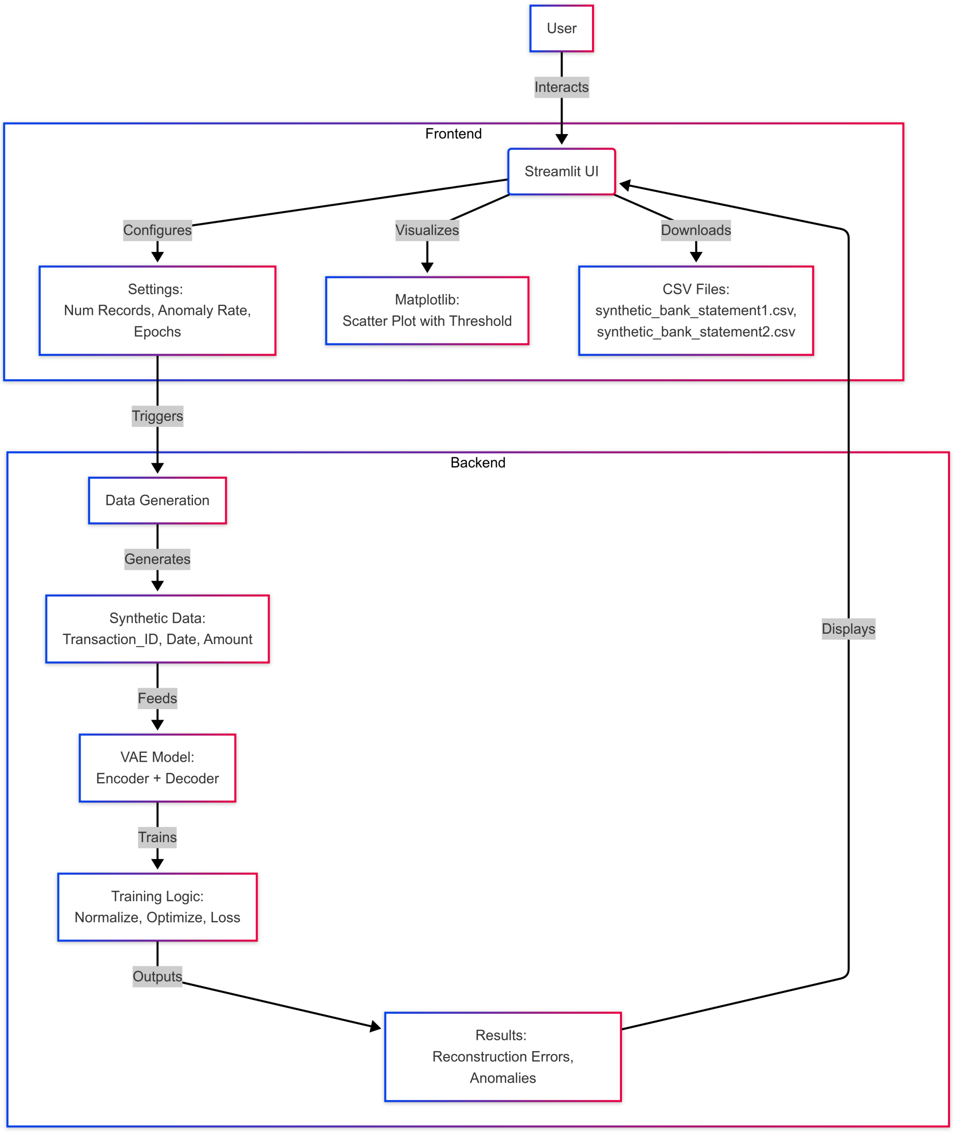

Figure 1: Architecture of the Anomaly Detection System with VAE

2 Overview

Title: Anomaly Detection System with Variational Autoencoder (VAE)

This program implements an anomaly detection system for synthetic financial transaction data using a Variational Autoencoder (VAE). The system is built with a modular architecture, separating data generation, model training, and user interaction. Key components include:

- **Data Generation**: Synthetic transaction data is generated with controlled anomalies to simulate real-world scenarios.
- **VAE Model**: A PyTorch-based VAE learns the latent representation of the data and identifies anomalies based on reconstruction errors.
- **Streamlit UI**: An interactive frontend allows users to configure parameters, visualize results, and download datasets.
- **Visualization**: Matplotlib is used to plot reconstruction errors, highlighting anomalies with a threshold.
- **Output**: The system outputs detected anomalies and allows downloading of synthetic datasets as CSV files.

The architecture is divided into two layers:

- **Frontend**: Handles user interaction, visualization, and data downloads via Streamlit.
- **Backend**: Manages data generation, VAE training, and anomaly detection.

3 Solutioning Notes

1. **Modularity**:
   - The program separates concerns into distinct functions: `generate_synthetic_data` for data creation, `VAE` class for the model, `train_vae` for training, and `main` for the UI.
   - This design allows easy extension, such as adding new anomaly detection algorithms or data sources.
2. **Scalability**:
   - The VAE model can scale to larger datasets by adjusting `input_dim`, `hidden_dim`, and `latent_dim`.
   - Training epochs and anomaly rates are configurable via the UI, enabling experimentation with different dataset sizes and complexities.
3. **Interactivity**:
   - Streamlit provides sliders for `num_records`, `anomaly_rate`, and `epochs`, allowing real-time parameter tuning.
   - A button triggers anomaly detection, and results are displayed immediately, enhancing user experience.
4. **Debugging and Validation**:
   - Debug outputs (e.g., number of anomalies detected, threshold value) are included to help tune the model.
   - The reconstruction error plot visually validates the anomaly detection process by showing the threshold and flagged anomalies.
5. **Potential Improvements**:
   - **Data Sources**: Extend the system to support real-world datasets (e.g., CSV uploads) instead of synthetic data.

- **Model Enhancements**: Incorporate additional anomaly detection methods (e.g., Isolation Forest from scikit-learn) as a fallback.
- **Threshold Tuning**: Allow users to adjust the anomaly detection threshold dynamically via the UI.
- **Performance**: Optimize VAE training for larger datasets by using mini-batches or GPU acceleration.