AI Anomaly Detection System - Implementation and Sprint Plan

Version: 1.0
Last Updated: March 12, 2025

1. Implementation Overview

1.1 Front End

- Set up SignalR client
- Build UI with ReCharts for anomaly visualization
- Add user feedback buttons (true/false for anomalies)
- Send feedback to API Gateway

1.2 API Gateway

- Set up SignalR server
- Create endpoint to receive new data
- Push data to the data queue
- Create endpoint to receive user feedback
- Push feedback to the feedback queue
- Create endpoint to receive notifications from the AI Service
- Send anomaly notifications to the front end

1.3 AI Service

- Set up FastAPI service
- Get data from data queue
- Implement Autoencoder-based anomaly detection for data
- Call API Gateway to notify of detected anomalies
- Create endpoint to allow model updates from the AI Retrainer

1.4 AI Retrainer

- Get feedback from feedback queue
- Periodically retrain the model when confidence is low
- Deploy updated model to the AI Service

1.5 Infrastructure

- Create Azure resources and GitHub workflows

---

2. Sprint Plan

Sprint	Focus
Sprint 1	Core Infrastructure, API Gateway, Front End Anomaly Receipt
Sprint 2	AI Service and Anomaly Detection
Sprint 3	Front End Anomaly Display and User Feedback
Sprint 4	Model Retraining and Feedback Loop

2.1 Sprint 1 - Core Infrastructure and API Gateway

Goal: Set up the Azure infrastructure and basic communication channels (Event Hub, API Gateway, SignalR).

Work Items:

- Azure
    - Create subscription, resource group, event hub namespace, keyvault, etc.
    - Create data and feedback event hubs
    - Create web apps for the gateway and the front end
- API Gateway
    - Set up a SignalR hub
    - Implement /data endpoint to receive new data and push it to the data queue
    - Implement /feedback endpoint to receive user feedback and push it to the feedback queue
    - Implement /anomaly endpoint to receive notifications from the AI Service and push them to SignalR
    - Set up JWT auth endpoint, secure the API and SignalR connections
- React front end
    - Set up a simple SignalR client
- Deployments
    - Create GitHub CI/CD workflows

2.2 Sprint 2 - AI Service and Anomaly Detection

Goal: Set up the AI Service and get the pipeline going from data ingestion to anomaly detection

Work Items:

- AI Service
    - Implement FastAPI-based REST API
    - Implement data queue consumption
    - Implement Autoencoder-based anomaly detection
    - Call /anomaly endpoint to send detected anomalies
- API Gateway
    - Handle anomaly notifications from the AI and push them to the front end

Notes:

- AI Service needs data in the queue to test properly. Write a simple script to generate test data and push it to the queue.
- Anomaly detections need to be visualized, but the front end isn't ready yet. Log results in the front end instead.

2.3 Sprint 3 - Front End Display and User Feedback

Goal: Implement the UI to visualize detected anomalies and allow user feedback.

Work Items:

- React Front End
    - Set up SignalR connection to listen for anomaly notifications.
    - Display anomaly detection results using ReCharts
    - Add UI Elements for user feedback buttons (true/false)
    - Send user feedback to the API Gateway

Notes:

- The front end needs real-time data. Possible enhancement of test data generator to keep data coming in.
- Model retraining isn't implemented yet, so feedback won't do anything.

2.4 Sprint 4 - Model Retraining and Feedback Loop

Goal: Close the loop with feedback-based model improvements.

Work Items:

- AI Service
    - Implement /update endpoint for model updates
- AI Model Retraining
    - Consume feedback from the feedback queue
    - Trigger model retraining when confidence is low
    - Deploy updated model to the AI Service