ABB Team 6

Sreemahi Gangadharan 220905251
Ayanika Paul 220905128
Aanya Shantharam 220953438
Ananthula Srinivya 220968404
MIT Manipal

Design Document

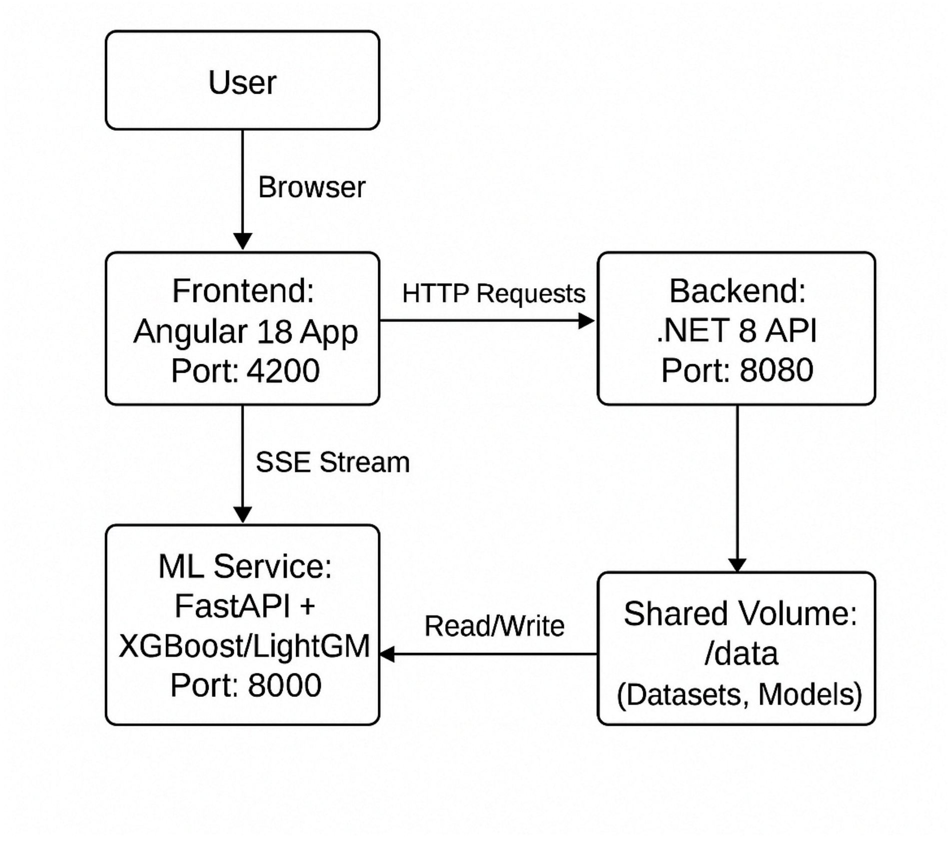
IntelliInspect System Design

This document outlines the architecture, data flow, and API specifications for IntelliInspect, a machine learning-based inspection simulation tool developed during a 48-hour hackathon. The system is built using Angular 18 (frontend), .NET 8 (backend API), Python 3.13 + FastAPI (ML service), and Docker Compose.

System Architecture Diagram

The system comprises three main services orchestrated via Docker Compose: a frontend Angular application, a .NET 8 backend API, and a Python FastAPI ML service. These services communicate over HTTP, with the backend acting as a proxy to the ML service and streaming simulation data via Server-Sent Events (SSE) to the frontend. A shared Docker volume (/data) stores datasets and models persistently.

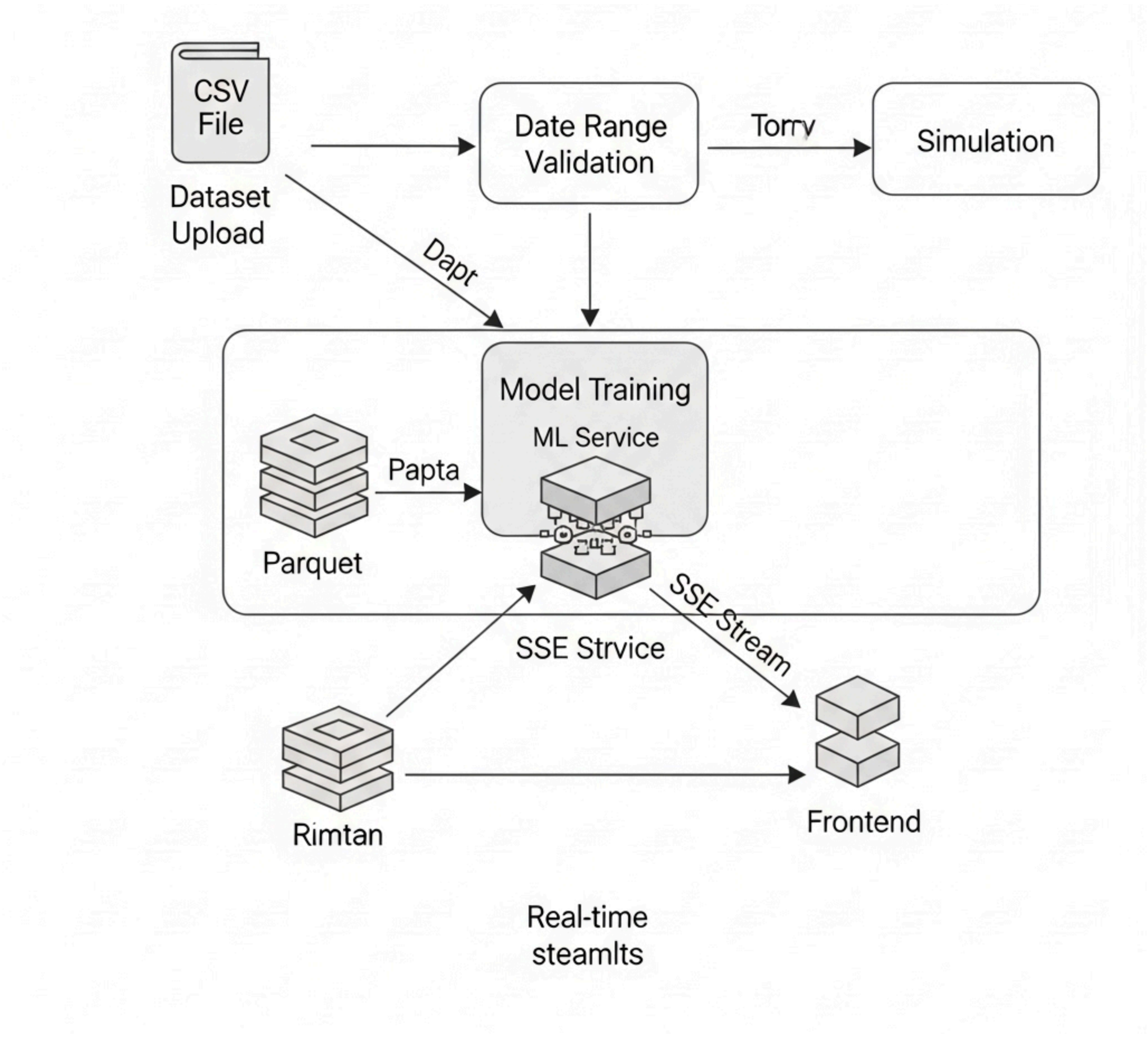
Below is a Mermaid diagram representing the high-level architecture:


User
Browser
Frontend: Angular 18 App Port: 4200
HTTP Requests
Backend: .NET 8 API Port: 8080
SSE Stream
ML Service: FastAPI + XGBoost/LightGM Port: 8000
Read/Write
Shared Volume: /data (Datasets, Models)

Data Flow Diagram

The data flow follows a four-step process: dataset upload, date range validation, model training, and simulation. Data is ingested as CSV, processed into Parquet for efficiency, and managed through the backend and ML service. Simulation results are streamed live to the frontend via SSE.

Below is a Mermaid diagram representing the data flow:



API Contract and Payload Structure

Backend (.NET)

- **POST /dataset/upload** (multipart/form-data: file)
    - **200 Response**: { filename, rows, cols, passRate, tsMin, tsMax }
        - filename: String, name of the uploaded file.
        - rows: Integer, total number of rows.
        - cols: Integer, total number of columns.
        - passRate: Float, percentage of rows where Response == 1.
        - tsMin: String (ISO datetime), minimum synthetic_timestamp.
        - tsMax: String (ISO datetime), maximum synthetic_timestamp.
- **POST /dates/validate**
    - **Body**: { trainStart, trainEnd, testStart, testEnd, simStart, simEnd }
        - trainStart: String (ISO datetime), start of training period.
        - trainEnd: String (ISO datetime), end of training period.
        - testStart: String (ISO datetime), start of testing period.
        - testEnd: String (ISO datetime), end of testing period.
        - simStart: String (ISO datetime), start of simulation period.
        - simEnd: String (ISO datetime), end of simulation period.
- **200 Response**: { status: "Valid"|"Invalid", counts:{train, test, sim}, durations:{train, test, sim}, monthly:[{month, train, test, sim}] }
        - status: String, validation result.
        - counts: Object with integer row counts for each period.
        - durations: Object with duration values for each period.
        - monthly: Array of objects with month and counts for train, test, sim.

POST /train-model

- **Body**: { trainStart, trainEnd, testStart, testEnd }
    - trainStart: String (ISO datetime), start of training period.
    - trainEnd: String (ISO datetime), end of training period.
    - testStart: String (ISO datetime), start of testing period.
    - testEnd: String (ISO datetime), end of testing period.

200 Response: { accuracy, precision, recall, f1, cm:{tn, fp, fn, tp}, loss:[...], acc:[...] }

- accuracy: Float, model accuracy.
- precision: Float, precision score.
- recall: Float, recall score.
- f1: Float, F1 score.
- cm: Object with confusion matrix values.
- loss: Array of floats, loss values.

- acc: Array of floats, accuracy values.

GET /simulate/stream (SSE)

- **Event data**: { ts, id, pred, conf, temperature, pressure, humidity, totals:{n, pass, fail, avgConf} }
    - ts: String (ISO datetime), row timestamp.
    - id: String or integer, row identifier.
    - pred: Integer (0 or 1), predicted value.
    - conf: Float (0-1), prediction confidence.
    - temperature: Float, temperature value.
    - pressure: Float, pressure value.
    - humidity: Float, humidity value.
    - totals: Object with counters.
- **Final event**: { done: true, totals:{n, pass, fail, avgConf} }

ML (FastAPI)

- **POST /train**
    - **Body**: { trainStart, trainEnd, testStart, testEnd }
        - trainStart: String (ISO datetime), start of training period.
        - trainEnd: String (ISO datetime), end of training period.
        - testStart: String (ISO datetime), start of testing period.
        - testEnd: String (ISO datetime), end of testing period.
    - **200 Response**: { accuracy, precision, recall, f1, cm, loss, acc }
        - accuracy: Float, model accuracy.
        - precision: Float, precision score.
        - recall: Float, recall score.
        - f1: Float, F1 score.
        - cm: Array or object, confusion matrix.
        - loss: Array of floats, loss values.
        - acc: Array of floats, accuracy values.

POST /predict

- **Body**: { features: { ...single row columns... } }
    - features: Object, key-value pairs of column values (excluding 'Response', 'synthetic_timestamp', 'id').
- **200 Response**: { pred, prob }
    - pred: Integer (0 or 1), predicted value.
    - prob: Float (0-1), prediction probability.