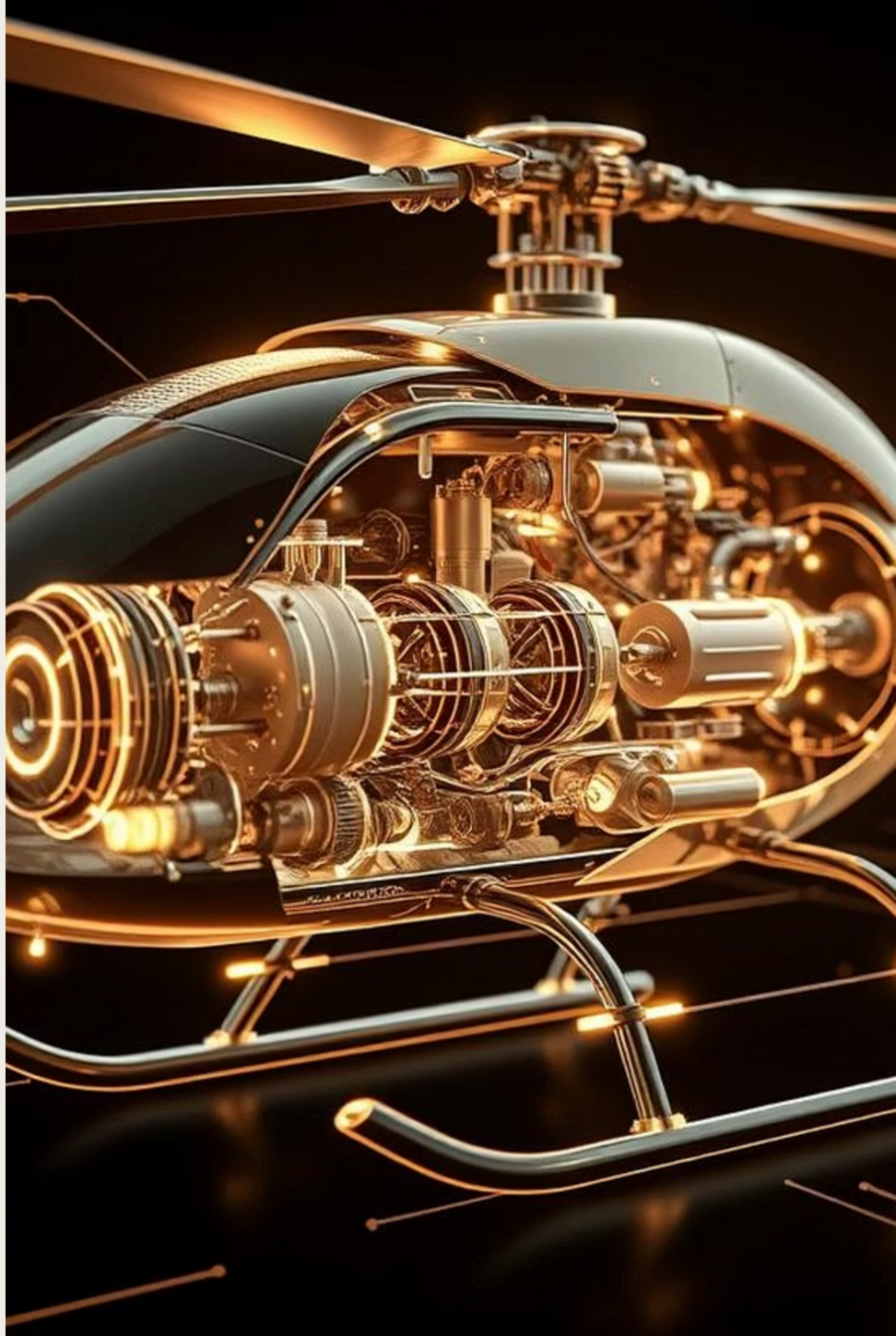
Digital Twin for Helicopter Fuel Injection System

Team Name : RotorX



Problem Statement

- Modern helicopters require real-time monitoring and predictive maintenance for safe and efficient operation. A key subsystem is the fuel injection system, which directly impacts engine performance and fuel economy.
- This project aims to develop a Digital Twin of the helicopter’s fuel injection system — a virtual, data-driven replica that simulates engine behavior, visualizes fuel flow, detects anomalies using AI, and enables “what-if” fault simulations.
- The system supports continuous monitoring, early fault detection, and provides engineers with diagnostic insights through an interactive dashboard.

Objectives

1 **Simulate Real-time Behavior**

Simulate real-time fuel injection system behavior (RPM, flow, pressure, temp)

2 **Stream and Visualize Telemetry**

Stream telemetry via MQTT and visualize in a dashboard

3 **Store and Retrieve Telemetry**

Store and retrieve telemetry using a database

4 **Detect Anomalies**

Detect anomalies using machine learning models

5 **Simulate Faults**

Simulate faults (e.g., clogging, pressure drop) and study impact

6 **Provide Diagnosis**

Provide diagnosis with probable causes when faults occur

7 **Enable Replay and Logging**

Enable replay, logging, and export of telemetry for analysis

Tech Stack

Frontend	React.ts + Vite + Tailwind CSS
Backend	FastAPI (Python)
MQTT	HiveMQ / Mosquitto
Database	MongoDB (via MongoDB Compass)
ML Models	Isolation Forest (joblib), Pandas
3D Viz	Three.js via @react-three/fiber
WebSocket	Socket.IO (optional)
Deployment	Vercel / Netlify + Render

Datasets

The system utilizes various datasets for simulation, fault injection, and model training.

Simulated Data	Real-time telemetry via Python
Fault Data	Injected using /simulate-fault
Model Training	Data logged in MongoDB → exported to CSV for ML training

Methodology

1. System Modeling & Simulation

Build a mathematical model of the helicopter's fuel injection system.

2. Data Transmission via IoT Protocol

Transmit simulated data continuously using MQTT protocol or WebSocket.

3. Backend Data Ingestion & Storage

Use a backend framework and Store timestamped data for logging.

4. Machine Learning-based Anomaly Detection

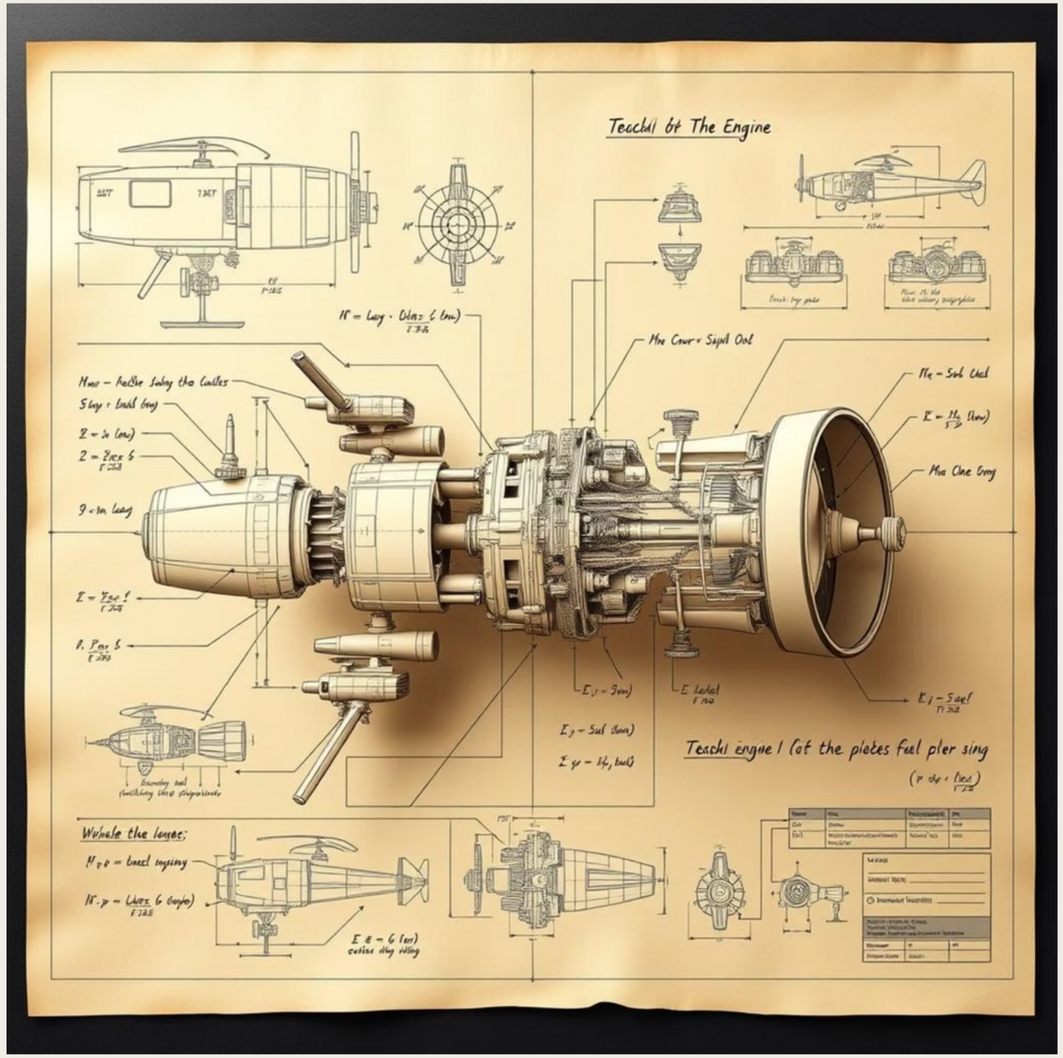
Train an unsupervised anomaly detection model

5. Fault Injection and Diagnosis

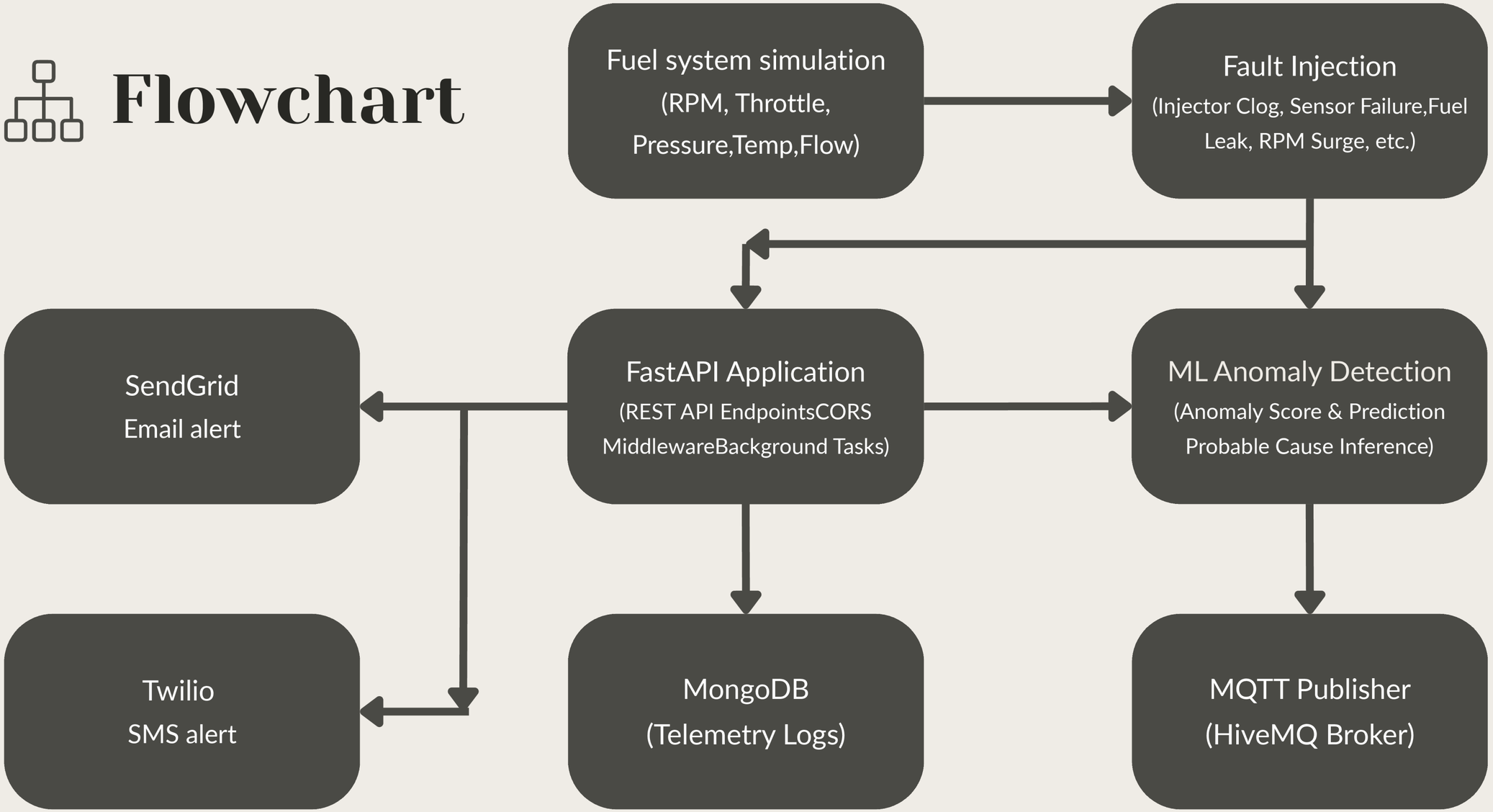
Analyze how these faults affect engine parameters.

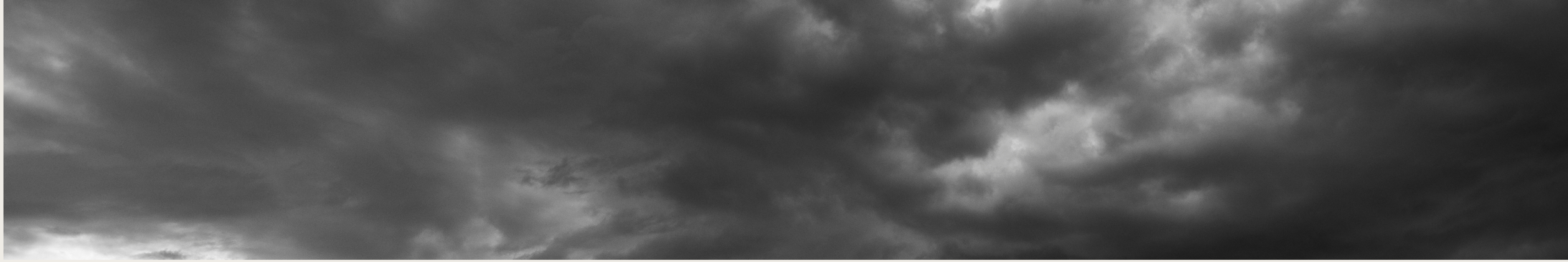
6. Data Visualization & Dashboard Interface

Design an interactive frontend dashboard Enable playback of past telemetry for review and analysis.



Flowchart


Fuel system simulation
(RPM, Throttle,
Pressure,Temp,Flow)
Fault Injection
(Injector Clog, Sensor Failure,Fuel
Leak, RPM Surge, etc.)
SendGrid
Email alert
FastAPI Application
(REST API EndpointsCORS
MiddlewareBackground Tasks)
ML Anomaly Detection
(Anomaly Score & Prediction
Probable Cause Inference)
Twilio
SMS alert
MongoDB
(Telemetry Logs)
MQTT Publisher
(HiveMQ Broker)



Data Flow Process

1. Simulate fuel system parameters (RPM, throttle, pressure, temp, flow)
2. Optional fault injection (clog, leak, surge, sensor failure)
3. ML anomaly detection using Isolation Forest model
4. Infer probable cause if anomaly detected
5. Store telemetry data in MongoDB with anomaly flag
6. Publish clean telemetry to MQTT broker (external systems)
7. Send email alerts via SendGrid for anomalies and SMS alerts via Twilio.
8. Export anomaly data as CSV/JSON for analysis
9. Continuous monitoring and model retraining capability

Key Features:

1. **Real-time fuel system simulation**

   The system includes a real-time simulation of the helicopter fuel system, providing dynamic data for monitoring and analysis.

2. **MQTT telemetry streaming**

   Telemetry data is streamed in real-time using MQTT, ensuring low-latency data transfer.

3. **MongoDB data logging**

   All telemetry and fault data are logged in MongoDB for historical analysis and model training.

4. **REST APIs for telemetry + faults**

   REST APIs are provided for seamless integration and access to telemetry and fault data.

5. **AI-based anomaly detection (Isolation Forest)**

   An AI-based anomaly detection system, utilizing Isolation Forest, identifies unusual patterns in the fuel system data.

6. **React dashboard with live charts**

   A user-friendly React dashboard displays live charts of the fuel system parameters, providing real-time insights.

Use Case: Aviation Maintenance Crew

The system is designed to empower Aviation Maintenance Crew with advanced tools for fuel system management.

Monitor helicopter fuel systems in real time

Provides a comprehensive overview of fuel system performance with live data.

Detect anomalies before failures occur

Proactive identification of potential issues, preventing costly downtime.

Understand root causes using AI

AI-driven insights help pinpoint the underlying reasons for anomalies.

Perform virtual fault tests to improve safety

Simulate fault scenarios to train and validate maintenance procedures without risk.

Replay and study failure scenarios from logs

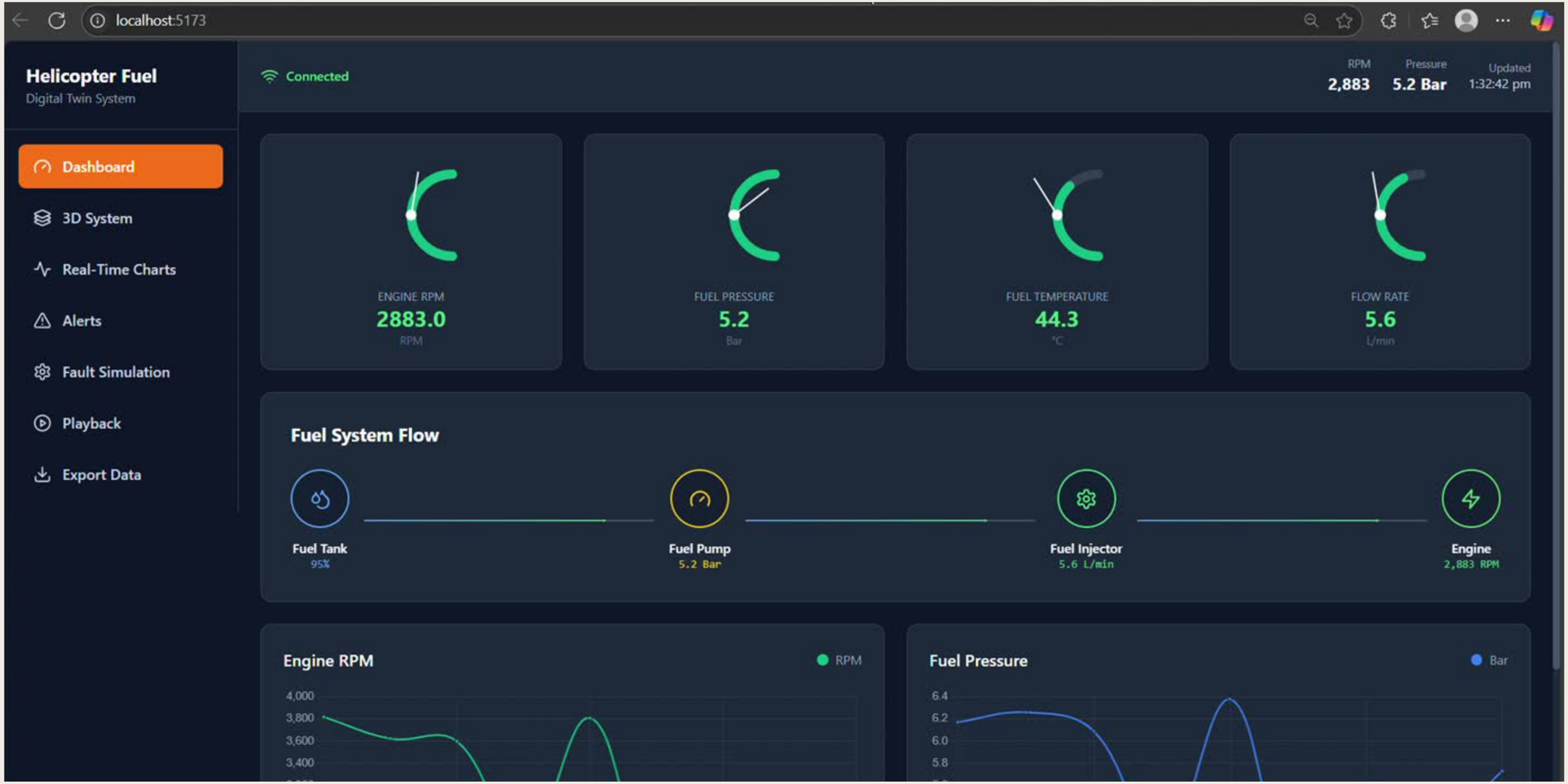
Analyze past incidents to learn and improve future maintenance strategies.

Specific Requirements

The system adheres to specific technical and performance requirements to ensure optimal operation.

Requirement	Details
Real-time data	1 second interval via MQTT
Backend inference latency	< 200 ms (simulate + predict)
MongoDB storage	Up to 1000 entries, rotated
Model compatibility	Must support joblib loading
Visuals	Live charts, optional 3D flow model

Prototype


localhost:5173
Helicopter Fuel
Digital Twin System
Connected
RPM
2,883
Pressure
5.2 Bar
Updated
1:32:42 pm
Dashboard
3D System
Real-Time Charts
Alerts
Fault Simulation
Playback
Export Data
ENGINE RPM
2883.0
RPM
FUEL PRESSURE
5.2
Bar
FUEL TEMPERATURE
44.3
°C
FLOW RATE
5.6
L/min
Fuel System Flow
Fuel Tank
95%
Fuel Pump
5.2 Bar
Fuel Injector
5.6 L/min
Engine
2,883 RPM
Engine RPM
RPM
4,000
3,800
3,600
3,400
Fuel Pressure
Bar
6.4
6.2
6.0
5.8

Thank You