AI-Based PMD **failure prediction**

TEAM

FAULT FORCE

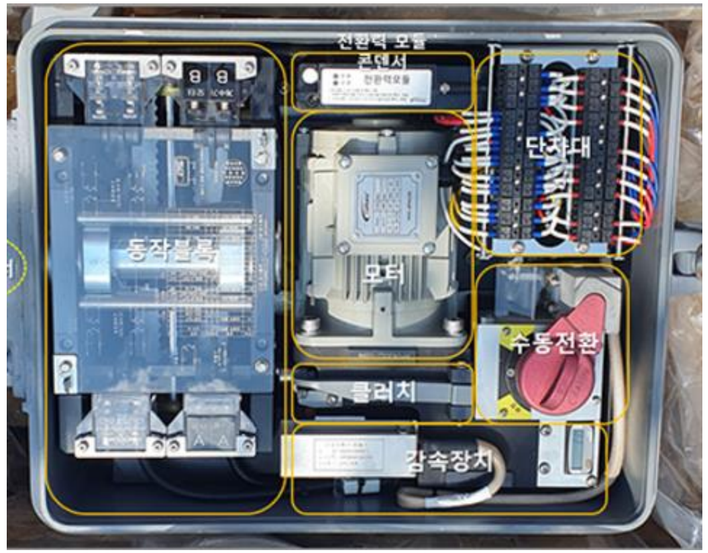
우송대학교
WOOSONG UNIVERSITY

PARTICIPATING COMPANY

Sehwa Co., Ltd

철도 선로전환기

Team Visit to Sehwa



우송대학교 Woosong university

AGENDA

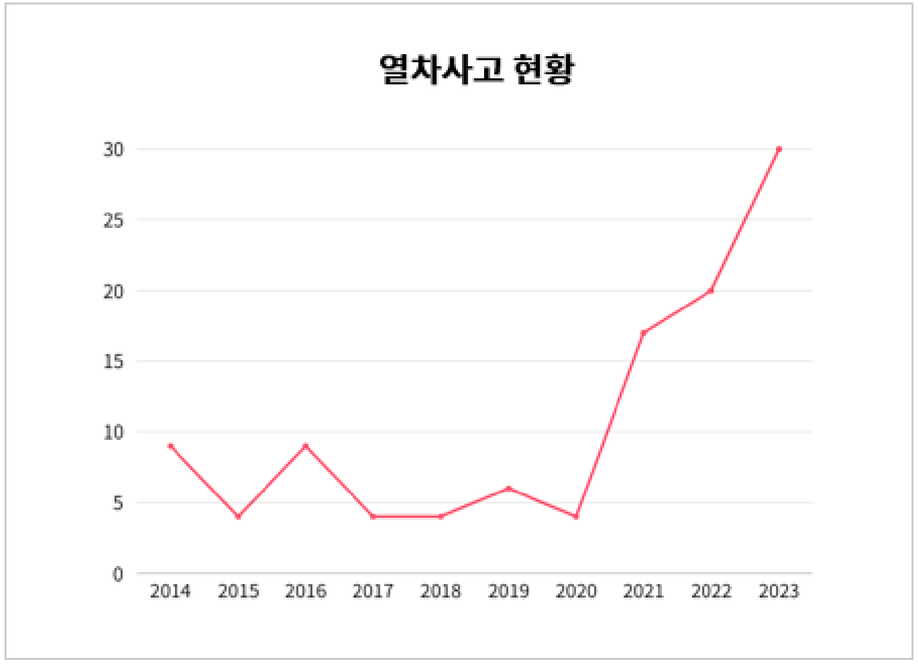
1. Motivation
2. Our Solution
3. AI Progress
4. Result
5. Demo
6. Future Plan

Motivation

Increase in train accident rates

- Train accidents surged
  from 4 to 30 in 2020~23

Current Status of Train Accidents on the Rise



출처 : 교통안전공단 철도안전정보포털(www.railsafety.or.kr)

The most accident is

- According to the analysis of railway signal control system failures from 2000 to the last 10 years, PMD failures accounted for 27%

Development of error prediction model

- Predicting the possibility of failure of the track converter to minimize train accidents

Our Solution

Sensor data

- Huge amount of Sensor data from Point Machine

↓

- Pre-processing unnecessary parts and storing them in the database

AI Capabilities

- Training AI models for error detection and prediction

↓

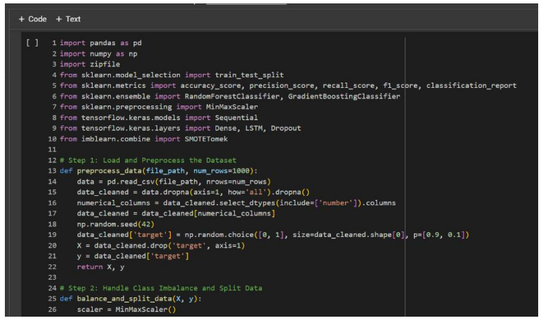
- Detect anomalies and predict potential failures with high accuracy

Our Solution

AI based Failure Prediction

AI

- AI powered error detection
- AI powered failure prediction

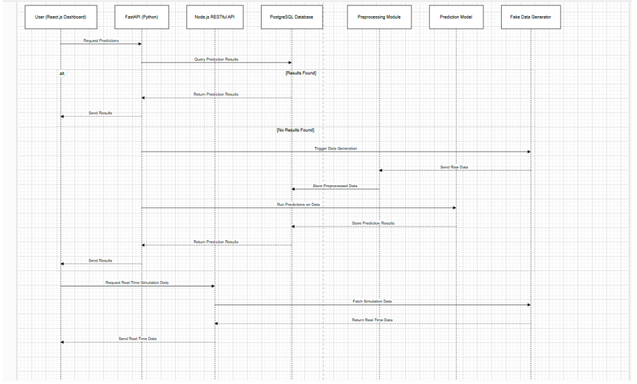


```
+ Code  + Text

[ ]  1 import pandas as pd
     2 import numpy as np
     3 import zipfile
     4 from sklearn.model_selection import train_test_split
     5 from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report
     6 from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
     7 from sklearn.preprocessing import MinMaxScaler
     8 from tensorflow.keras.models import Sequential
     9 from tensorflow.keras.layers import Dense, LSTM, Dropout
    10 from imblearn.combine import SMOTETomek
    11
    12 # Step 1: Load and Preprocess the Dataset
    13 def preprocess_data(file_path, num_rows=1000):
    14     data = pd.read_csv(file_path, nrows=num_rows)
    15     data_cleaned = data.dropna(axis=1, how='all').dropna()
    16     numerical_columns = data_cleaned.select_dtypes(include=['number']).columns
    17     data_cleaned = data_cleaned[numerical_columns]
    18     np.random.seed(42)
    19     data_cleaned['target'] = np.random.choice([0, 1], size=data_cleaned.shape[0], p=[0.9, 0.1])
    20     X = data_cleaned.drop('target', axis=1)
    21     y = data_cleaned['target']
    22     return X, y
    23
    24 # Step 2: Handle Class Imbalance and Split Data
    25 def balance_and_split_data(X, y):
    26     scaler = MinMaxScaler()
```

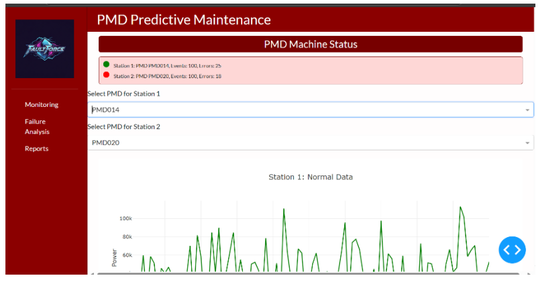
Backend

- Storing data
- Integrating AI models with Dashboard

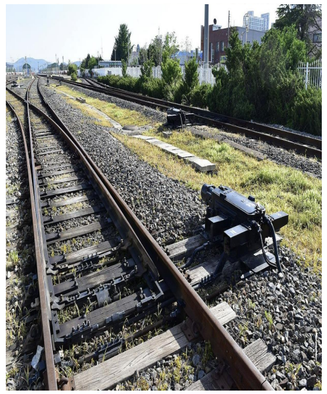


Frontend

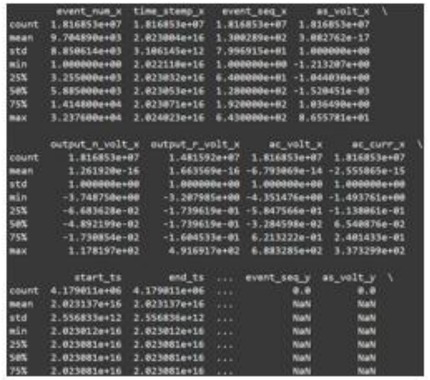
- Maintenance dashboard



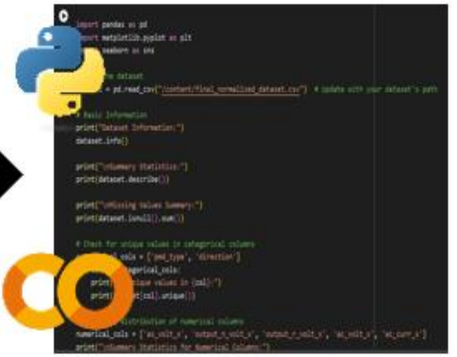
System Operation Process



Loaded the sensor data from PMD(선로전환기)

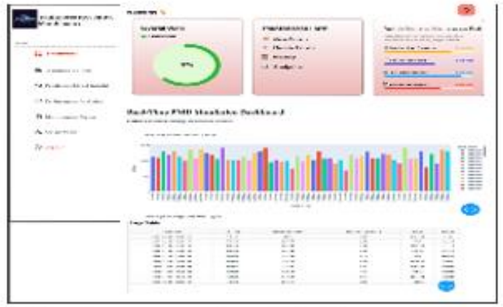


Cleaning and Preprocessing Sensor Data



AI model training for error detection and Prediction of PMD

Save PMD information



Display on the Maintenance Worker's Screen

HTTP Req (RESTful API)

HTTP Res (Restful API)

Backend Server

HTTP(Fast API, Flask)

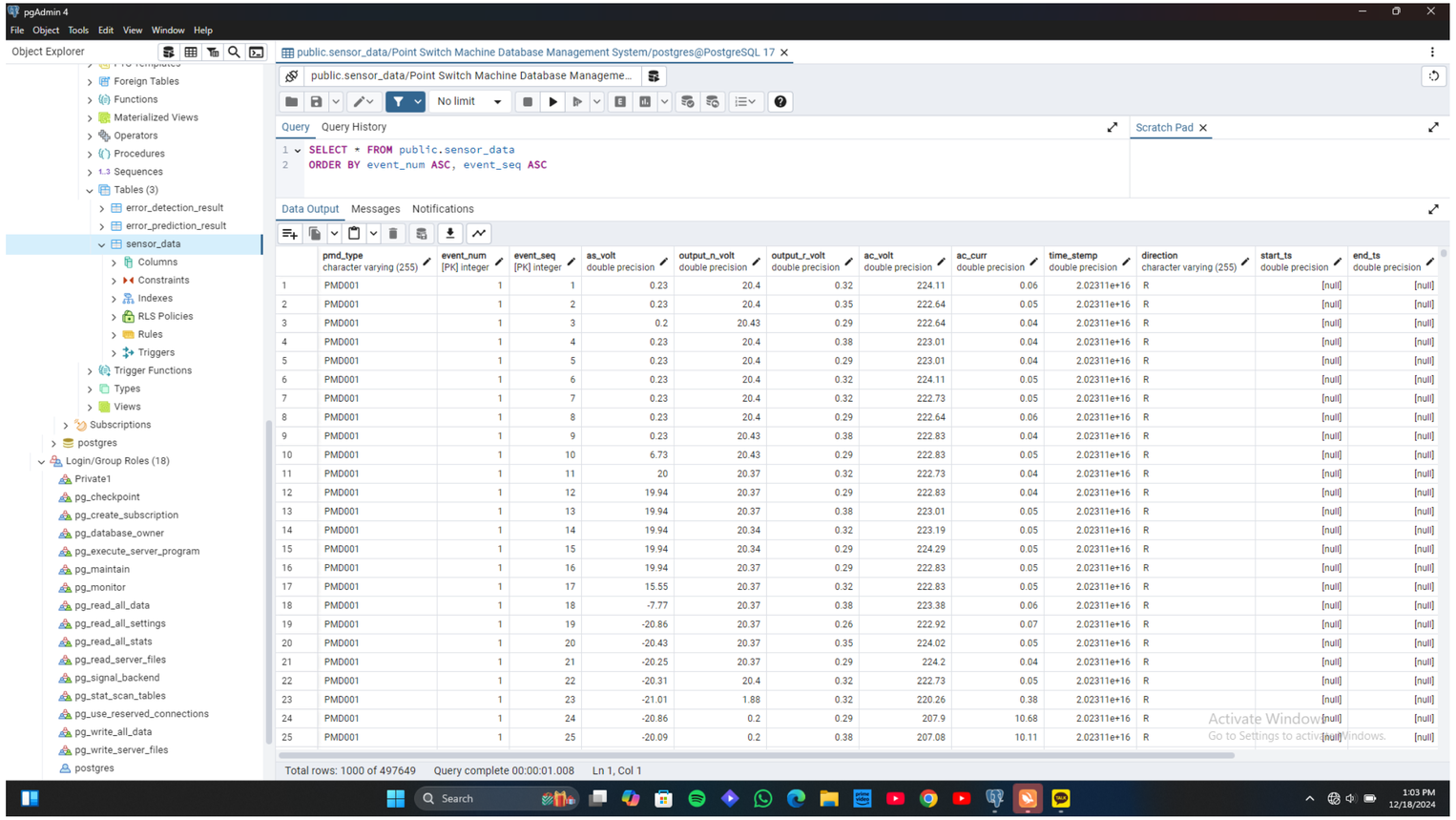
HTTP Res (FastAPI, Flask)

Point Switch Machine Database Management System

Stored sensor data from the point switch machine

Stored the predicted and detected data from the frontend

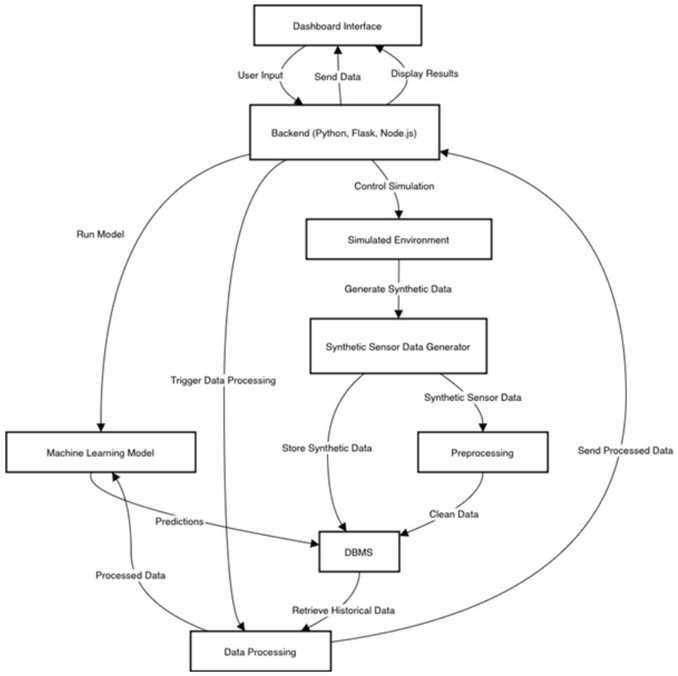
Supports real-time tracking and historical data analysis.



AI

- Process the sensor data
- Detect anomaly/abnormal data
- Predict the anomaly/abnormal data
- Capable of early fault detection.
- Store results in backend server.

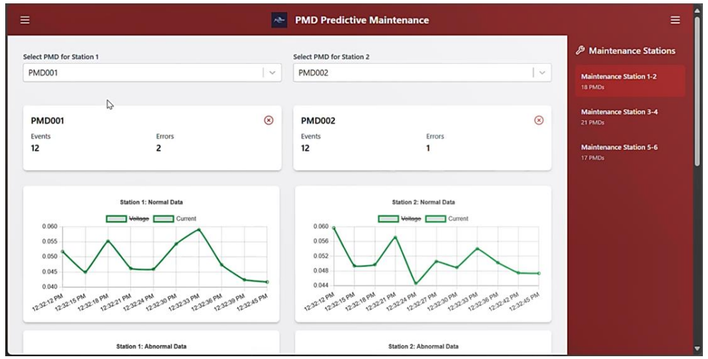
High Level Design



	event_num_x	time_stemp_x	event_seq_x	as_volt_x	\
count	1.816853e+07	1.816853e+07	1.816853e+07	1.816853e+07	
mean	9.704890e+03	2.023004e+16	1.300289e+02	3.082762e-17	
std	8.850614e+03	3.106145e+12	7.996915e+01	1.000000e+00	
min	1.000000e+00	2.022110e+16	1.000000e+00	-1.213207e+00	
25%	3.255000e+03	2.023032e+16	6.400000e+01	-1.044030e+00	
50%	5.885000e+03	2.023053e+16	1.280000e+02	-1.520451e-03	
75%	1.414800e+04	2.023071e+16	1.920000e+02	1.036490e+00	
max	3.237600e+04	2.024023e+16	6.430000e+02	8.655781e+01	

	output_n_volt_x	output_r_volt_x	ac_volt_x	ac_curr_x	\
count	1.816853e+07	1.481592e+07	1.816853e+07	1.816853e+07	
mean	1.261920e-16	1.663569e-16	-6.793069e-14	-2.555865e-15	
std	1.000000e+00	1.000000e+00	1.000000e+00	1.000000e+00	
min	-3.748750e+00	-3.207985e+00	-4.351476e+00	-1.493761e+00	
25%	-6.683628e-02	-1.739619e-01	-5.847566e-01	-1.138061e-01	
50%	-4.892199e-02	-1.739619e-01	-3.284598e-02	6.540876e-02	
75%	-1.730854e-02	-1.604533e-01	6.213222e-01	2.401433e-01	
max	1.178197e+02	4.916917e+02	6.883285e+02	3.373299e+02	

	start_ts	end_ts	...	event_seq_y	as_volt_y	\
count	4.179011e+06	4.179011e+06	...	0.0	0.0	
mean	2.023137e+16	2.023137e+16	...	NaN	NaN	
std	2.556833e+12	2.556836e+12	...	NaN	NaN	
min	2.023012e+16	2.023012e+16	...	NaN	NaN	
25%	2.023081e+16	2.023081e+16	...	NaN	NaN	
50%	2.023081e+16	2.023081e+16	...	NaN	NaN	
75%	2.023081e+16	2.023081e+16	...	NaN	NaN	



```python
# Step 3: Build LSTM Autoencoder
def build_lstm_autoencoder(input_dim):
    model = Sequential([
        LSTM(128, activation='relu', input_shape=(input_dim, 1), return_sequences=True),
        Dropout(0.3),
        LSTM(64, activation='relu', return_sequences=False),
        Dense(64, activation='relu'),
        Dense(input_dim, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='mse')
    return model
```

```python
# Random Forest Classifier
rf_model = RandomForestClassifier(random_state=42, n_jobs=-1)
rf_model.fit(X_train, y_train)
rf_preds = rf_model.predict_proba(X_test)[:, 1]
```

```python
# Deep Neural Network (DNN)
def build_dnn(input_dim):
    model = Sequential([
        Dense(256, activation='relu', input_dim=input_dim),
        Dropout(0.4),
        Dense(128, activation='relu'),
        Dropout(0.4),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

```
Metadata not available for detected anomalies.
Model Performance Metrics:
Accuracy: 0.96290
Precision: 0.98178
Recall: 0.94328
F1 Score: 0.96214

Classification Report:
              precision    recall  f1-score   support

           0    0.94545   0.98251   0.96362      1429
           1    0.98178   0.94328   0.96214      1428

    accuracy                        0.96290      2857
   macro avg    0.96362   0.96289   0.96288      2857
weighted avg    0.96361   0.96290   0.96288      2857


Abnormal Data Detected:
Number of Abnormal Data: 1372
Abnormal Event Numbers: [4, 8, 11, 13, 15, 16, 17, 19,
```

```python
# Step 2: Function to prepare sequence data
def create_sequences(event_nums, window_size=3):
```

```python
# Step 3: Prepare the data
window_size = 3  # Predict the next event based on the past 3 events
X, y = create_sequences(abnormal_event_numbers, window_size)
X = X.reshape((X.shape[0], X.shape[1], 1))  # Reshape to match LSTM input format
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

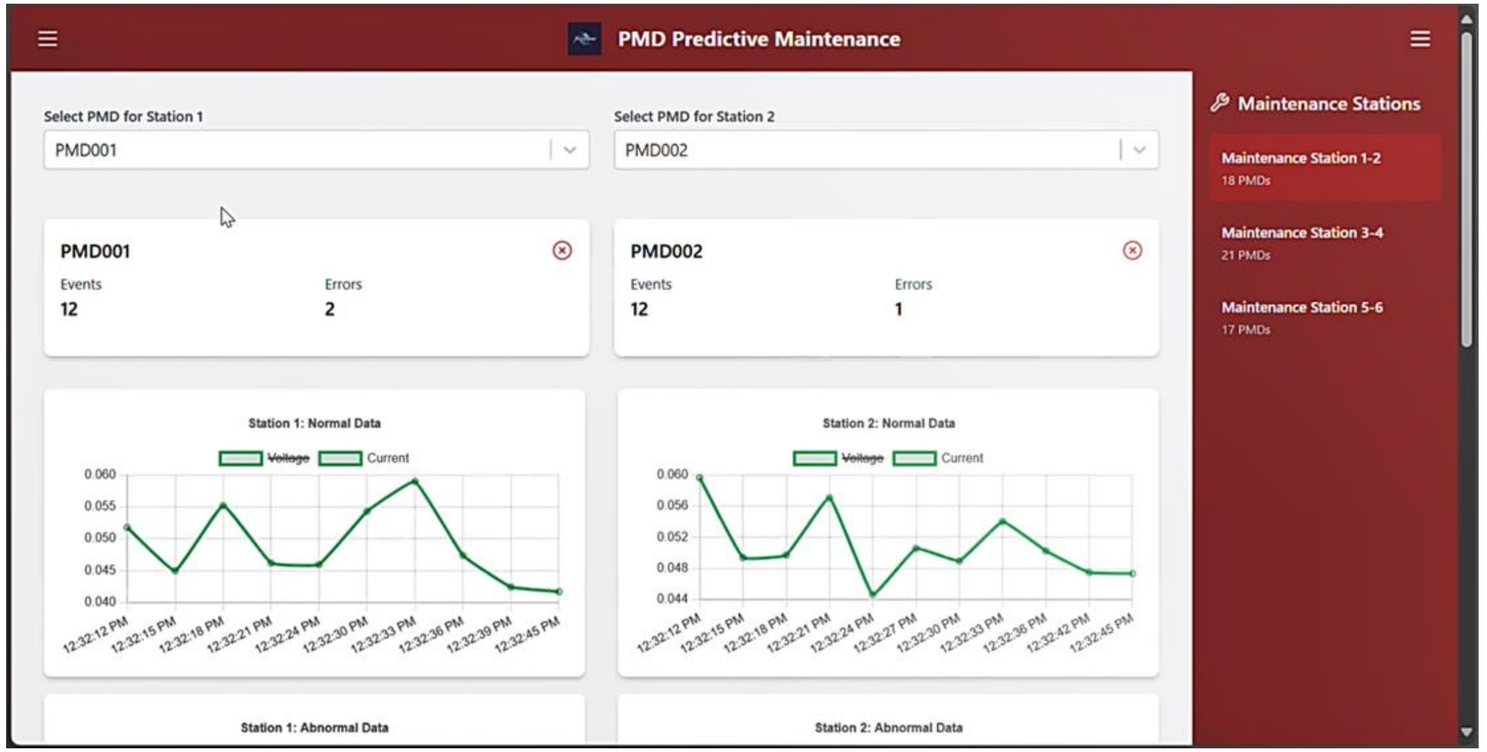
```
Top Predicted Error Events:
      event_num_x  error_prob
2281         2286    0.870000
1768         1773    0.870000
5105         5110    0.817952
2410         2415    0.802500
5397         5402    0.759667
```

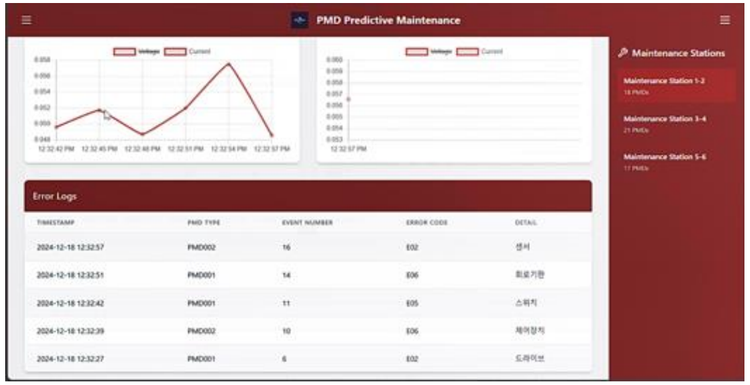
```python
# Step 4: Build the LSTM model
def build_lstm_model(input_shape):
    model = Sequential([
        LSTM(64, activation='relu', input_shape=input_shape),
        Dropout(0.2),
        Dense(32, activation='relu'),
        Dense(1)  # next event_num
    ])
    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
    return model
```

```python
# Create and train the model
model = build_lstm_model((X.shape[1], 1))
print("start the learning model...")
history = model.fit(X_train, y_train, epochs=50, batch_size=128,
```

Frontend

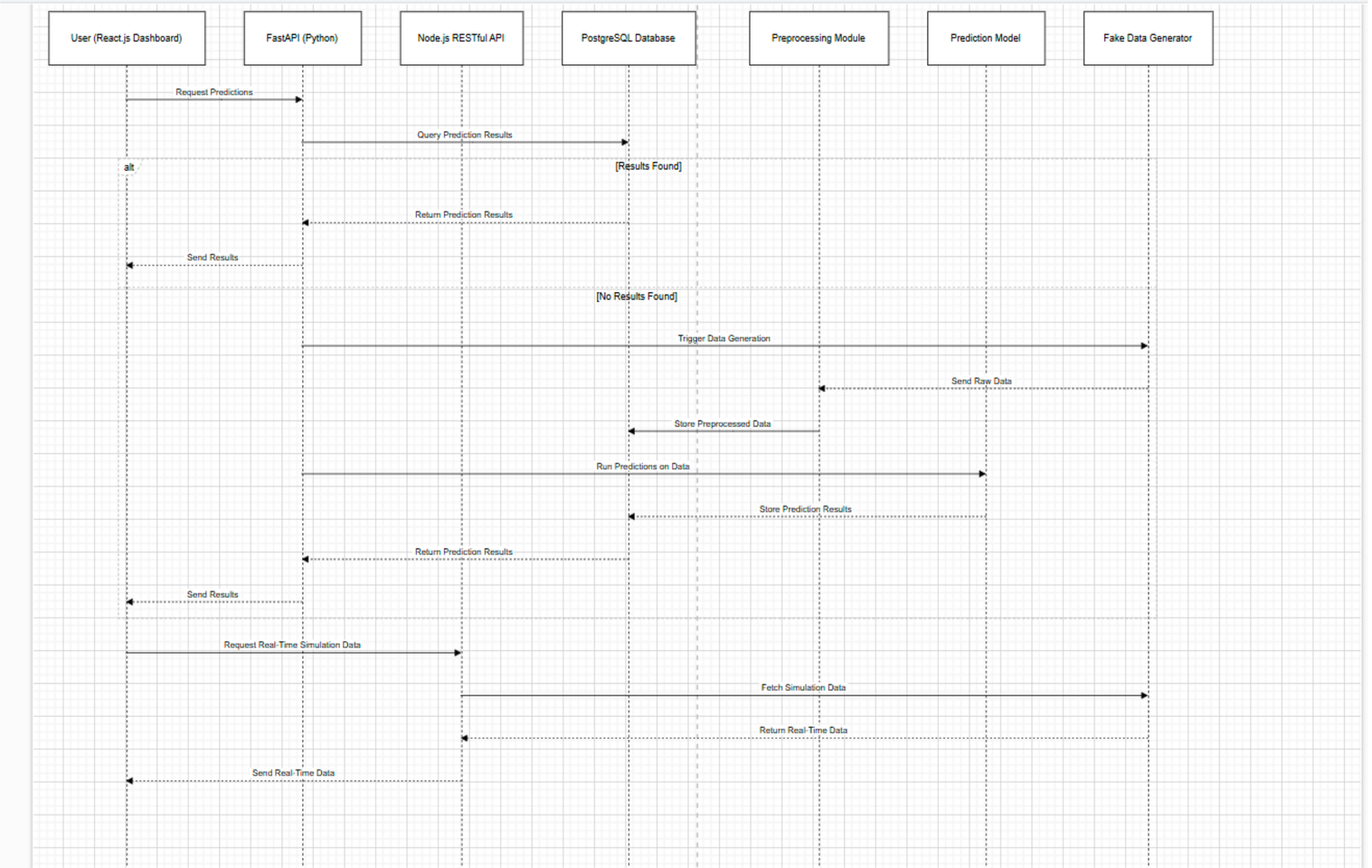
- User Interface
- Real-time monitoring
- Display detected errors.
- Alert maintenance worker about the error
- Visualizes predicted failure data and historical trends.



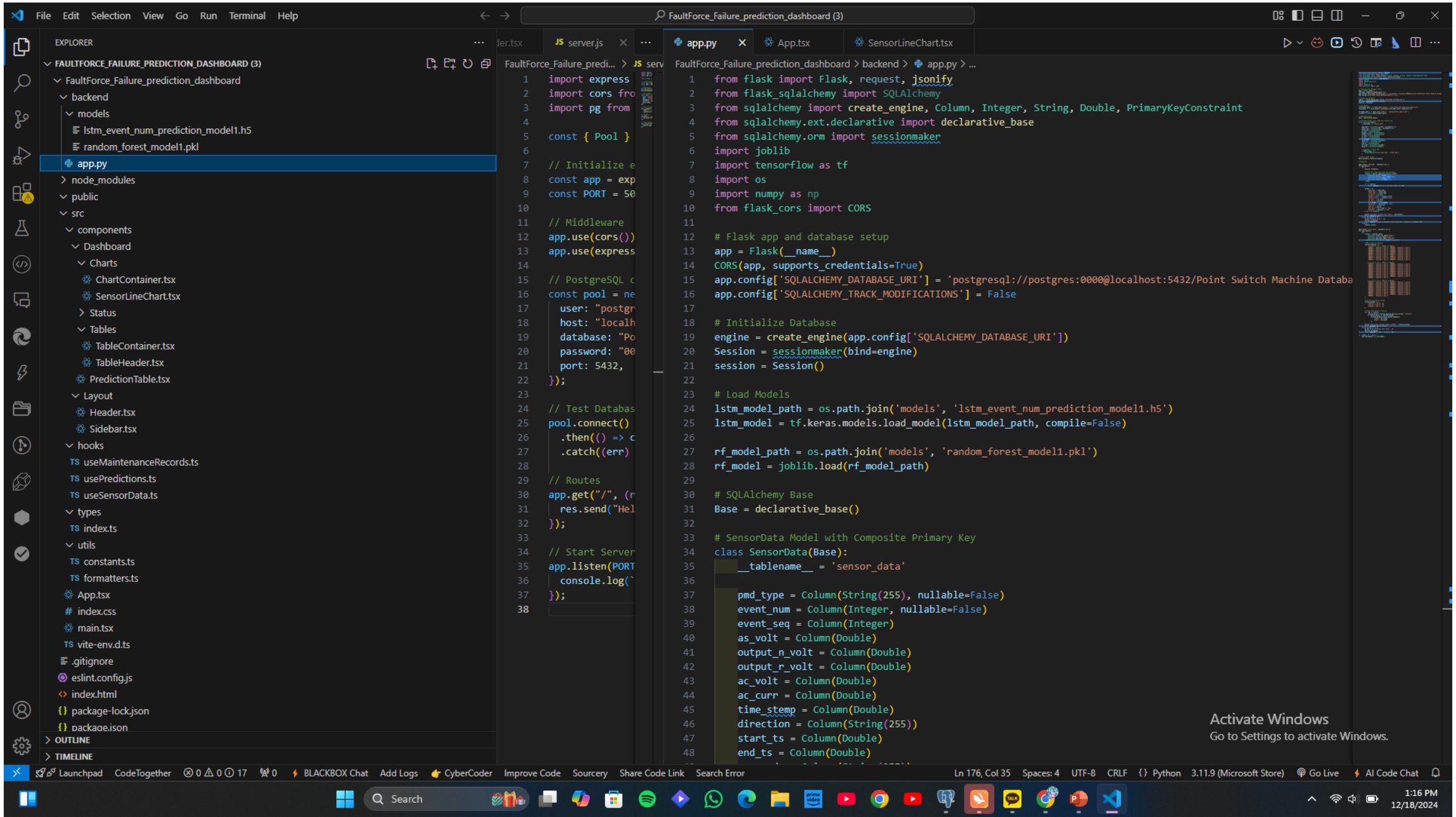


Backend

- Receive request
- Handle the request
  - Data storage
  - Data retrieval
  - Other
- Storing AI prediction data
- Processing data
- Listen to AI request and update collection data
- Providing data to Frontend



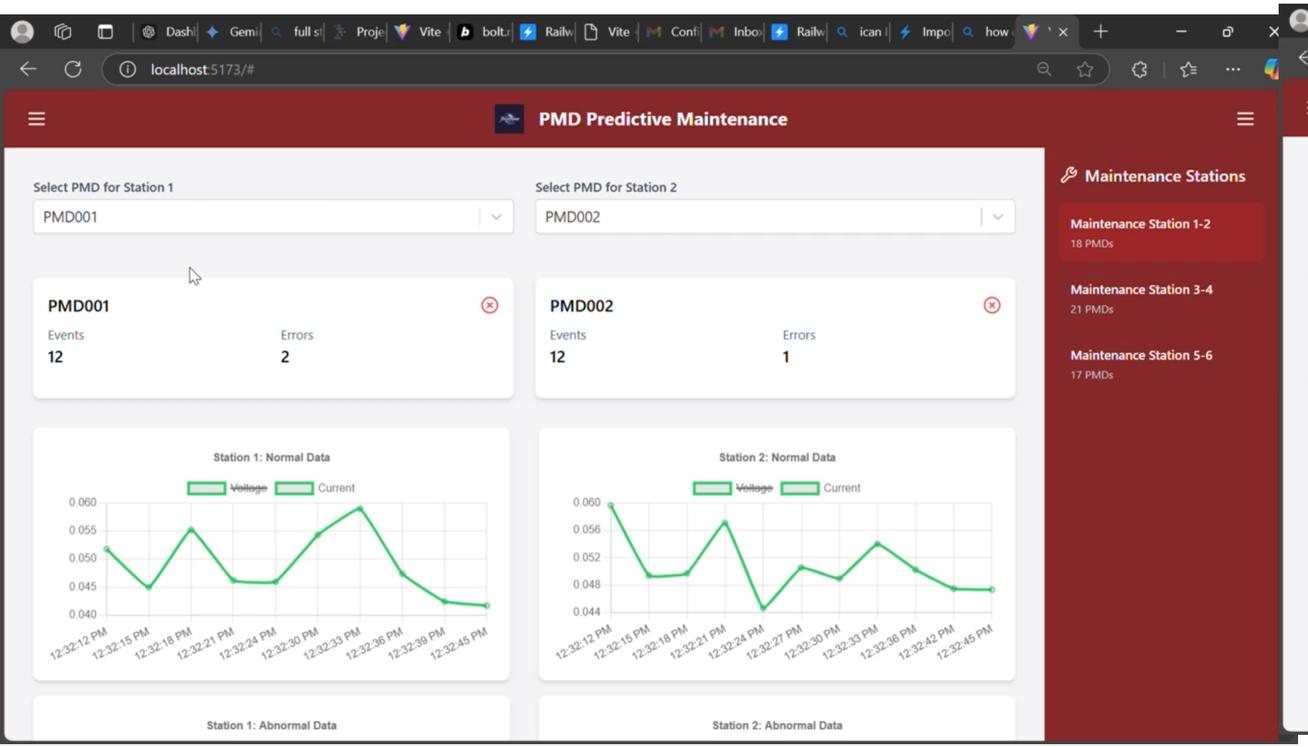
Backend Code Files

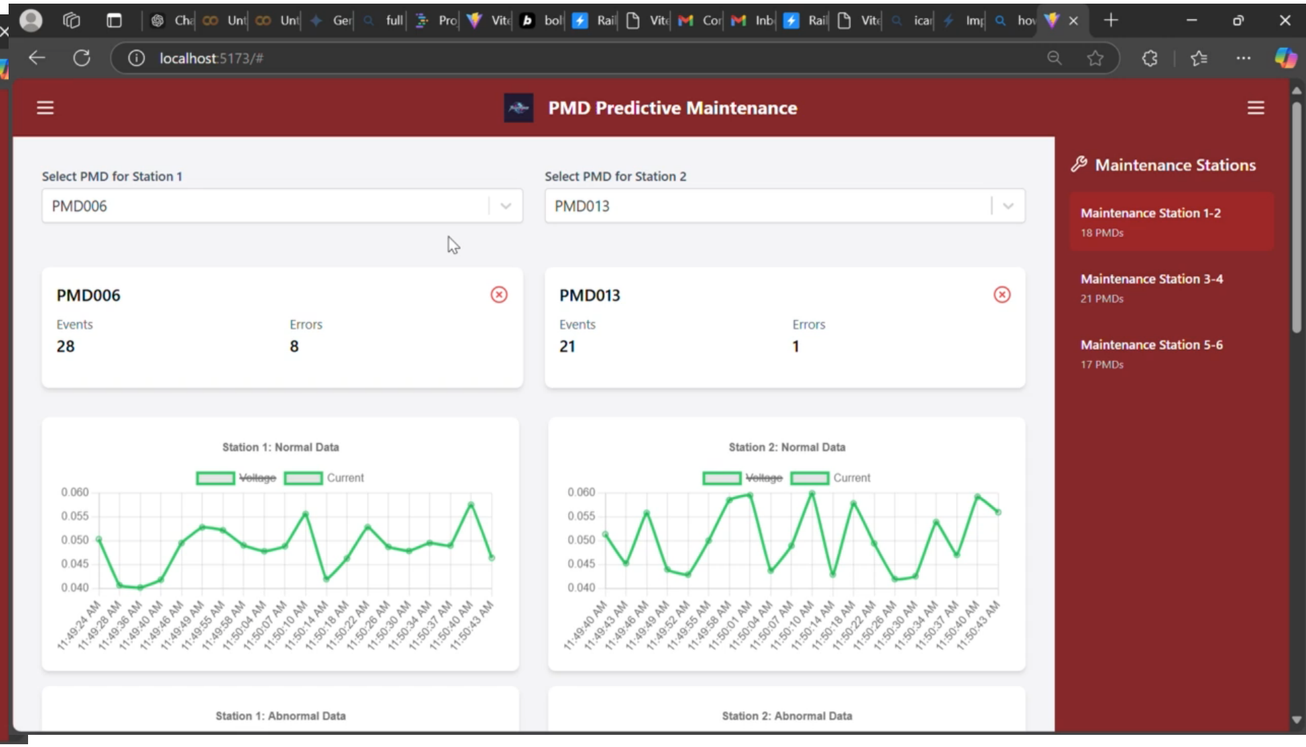


RESULTS

- Successfully trained and validated AI models.
- Backend systems operational.
- Fake data simulation UI developed for real-time visualization.
- Early fault notifications reduce maintenance downtime and accident risks.

DEMO


localhost:5173/#
PMD Predictive Maintenance
Select PMD for Station 1
PMD001
Select PMD for Station 2
PMD002
PMD001
Events
12
Errors
2
PMD002
Events
12
Errors
1
Station 1: Normal Data
Voltage
Current
Station 2: Normal Data
Voltage
Current
Station 1: Abnormal Data
Station 2: Abnormal Data
Maintenance Stations
Maintenance Station 1-2
18 PMDs
Maintenance Station 3-4
21 PMDs
Maintenance Station 5-6
17 PMDs


localhost:5173/#
PMD Predictive Maintenance
Select PMD for Station 1
PMD006
Select PMD for Station 2
PMD013
PMD006
Events
28
Errors
8
PMD013
Events
21
Errors
1
Station 1: Normal Data
Voltage
Current
Station 2: Normal Data
Voltage
Current
Station 1: Abnormal Data
Station 2: Abnormal Data
Maintenance Stations
Maintenance Station 1-2
18 PMDs
Maintenance Station 3-4
21 PMDs
Maintenance Station 5-6
17 PMDs

NEXT SOLUTIONS

1

Real Time Monitoring:

- Integration with real-time sensor data for live updates.

2

Risk Level Development:

- Provides more granular notification to maintenance personnel by separating risk levels

3

Complete Database Integration:

- Integration with the bigger PMD sensor database

4

UI Enhancement:

- Add more functions and individual power graphs for Voltage and Current in the Prediction Dashboard

THANK YOU