Phase 3: Implementation of Project

Title: Building Performance Analysis System

Objective

The goal of Phase 3 is to implement the core components of the Building Performance Analysis System based on the planning and research completed in earlier phases. This includes the development of data collection modules, analytical tools for assessing energy efficiency and structural integrity, sensor integration, and implementation of data security protocols.

1. Data Collection Module

Overview
The performance analysis begins with accurate data acquisition. This module is responsible for collecting information about building parameters such as energy usage, temperature, humidity, occupancy, and more.

Implementation

* **Sensor Deployment**: IoT-based sensors are installed across key zones of the building to capture real-time environmental and energy data.
* **Data Sources**: Input is gathered from HVAC systems, smart meters, and occupancy sensors. Historical data logs are also used for trend analysis.

Outcome
By the end of this phase, a robust system for gathering building performance metrics will be operational.

---

2. Analytical Tools Development

Overview
The collected data is analyzed to evaluate various aspects of building performance, including energy efficiency, comfort levels, and maintenance needs.

Implementation

* **Algorithm Design**: Analytical algorithms process raw data to generate performance metrics such as Energy Use Intensity (EUI), thermal comfort scores, and system efficiency ratings.
* **Visualization**: Dashboards are developed to present actionable insights to building managers.

Outcome
The system will provide a clear, visual representation of performance parameters, enabling data-driven decisions.

---

3. Sensor Integration (Optional)

Overview
Although optional in this phase, initial integration with smart building systems is initiated to facilitate automation.

Implementation

* **Communication Protocols**: Use of MQTT or BACnet to enable real-time data transmission from sensors.
* **Device Compatibility**: Ensure the framework supports data from third-party devices (e.g., Nest, Ecobee).

Outcome
The system will demonstrate the capability to integrate with building infrastructure and pull real-time performance metrics where devices are available.

---

4. Data Security Implementation

Overview
Given the privacy and operational importance of building data, security is a key focus. Basic protections are implemented in this phase.

Implementation

* **Encryption**: All performance data is encrypted during storage and transmission.
* **Access Control**: Only authorized users can access the system dashboard and raw data.

Outcome
Sensitive performance data will be securely stored and shared, meeting basic compliance and safety standards.

5. Testing and Feedback Collection

Overview
Initial testing evaluates the accuracy and reliability of data collection and analysis systems.

Implementation

* **Pilot Test Sites**: One or two buildings are selected for trial runs.
* **Feedback Channels**: Building managers provide input on usability, report clarity, and perceived system value.

Outcome
Collected feedback will help improve algorithms, sensor placement strategies, and user interfaces in the next phase.

Challenges and Solutions

1. Sensor Calibration

   * *Challenge*: Variability in sensor readings.
   * *Solution*: Regular calibration and data normalization techniques.

2. Data Overload

   * *Challenge*: Large volumes of data can slow analysis.
   * *Solution*: Data preprocessing and edge computing filters.

3. Integration Complexity

   * *Challenge*: Multiple protocols and device standards.
   * *Solution*: Develop modular integration layers using standard APIs.

Outcomes of Phase 3

1. Fully functional data collection system for performance metrics.
2. Operational analysis engine providing building insights.
3. Optional real-time sensor integration in test buildings.
4. Secure data management architecture in place.
5. Initial feedback gathered for future improvement.

Next Steps for Phase 4

1. Enhance accuracy and resolution of analytical outputs.
2. Expand to more building types and environments.
3. Optimize system for real-time decision-making and predictive analytics

main.py

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
import warnings
warnings.filterwarnings('ignore')
def generate_building_data(num_days=30, interval_minutes=15):
    total_points = (24 * 60 // interval_minutes) * num_days
    timestamps = pd.date_range(end=datetime.datetime.now(), periods
        =total_points, freq=f'{interval_minutes}min')

    temperature = np.random.normal(loc=22, scale=2, size=total_points)
    humidity = np.random.normal(loc=50, scale=6, size=total_points)
    occupancy = np.random.randint(0, 100, size=total_points)
    external_temp = np.random.normal(loc=30, scale=5, size=total_points)
    base_load = 100
    hvac_load = (temperature - 21) * 3
    occupancy_load = occupancy * 1.2
    external_influence = (external_temp - 25) * 2

    energy_usage = base_load + hvac_load + occupancy_load + external_influence
        + np.random.normal(0, 10, total_points)
    energy_usage = np.clip(energy_usage, 50, 500)

    df = pd.DataFrame({
        'timestamp': timestamps,
        'internal_temp': temperature,
        'external_temp': external_temp,
        'humidity': humidity,
        'occupancy': occupancy,
        'energy_usage': energy_usage
    })
    return df
df = generate_building_data()
df['hour'] = df['timestamp'].dt.hour
df['day_of_week'] = df['timestamp'].dt.dayofweek
features = ['internal_temp', 'external_temp', 'humidity', 'occupancy', 'hour',
    'day_of_week']
X = df[features]
y = df['energy_usage']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=42)
model = RandomForestRegressor(n_estimators=100, random_state=42)
```

```python
model.fit(X_train, y_train)
df['predicted_energy'] = model.predict(X)
threshold = 30
df['anomaly'] = np.where(abs(df['energy_usage'] - df['predicted_energy']) >
    threshold, 1, 0)
print("Model MAE:", mean_absolute_error(y_test, model.predict(X_test)))
print("Total Anomalies Detected:", df['anomaly'].sum())
plt.figure(figsize=(12, 6))
sns.lineplot(data=df[:500], x='timestamp', y='energy_usage', label='Actual')
sns.lineplot(data=df[:500], x='timestamp', y='predicted_energy', label
    ='Predicted')
plt.title('Energy Usage vs Prediction')
plt.xlabel('Time')
plt.ylabel('Energy (kWh)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
df.to_csv("building_performance_report.csv", index=False)
print("Report saved as 'building_performance_report.csv'")
```