IOT Phase-2 Project; Smart Public Restroom

Creating a predictive maintenance system for a smart public restroom involves several steps. Here's a high-level overview with some coding guidance:

1.Data Collection:

- First, gather sensor data from the restroom equipment, such as toilet usage, paper towel dispensers, soap dispensers, and water usage. This data can be collected using IoT sensors and sent to a central server.
- Use libraries like pandas and numpy for data manipulation.

Coding's;

```
import pandas as pd

import numpy as np

# Load sensor data into a DataFrame

sensor_data = pd.read_csv('sensor_data.csv')
```

2.Data Preprocessing:

- Clean and preprocess the sensor data. This may involve handling missing values, smoothing noisy data, and converting timestamps into a usable format. Python libraries like Pandas can be helpful for this.

Coding's;

```
# Handling missing values by filling with mean
```

```
sensor_data.fillna(sensor_data.mean(), inplace=True)


# Removing outliers using z-scores
from scipy import stats
z_scores = np.abs(stats.zscore(sensor_data))
sensor_data = sensor_data[(z_scores < 3).all(axis=1)]
```

3.Feature Engineering:

- Create relevant features from the sensor data that can be used for predictive maintenance. For example, calculate the frequency of toilet flushes or soap dispenser refills over time.

Coding's;

```
# Calculate daily toilet flush counts
sensor_data['Daily_Flush_Count'] =
sensor_data.groupby('Date')['Toilet_Flush'].transform('count')
```

4.Machine Learning Model Selection:

- Choose a suitable machine learning algorithm for predictive maintenance. Common choices include regression models, decision trees, or deep learning models. Python libraries like scikit-learn and TensorFlow can be used here.

Coding's;

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Define features and target variable
X = sensor_data[['Feature1', 'Feature2', ...]]
y = sensor_data['Maintenance_Needed']
# 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)
# Create and train a Random Forest classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy}")
```

5.Training the Model:

- Split your data into training and testing sets. Train the machine learning model on the training data, using historical sensor data to learn patterns of equipment failures or maintenance needs.

6.Model Evaluation:

- Evaluate the model's performance on the testing data using metrics like accuracy, precision, recall, or F1-score. Make sure it provides reliable predictions.

7.Deployment:

- Deploy the trained model to a server that can continuously process incoming sensor data.

8.Real-Time Predictions:

- Continuously feed real-time sensor data into the deployed model to make predictions about when maintenance is needed. You can use libraries like Flask or Django to create a web service for this purpose.

Coding’s;

```
# Assuming new_data is the real-time sensor data
new_data = pd.read_csv('new_sensor_data.csv')
new_features = new_data[['Feature1', 'Feature2', ...]]
# Make real-time predictions
real_time_predictions = model.predict(new_features)
```

9.Alerting System:

- Implement an alerting system that notifies maintenance staff when the model predicts that maintenance is required. You can use email, SMS, or push notifications for this.

Coding’s;

```
if any(real_time_predictions):
   send_alert_to_maintenance("Maintenance needed in the restroom!")
```

10.Feedback Loop:

- Periodically retrain your model with new data to keep it up-to-date and improve its accuracy over time.

Here's a simplified Python code snippet using scikit-learn for creating a basic predictive maintenance model:

Code’s:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
# Load and preprocess data
data = pd.read_csv('sensor_data.csv')
# Perform data preprocessing and feature engineering here
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
```

```
# Create and train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
# Make predictions
y_pred = model.predict(X_test)
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy}")
```

This code provides a basic framework for implementing predictive maintenance in a smart public restroom.we should adapt it to the specific sensor data, features, and requirements. Additionally, consider more advanced techniques and monitoring for model performance over time.