University College Of Engineering ,Ariyalur

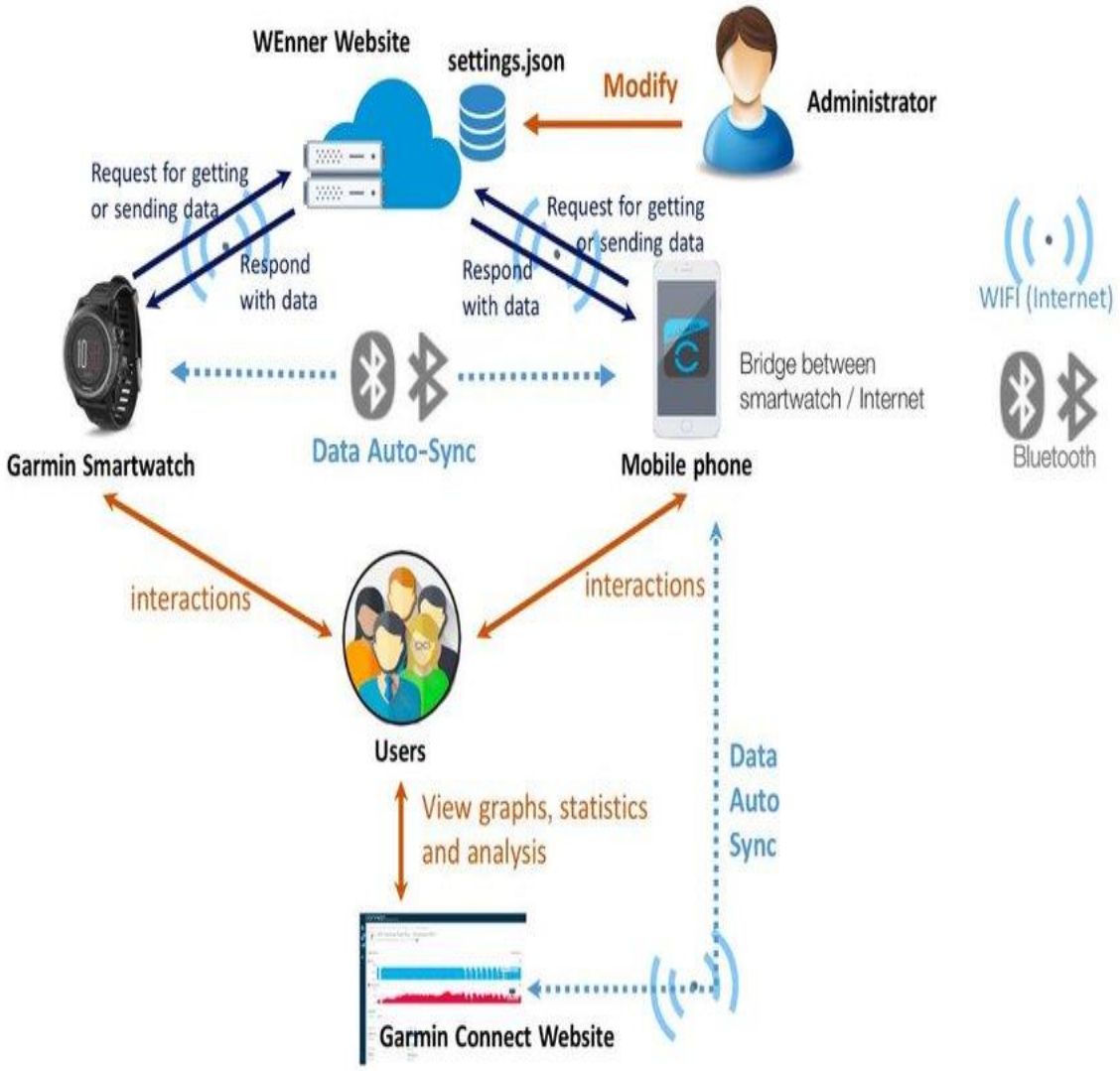
# Noise Pollution Monitoring

Phase 3

S.Preethi
814821106023

# Phase 3: Development Part 1

*Start Building Noise Pollution Monitoring By Loading and Preprocessing Dataset*



## Preprocess noise data:

- Noise can be handled using binning. In this technique, sorted data is placed into bins or buckets. Bins can be created by equal-width (distance) or equal-depth (frequency) partitioning. On these bins, smoothing can be applied.
- Data cleaning methods such as data validation, normalization, transformation, or correction can be employed to remove or fix any errors or inconsistencies and improve the accuracy.

## Loading Noise pollution:

- The WHO classifies noise above 65 dB as pollution. Noise is detrimental at 75 dB and agonizing at 120 dB. Thus, daytime noise levels should be kept below 65 dB, and nighttime ambient noise beyond 30 dB prevents restful sleep
- The annoyance produced by a sound is directly related to its sound energy, the basic indicator of which is the sound pressure level. This indicator varies over time and in environmental studies is measured in decibels (dB) and expressed by $L_A$.

# Building the Noise Pollution Monitoring Loading

## And Preprocessing the Dataset

follow these steps:

### Step 1: Import Libraries

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler
```

### Step 2: Load the Dataset

```
# Assuming your dataset is in CSV format

dataset_path = 'path/to/your/noise_pollution_dataset.csv'

df = pd.read_csv(dataset_path)


# Display the first few rows to understand the structure of the dataset

print(df.head())
```

## Step 3: Exploratory Data Analysis (EDA)

- Explore the dataset to understand its characteristics and identify potential preprocessing steps

```
# Display basic statistics
print(df.describe())
# Visualize the distribution of noise levels
import matplotlib.pyplot as plt
plt.hist(df['noise_level'], bins=20, edgecolor='black')
plt.xlabel('Noise Level')
plt.ylabel('Frequency')
plt.title('Distribution of Noise Levels')
plt.show()
# Explore relationships between features
import seaborn as sns
sns.pairplot(df)
plt.show()
```

## Step 4: Handling Missing Values

```
# Check for missing values
print(df.isnull().sum())

# Handle missing values (e.g., drop or impute)
df = df.dropna()
```

## Step 5: Feature Engineering

- Create new features or transform existing ones if necessary.

```
# Example: Extract hour and minute from a timestamp
df['hour'] = pd.to_datetime(df['timestamp']).dt.hour
df['minute'] = pd.to_datetime(df['timestamp']).dt.minute
```

## Step 6: Encoding Categorical Variables

```
# If you have categorical variables, encode them
df = pd.get_dummies(df, columns=['location', 'day_of_week'])
```

## Step 7: Scaling Numerical Features

```
# If your numerical features are on different scales, standardize them

scaler = StandardScaler()

df[['feature1', 'feature2']] = scaler.fit_transform(df[['feature1', 'feature2']])
```

## Step 8: Splitting the Data

```
# Assuming 'target' is your target variable

X = df.drop('target', axis=1)

y = df['target']

# Split the dataset 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)
```

## Step 9: Final Check

```
# Display basic information about the preprocessed dataset

print(df.info())
```

**Conclusion:**

- This is a basic outline to get you started with loading and exploring your noise pollution dataset.
- The specific steps may vary depending on the characteristics of your data.
- If you have more details about your dataset or encounter specific issues, feel free to provide additional information for more targeted assistance.

*Thank you*