Smart parking using IoT

Smart parking:

Creating a smart parking project using IoT involves various components and steps.

Sensor Installation: Deploy IoT sensors (e.g., ultrasonic or infrared) in each parking space to detect occupancy. These sensors will collect data about whether a parking spot is vacant or occupied.

Sensor Data Collection: The sensors send data to a central hub or gateway. This data usually includes information about the parking spot's status, such as "occupied" or "vacant," along with a unique identifier for each parking space.

Data Transmission: The central hub or gateway transmits the collected data to a cloud platform or a local server using a communication protocol such as MQTT or HTTP.

Data Storage: On the cloud platform or server, the data is stored in a database. You can use databases like MySQL, MongoDB, or cloud-based solutions like AWS DynamoDB or Azure Cosmos DB.

Data Analysis and Processing: Implement data processing logic to analyze the data. This can include identifying patterns, generating statistics, and managing real-time alerts for parking availability.

User Interface: Develop a user-friendly interface, often a mobile app or a web application, where users can view parking spot availability in real time.

Data Import: In your project, data import can involve periodic updates from sensors or real-time streaming. You might need to implement APIs or data pipelines to facilitate this data import process.

Data Visualization: Present the imported data in a user-friendly format, such as a map with color-coded parking spots or a list of available parking spaces.

User Notifications: Set up a notification system to inform users when a parking spot becomes available or to reserve a spot in advance.

Reporting and Analytics: Create reports and analytics to track parking usage, trends, and historical data.

Please note that the specific tools, platforms, and technologies used for your project may vary based on your preferences and the IoT hardware and software you have available.

Step 1:Import the necessary libraries

Python code to import necessary libraries,

```
i
m
```

Step 2: Load the dataset

Python code,

```
# Load the dataset
```

Step 3: Data cleaning

The goal of data cleaning is to remove any errors or inconsistencies from the dataset. This may involve:

- Checking for missing values: We can use the *isna()* function in Pandas to check for missing values in each column of the dataset. If we find any missing values, we can either drop the rows with missing values or impute the missing values with a reasonable value.
- Dropping duplicate rows: We can use the *duplicated()* function in Pandas to check for duplicate rows in the dataset. If we find any duplicate rows, we can drop them from the dataset.
- Converting the data types to the appropriate format: We should check the data types of each column in the dataset and convert them to the appropriate format if necessary. For example, we may want to convert the *datetime* column to a datetime object.

  Python code,

```
#

C
h
e
c
k
```

Step 4: Data analysis

Once the data has been cleaned, we can begin to analyze it. Here are some examples of data analysis that we can perform on a smart parking dataset:

- Get the number of parking slots available and occupied at each time interval: We can use the groupby() function in Pandas to group the data by

datetime and then use the value_counts() function to count the number of parking slots available and occupied at each time interval.

- Plot the number of parking slots available and occupied over time: We can use the plot() function in Matplotlib to plot the number of parking slots available and occupied over time. This can help us to identify peak parking times and trends in parking occupancy.
- Calculate the average parking occupancy rate: We can calculate the average parking occupancy rate by dividing the number of parking slots occupied by the total number of parking slots and then multiplying by 100.
- Identify the most popular parking areas: We can use the groupby() function in Pandas to group the data by parking_location and then use the value_counts() function to count the number of vehicles parked in each parking area. This can help us to identify the most popular parking areas.
- Analyze the relationship between parking occupancy rate and other factors: We can use correlation analysis to analyze the relationship between parking occupancy rate and other factors, such as weather, traffic conditions, and events. This can help us to understand how these factors impact parking demand.
- Use machine learning to predict future parking demand and recommend parking spots to drivers: We can use machine learning algorithms to predict future parking demand and recommend parking spots to drivers. This can help drivers to find parking more easily and reduce traffic congestion.

Python code,

```
# Get the number of parking slots available and occupied at each timeinterval
df_grouped = df.groupby('datetime')['parking_status'].value_counts()df_grouped =
```

```
# Calculate the average parking occupancy rate
```

This code will produce a plot of the number of parking slots available and occupied over time, as well as the average parking occupancy rate.

By performing data cleaning and analysis on a smart parking dataset, we can gain valuable insights into how parking resources are being used and identify areas for improvement. This information can be used to develop more efficient and sustainable parking solutions.