PHASE 4: DEVELOPMENT PART 2

BUILDING THE PROJECT BY DEVELOPING THE MOBILE APP USING PYTHON.

**Introduction:**

we are going to Create a Vehicle Parking Management in Python. This project helps maintain the details of the vehicle owner, number, type, date, and the amount of time the vehicle is parked in the area. Accordingly, the bill generates for the particular vehicle parked in the area. This information is useful for all those who want to maintain a database of the individual who has parked their vehicles in the surroundings.

### Import Function and Initializing the variables:

1. #Import Time
2. import time
3. Vehicle\_Number=['XXXX-XX-XXXX']
4. Vehicle\_Type=['Bike']
5. vehicle\_Name=['Intruder']
6. Owner\_Name=['Unknown']
7. Date=['22-22-3636']
8. Time=['22:22:22']
9. bikes=100
10. cars=250
11. bicycles=78

In this code block, we are importing the [time](https://www.programiz.com/python-programming/time) module to implement its methods and function in the project. We have initialized the variables vehicle number, vehicle type, vehicle name, owner name date, and time to some default value. As well as bikes, cars, and bicycles with some initial value.

### 2. Create a while loop block to display the options in Vehicle Parking Management Project:

def main():

global bikes,cars,bicycles

try:

while True:

print("-----------------------------------------------------")

print("\t\tParking Management System")

print("-------------------------------------------------------")

print("1.Vehicle Entry")

print("2.Remove Entry" )

print("3.View Parked Vehicle ")

print("4.View Left Parking Space ")

print("5.Amount Details ")

print("6.Bill")

print("7.Close Programme ")

print("+---------------------------------------------+")

ch=int(input("\tSelect option:"))

In this code block, we have initialized the bikes, cars, and bicycles as global variables. They are accessible through the entire main block. Here we are providing the options to choose the service options from the list, for the vehicle parking management system.

Now we will understand each service option in detail.

Code for vehicle number entry:

if ch==1:

no=True

while no==True:

Vno=input("\tEnter vehicle number (XXXX-XX-XXXX) - ").upper()

if Vno=="":

print("###### Enter Vehicle No. ######")

elif Vno in Vehicle\_Number:

print("###### Vehicle Number Already Exists")

elif len(Vno)==12:

no=not True

Vehicle\_Number.append(Vno)

else:

print("###### Enter Valid Vehicle Number ######")

Ch is for choice, Once we select the ch option as 1 which is for vehicle entry number, then we provide the while loop. while the number(no is True). We will store the vehicle number in V no. If the vno is empty i.e vno==“”.The user asks to enter the vehicle number, else If the vno entered is already present in the vehicle number then it prints the vehicle number already exists. Else if len(vno)==12, It will ask to append the info to the vehicle number variable.

### Code to enter the vehicle type:

typee=True

while typee==True:

Vtype=str(input("\tEnter vehicle type(Bicycle=A/Bike=B/Car=C):")).lower()

if Vtype=="":

print("###### Enter Vehicle Type ######")

elif Vtype=="a":

Vehicle\_Type.append("Bicycle")

bicycles-=1

typee=not True

elif Vtype=="b":

Vehicle\_Type.append("Bike")

bikes-=1

typee=not True

elif Vtype=="c":

Vehicle\_Type.append("Car")

cars-=1

typee=not True

else:

print("###### Please Enter Valid Option ######")

Here we have to initialize the typee variable to true. While the condition is True, the system asks to enter the vehicle type i.e a,b, or c which will accept the input in the lower case. Here A is for bicycle, B is for Bike and C is for Car. Any vehicle type you enter is stored in the variable Vtype. If the Vtype==””(empty). It will ask to enter the vehicle type.  According to the type of variable you enter the vehicle type will be stored in the variable and typee variable is set to not True.

### Code to enter the vehicle name:

name=True

while name==True:

vname=input("\tEnter vehicle name - ")

if vname=="":

print("########Please Enter Vehicle Name ########")

else:

vehicle\_Name.append(vname)

name=not True

Here we have set the name== True. While the name == True i.e until we enter the name.vname store the value i.e. vehicle name.  if the vname is empty system asks to enter the vehicle name, else it will store the name using the append function to the vehicle name variable The name variable is initialized to not True.

### Code to enter the owners name

o=True

while o==True:

OName=input("\tEnter owner name - ")

if OName=="":

print("###### Please Enter Owner Name ######")

else:

Owner\_Name.append(OName)

o=not True

O is initialized to True. While the condition satisfies the Owner’s name is stored in the OName variable. If the OName is empty it system asks to enter the owner name else it will store the Owner name in the owner name variable. O is now initialized to Not True.

### Code enter the date and time:

d=True

while d==True:

date=input("\tEnter Date (DD-MM-YYYY) - ")

if date=="":

print("###### Enter Date ######")

elif len(date)!=10:

print("###### Enter Valid Date ######")

else:

Date.append(date)

d=not True

t=True

while t==True:

time=input("\tEnter Time (HH:MM:SS) - ")

if t=="":

print("###### Enter Time ######")

elif len(time)!=8:

print("###### Please Enter Valid Time ######")

else:

Time.append(time)

t=not True

print("\n............................................................Record")

Similarly, we have to create a while loop to enter the date and time initializing d and t to 0. Date variable stores the date and the time-variable stores time. The date and time variable checks the condition and accordingly execute further.

### Code to remove the entry from the register

elif ch==2:

no=True

while no==True:

Vno=input("\tEnter vehicle number to Delete(XXXX-XX-XXXX) - ").upper()

if Vno=="":

print("###### Enter Vehicle No. ######")

elif len(Vno)==12:

if Vno in Vehicle\_Number:

i=Vehicle\_Number.index(Vno)

Vehicle\_Number.pop(i)

Vehicle\_Type.pop(i)

vehicle\_Name.pop(i)

Owner\_Name.pop(i)

Date.pop(i)

Time.pop(i)

no=not True

print("\n............................................................Removed Sucessfully..................................................................")

elif Vno not in Vehicle\_Number:

print("###### No Such Entry ######")

else:

print("Error")

else:

print("###### Enter Valid Vehicle Number ######")

### Code to display the vehicles present in the parking area

elif ch==3:

count=0

print("----------------------------------------------------------------------------------------------------------------------")

print("\t\t\t\tParked Vehicle")

print("----------------------------------------------------------------------------------------------------------------------")

print("Vehicle No.\tVehicle Type Vehicle Name\t Owner Name\t Date\t\tTime")

print("----------------------------------------------------------------------------------------------------------------------")

for i in range(len(Vehicle\_Number)):

count+=1

print(Vehicle\_Number[i],"\t ",Vehicle\_Type[i],"\t ",vehicle\_Name[i],"\t ",Owner\_Name[i]," " ,Date[i]," ",Time[i])

print("----------------------------------------------------------------------------------------------------------------------")

print("------------------------------------------ Total Records - ",count,"-------------------------------------------------------")

print("--------------------------------------------------------------------")

Here ch==3  is for displaying the parked vehicles in the parking area. For this, we have to use the [for loop](https://copyassignment.com/python/python-for-loop/) function. It counts the length of the vehicle number. This will display the whole information of the vehicles.

### Code for spaces left in the parking area

elif ch==4:

print("----------------------------------------------------------------------------------------------------------------------")

print("\t\t\t\tSpaces Left For Parking")

print("----------------------------------------------------------------------------------------------------------------------")

print("\tSpaces Available for Bicycle - ",bicycles)

print("\tSpaces Available for Bike - ",bikes)

print("\tSpaces Available for Car - ",cars)

print("---------------------------------------------------------------------")

This block of code displays the spaces left for parking in the parking area.

### Code for displaying the parking rate

elif ch==5:

print("----------------------------------------------------------------------------------------------------------------------")

print("\t\t\t\tParking Rate")

print("----------------------------------------------------------------------------------------------------------------------")

print("\*1.Bicycle Rs20 / Hour")

print("\*2.Bike Rs40/ Hour")

print("\*3.Car Rs60/ Hour")

print("---------------------------------------------------------------------")

It displays the parking rate of different types of vehicles.

### Code to generate bills for different types of vehicles parked

elif ch==6:

print(".............................................................. Generating Bill ..........................................................................")

no=True

while no==True:

Vno=input("\tEnter vehicle number to Delete(XXXX-XX-XXXX) - ").upper()

if Vno=="":

print("###### Enter Vehicle No. ######")

elif len(Vno)==12:

if Vno in Vehicle\_Number:

i=Vehicle\_Number.index(Vno)

no=not True

elif Vno not in Vehicle\_Number:

print("###### No Such Entry ######")

else:

print("Error")

else:

print("###### Enter Valid Vehicle Number ######")

print("\tVehicle Check in time - ",Time[i])

print("\tVehicle Check in Date - ",Date[i])

print("\tVehicle Type - ",Vehicle\_Type[i])

inp=True

amt=0

while inp==True:

hr=input("\tEnter No. of Hours Vehicle Parked - ").lower()

if hr=="":

print("###### Please Enter Hours ######")

elif int(hr)==0 and Vehicle\_Type[i]=="Bicycle":

amt=20

inp=not True

elif int(hr)==0 and Vehicle\_Type[i]=="Bike":

amt=40

inp=not True

elif int(hr)==0 and Vehicle\_Type[i]=="Car":

amt=60

inp=not True

elif int(hr)>=1:

if Vehicle\_Type[i]=="Bicycle":

amt=int(hr)\*int(20)

inp=not True

elif Vehicle\_Type[i]=="Bike":

amt=int(hr)\*int(40)

inp=not True

elif Vehicle\_Type[i]=="Car":

amt=int(hr)\*int(60)

inp=not True

print("\t Parking Charge - ",amt)

ac=18/100\*int(amt)

print("\tAdd. charge 18 % - ",ac)

print("\tTotal Charge - ",int(amt)+int(ac))

print("..............................................................Thank you for using our service...........................................................................")

a=input("\tPress Any Key to Proceed - ")

elif ch==7:

print("..............................................................Thank you for using our service...........................................................................")

print(" \*\*\*\*\*\*\*\*\*\*(: Bye Bye :)\*\*\*\*\*\*\*\*\*\*")

break

quit

**Complete Code to Create a Vehicle Parking Management Project in Python:**

**#Import Time**

**import time**

**Vehicle\_Number=['XXXX-XX-XXXX']**

**Vehicle\_Type=['Bike']**

**vehicle\_Name=['Intruder']**

**Owner\_Name=['Unknown']**

**Date=['22-22-3636']**

**Time=['22:22:22']**

**bikes=100**

**cars=250**

**bicycles=78**

**def main():**

**global bikes,cars,bicycles**

**try:**

**while True:**

**print("----------------------------------------------------------------------------------------")**

**print("\t\tParking Management System")**

**print("----------------------------------------------------------------------------------------")**

**print("1.Vehicle Entry")**

**print("2.Remove Entry" )**

**print("3.View Parked Vehicle ")**

**print("4.View Left Parking Space ")**

**print("5.Amount Details ")**

**print("6.Bill")**

**print("7.Close Programme ")**

**print("+---------------------------------------------+")**

**ch=int(input("\tSelect option:"))**

**if ch==1:**

**no=True**

**while no==True:**

**Vno=input("\tEnter vehicle number (XXXX-XX-XXXX) - ").upper()**

**if Vno=="":**

**print("###### Enter Vehicle No. ######")**

**elif Vno in Vehicle\_Number:**

**print("###### Vehicle Number Already Exists")**

**elif len(Vno)==12:**

**no=not True**

**Vehicle\_Number.append(Vno)**

**else:**

**print("###### Enter Valid Vehicle Number ######")**

**typee=True**

**while typee==True:**

**Vtype=str(input("\tEnter vehicle type(Bicycle=A/Bike=B/Car=C):")).lower()**

**if Vtype=="":**

**print("###### Enter Vehicle Type ######")**

**elif Vtype=="a":**

**Vehicle\_Type.append("Bicycle")**

**bicycles-=1**

**typee=not True**

**elif Vtype=="b":**

**Vehicle\_Type.append("Bike")**

**bikes-=1**

**typee=not True**

**elif Vtype=="c":**

**Vehicle\_Type.append("Car")**

**cars-=1**

**typee=not True**

**else:**

**print("###### Please Enter Valid Option ######")**

**name=True**

**while name==True:**

**vname=input("\tEnter vehicle name - ")**

**if vname=="":**

**print("########Please Enter Vehicle Name ########")**

**else:**

**vehicle\_Name.append(vname)**

**name=not True**

**o=True**

**while o==True:**

**OName=input("\tEnter owner name - ")**

**if OName=="":**

**print("###### Please Enter Owner Name ######")**

**else:**

**Owner\_Name.append(OName)**

**o=not True**

**d=True**

**while d==True:**

**date=input("\tEnter Date (DD-MM-YYYY) - ")**

**if date=="":**

**print("###### Enter Date ######")**

**elif len(date)!=10:**

**print("###### Enter Valid Date ######")**

**else:**

**Date.append(date)**

**d=not True**

**t=True**

**while t==True:**

**time=input("\tEnter Time (HH:MM:SS) - ")**

**if t=="":**

**print("###### Enter Time ######")**

**elif len(time)!=8:**

**print("###### Please Enter Valid Date ######")**

**else:**

**Time.append(time)**

**t=not True**

**print("\n............................................................Record detail saved..................................................................")**

**elif ch==2:**

**no=True**

**while no==True:**

**Vno=input("\tEnter vehicle number to Delete(XXXX-XX-XXXX) - ").upper()**

**if Vno=="":**

**print("###### Enter Vehicle No. ######")**

**elif len(Vno)==12:**

**if Vno in Vehicle\_Number:**

**i=Vehicle\_Number.index(Vno)**

**Vehicle\_Number.pop(i)**

**Vehicle\_Type.pop(i)**

**vehicle\_Name.pop(i)**

**Owner\_Name.pop(i)**

**Date.pop(i)**

**Time.pop(i)**

**no=not True**

**print("\n............................................................Removed Sucessfully..................................................................")**

**elif Vno not in Vehicle\_Number:**

**print("###### No Such Entry ######")**

**else:**

**print("Error")**

**else:**

**print("###### Enter Valid Vehicle Number ######")**

**elif ch==3:**

**count=0**

**print("----------------------------------------------------------------------------------------------------------------------")**

**print("\t\t\t\tParked Vehicle")**

**print("----------------------------------------------------------------------------------------------------------------------")**

**print("Vehicle No.\tVehicle Type Vehicle Name\t Owner Name\t Date\t\tTime")**

**print("----------------------------------------------------------------------------------------------------------------------")**

**for i in range(len(Vehicle\_Number)):**

**count+=1**

**print(Vehicle\_Number[i],"\t ",Vehicle\_Type[i],"\t ",vehicle\_Name[i],"\t ",Owner\_Name[i]," " ,Date[i]," ",Time[i])**

**print("----------------------------------------------------------------------------------------------------------------------")**

**print("------------------------------------------ Total Records - ",count,"-------------------------------------------------------")**

**print("----------------------------------------------------------------------------------------------------------------------")**

**elif ch==4:**

**print("----------------------------------------------------------------------------------------------------------------------")**

**print("\t\t\t\tSpaces Left For Parking")**

**print("----------------------------------------------------------------------------------------------------------------------")**

**print("\tSpaces Available for Bicycle - ",bicycles)**

**print("\tSpaces Available for Bike - ",bikes)**

**print("\tSpaces Available for Car - ",cars)**

**print("----------------------------------------------------------------------------------------------------------------------")**

**elif ch==5:**

**print("----------------------------------------------------------------------------------------------------------------------")**

**print("\t\t\t\tParking Rate")**

**print("----------------------------------------------------------------------------------------------------------------------")**

**print("\*1.Bicycle Rs20 / Hour")**

**print("\*2.Bike Rs40/ Hour")**

**print("\*3.Car Rs60/ Hour")**

**print("----------------------------------------------------------------------------------------------------------------------")**

**elif ch==6:**

**print(".............................................................. Generating Bill ..........................................................................")**

**no=True**

**while no==True:**

**Vno=input("\tEnter vehicle number to Delete(XXXX-XX-XXXX) - ").upper()**

**if Vno=="":**

**print("###### Enter Vehicle No. ######")**

**elif len(Vno)==12:**

**if Vno in Vehicle\_Number:**

**i=Vehicle\_Number.index(Vno)**

**no=not True**

**elif Vno not in Vehicle\_Number:**

**print("###### No Such Entry ######")**

**else:**

**print("Error")**

**else:**

**print("###### Enter Valid Vehicle Number ######")**

**print("\tVehicle Check in time - ",Time[i])**

**print("\tVehicle Check in Date - ",Date[i])**

**print("\tVehicle Type - ",Vehicle\_Type[i])**

**inp=True**

**amt=0**

**while inp==True:**

**hr=input("\tEnter No. of Hours Vehicle Parked - ").lower()**

**if hr=="":**

**print("###### Please Enter Hours ######")**

**elif int(hr)==0 and Vehicle\_Type[i]=="Bicycle":**

**amt=20**

**inp=not True**

**elif int(hr)==0 and Vehicle\_Type[i]=="Bike":**

**amt=40**

**inp=not True**

**elif int(hr)==0 and Vehicle\_Type[i]=="Car":**

**amt=60**

**inp=not True**

**elif int(hr)>=1:**

**if Vehicle\_Type[i]=="Bicycle":**

**amt=int(hr)\*int(20)**

**inp=not True**

**elif Vehicle\_Type[i]=="Bike":**

**amt=int(hr)\*int(40)**

**inp=not True**

**elif Vehicle\_Type[i]=="Car":**

**amt=int(hr)\*int(60)**

**inp=not True**

**print("\t Parking Charge - ",amt)**

**ac=18/100\*int(amt)**

**print("\tAdd. charge 18 % - ",ac)**

**print("\tTotal Charge - ",int(amt)+int(ac))**

**print ("..............................................................Thank you for using our service...........................................................................")**

**a=input("\tPress Any Key to Proceed - ")**

**elif ch==7:**

**print ("..............................................................Thank you for using our service...........................................................................")**

**print(" \*\*\*\*\*\*\*\*\*\*(: Bye Bye :)\*\*\*\*\*\*\*\*\*\*")**

**break**

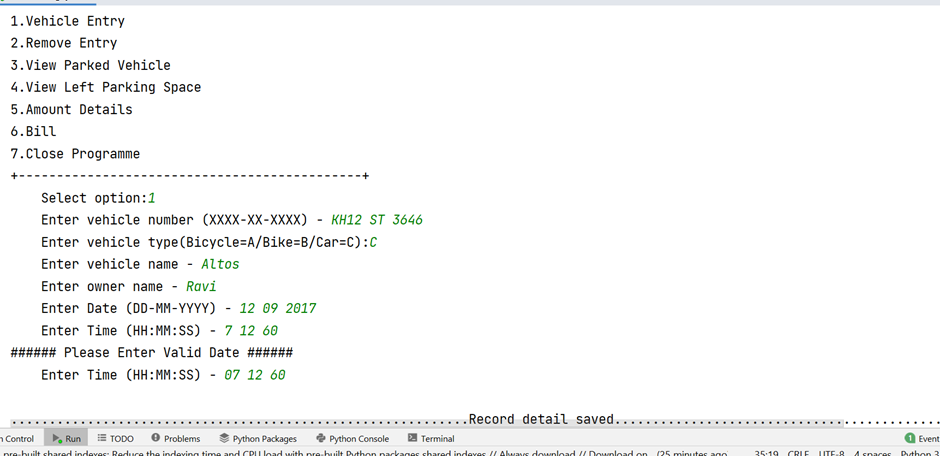
**quit**

**except:**

**main()**

**main()**

**Output:**



**USE A MOBILE APP DEVELOPMENT FRAMEWORK TO CREATE AN APP THAT DISPLAYS REALTIME PARKING AVAILABILITY.**

Creating a mobile app to display real-time parking availability data can be accomplished using various mobile app development frameworks and technologies. Here's a simplified example of how you might build such an app using React Native, a popular cross-platform framework:

**Prerequisites:**

1. Install Node.js and npm (Node Package Manager) on your computer.
2. Set up the development environment for React Native by installing Expo CLI (or React Native CLI).

**Steps to Create the App:**

1. **Initialize a New React Native Project:**

Open your command line interface and run the following commands:

**npm install -g expo-cli**

**expo init ParkingApp**

**cd ParkingApp**

**2.Install Required Packages:**

Install packages for mapping and navigation, such as **react-native-maps** for maps and **react-navigation** for navigation.

**npm install react-native-maps react-navigation**

**3.Create Components:**

Create components for your app, such as the main dashboard, parking location details, and any other necessary screens.

**4.Retrieve Real-Time Data:**

Implement a mechanism to retrieve real-time parking availability data. This might involve making API requests to your Raspberry Pi or a backend server where the data is stored. You can use libraries like **axios** for making HTTP requests.

**5.Display Data:**

Display parking availability information using the **react-native-maps** library to show parking locations on a map. You can use markers to represent the parking spots and update their status based on real-time data.

**6.Navigation:**

Implement navigation within your app using **react-navigation**. Create screens for viewing parking details, making reservations, and providing directions.

**7.Push Notifications:**

Integrate a push notification service like Firebase Cloud Messaging (FCM) to send real-time updates to users when parking availability changes significantly.

**8.User Authentication:**

If you want to provide features like favourites or user-specific data, implement user authentication. You can use libraries like Firebase Authentication for this purpose.

**9.Testing:**

Test your app on Android and iOS emulators/simulators and physical devices. Ensure that real-time data is displayed accurately and that all features work as expected.

**10.Deployment:**

When you're ready to deploy the app, you can publish it to app stores (Google Play Store and Apple App Store) or distribute it through other channels, depending on your needs.

Keep in mind that this is a simplified overview, and the actual implementation of your app may involve more specific features and technologies depending on your requirements. Additionally, you will need to handle secure communication with your Raspberry Pi, data synchronization, and user privacy considerations. Developing and deploying a real-world app also involves considerations related to user experience, design, and scalability.

Top of Form

DESIGN APP FUNCTIONS TO RECEIVE AND DISPLAY PARKING AVAILABILITY DATA RECEVIED FROM THE RASPBERRY PI

Designing an app to receive and display parking availability data from a Raspberry Pi involves several components and considerations. Below, I'll outline the functions and features you might include in such an app:

**App Features:**

1. **User Authentication:**
   * User registration and login to access parking availability data.
   * Admin authentication for managing data and settings.
2. **Dashboard:**
   * A user-friendly dashboard for users to view parking availability information.
   * Quick access to favourite or frequently used parking locations.
3. **Parking Locations:**
   * Allow users to search and select parking locations from a list or map view.
   * Display details about each parking location, including address, capacity, and real-time availability.
4. **Real-Time Data:**
   * Retrieve and display real-time parking availability data from the Raspberry Pi.
   * Update availability data at regular intervals to keep the information current.
5. **Notifications:**
   * Send push notifications or alerts to users when parking availability changes significantly, or when they approach their selected parking location.
6. **Reservation and Booking:**
   * Enable users to reserve or book parking spaces in advance, if supported by the system.
7. **Navigation:**
   * Provide directions to the selected parking location via integration with mapping services like Google Maps or Waze.
8. **Feedback and Reviews:**
   * Allow users to leave feedback, reviews, and ratings for parking locations.
9. **Favourites and History:**
   * Let users mark parking locations as favourites for easy access and keep a history of their recent parking selections.
10. **Admin Panel:**
    * An admin panel for managing parking data, adding new locations, and updating availability information.

**Technical Considerations:**

1. **Communication with Raspberry Pi:**
   * Establish a secure connection (e.g., WebSocket, REST API) between the app and the Raspberry Pi to receive real-time data.
2. **Data Storage:**
   * Store parking location details, availability data, user profiles, and reservation information in a database.
3. **Data Synchronization:**
   * Implement data synchronization mechanisms to ensure that the app receives the most up-to-date information from the Raspberry Pi.
4. **Security:**
   * Implement security measures to protect user data and the connection with the Raspberry Pi. Use encryption, secure authentication, and authorization.
5. **User Interface (UI):**
   * Design an intuitive and user-friendly interface for the app. Consider mobile and web versions for wider accessibility.
6. **Push Notifications:**
   * Integrate a push notification service to send real-time updates to users.
7. **Maps Integration:**
   * Integrate with mapping services like Google Maps for location-based features.
8. **Feedback and Review System:**
   * Implement a system for users to submit feedback and reviews, which may require a moderation process.
9. **Scalability:**
   * Design the system to scale as more parking locations and users are added.
10. **Testing and QA:**
    * Thoroughly test the app to ensure it works reliably and provides accurate parking availability information.