REPORT FOR VEHICLE PARKING MANAGEMENT SYSTEM

AS a project work for Course

PYTHON PROGRAMMING (INT213)

Name	: Utkarsh Gupta
Registration No.	: 12017462
Name	: Abhishek Yadav
Registration No.	: 12018271
Program	: CSE B.TECH
Semester	: Third
Name of the University	: Lovely Professional University
Date of submission	: 27 NOVEMBER 2021

Lovely Professional University

Jalandhar, Punjab ,India.

VEHICLE PARKING MANAGEMENT SYSTEM

20th NOVEMBER 2021

ABSTRACT:

Due to the increasing population in urban cities, there is an exponential rise in the number of vehicles which is leading to major problems leading to poor traffic management and congestion. Another major problem faced by the vehicle owners is the availability of parking space. The idea of Smart Cities is slowly gaining pace with the ever increasing technologies. Therefore, in the proposed parking system we are integrating the Wireless Parking Booking Technology with the Python Application so that the user can book or pre-book a slot. The vehicle owner will be able to reserve a slot for his/her vehicle from anywhere and will be provided with a unique slot which will be scanned on the entry of the parking area. Another feature our system provides is automatically booking your pending slot after the place is void

Acknowledgement:

I would like to thank my teacher – Miss Upinder Kaur for his advice and inputs on this project. Many thanks to my friends and seniors as well, who spent countless hours to listen and provide feedbacks.

TABLES OF CONTENTS

INTRODUCTION:

1.1 CONTEXT-

This project has been done as part of my course for the CSE at Lovely Professional University. Supervised by Upinder Kaur, We were given two months to fulfill the requirements in order to succeed the module

1.2 MOTIVATIONS AND IDEA

In Big Metropolitan cities of the world, where the infrastructure is very good due to which the open space left is very little, every building in these cities has working professionals who also have their personal vehicles and the parking all of them creates a hectic problem as they have to wait and wait and ultimately the parking is done in unordered way. In order to save these problems happening further it would be better if we could design a program so the that they can book their parking slot by simply sitting in their home, just by Using any program or application that’s how we got the idea about the project to remove inconvenience of the public.

TEAM MEMBERS:

UTKARSH GUPTA:

CONTRIBUTIONS:

1.Coding (Joined)

2.SQL

3.GUI (Joined)

4. REPORTS

5.BACKEND

ABHISHEK YADAV:

CONTRIBUTIONS:

1.Coding (Joined)

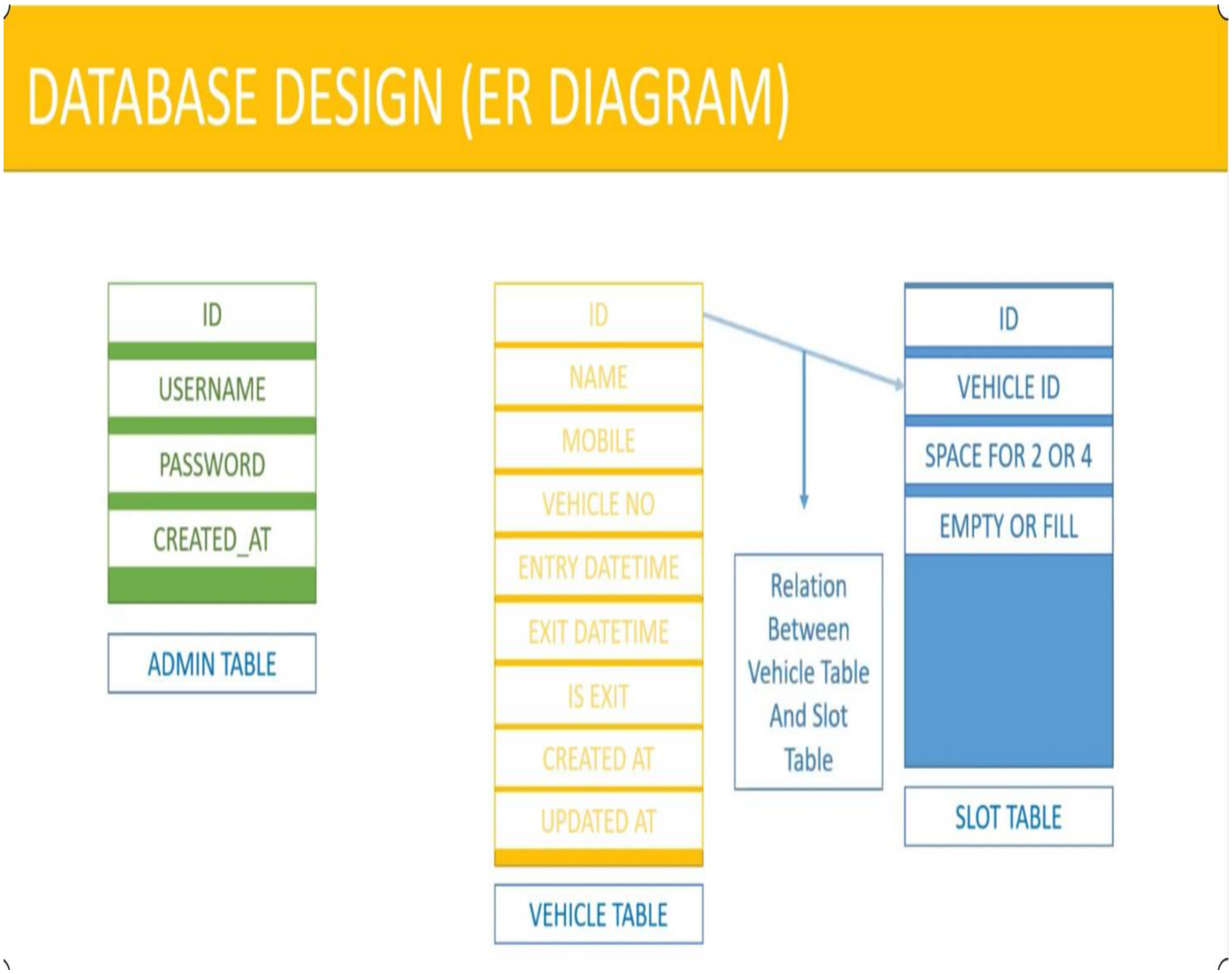
2.FRONTEND

3.GUI (Joined)

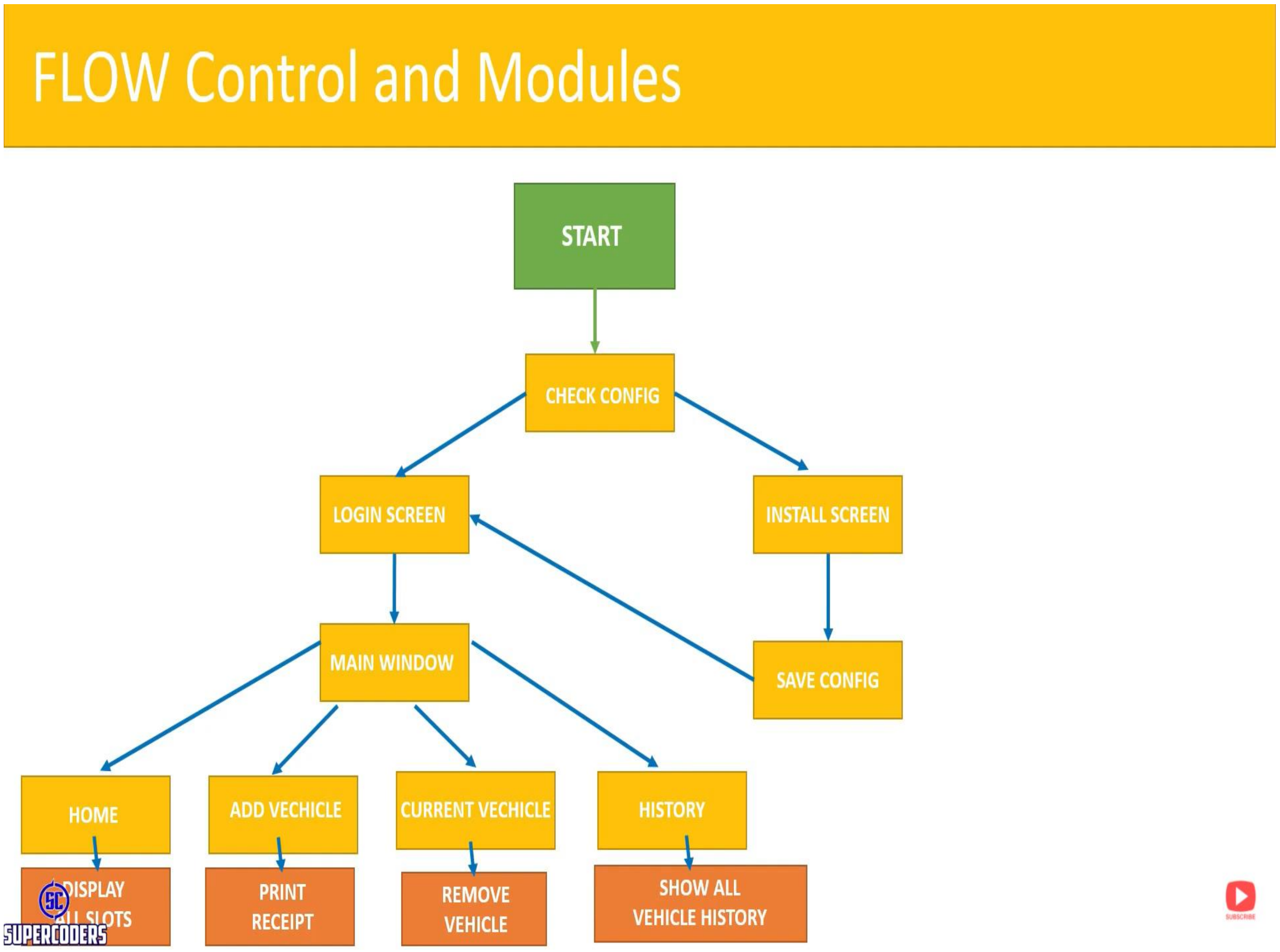
4. PPTS

5.WORKING AND TESTING

ER DIAGRAM:-


DATABASE DESIGN (ER DIAGRAM)
ID
USERNAME
PASSWORD
CREATED_AT
ADMIN TABLE
ID
NAME
MOBILE
VEHICLE NO
ENTRY DATETIME
EXIT DATETIME
IS EXIT
CREATED AT
UPDATED AT
VEHICLE TABLE
Relation Between Vehicle Table And Slot Table
ID
VEHICLE ID
SPACE FOR 2 OR 4
EMPTY OR FILL
SLOT TABLE

FLOW CONTROL AND MODULES:-



LIBRARIES:

1.PyQt5:

There are so many options provided by Python to develop GUI application and PyQt5 is one of them. PyQt5 is cross-platform GUI toolkit, a set of python bindings for Qt v5.One can develop an interactive desktop application with so much ease because of the tools and simplicity provided by this library. A GUI application consists of Front-end and Back-end. PyQt5 has provided a tool called ‘Qt Designer’ to design the front-end by drag and drop method so that development can becomes faster and one can give more time on back-end stuff.

2.TKinter:

tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

3.Mysql connector:

MySQL is a Relational Database Management System (RDBMS) whereas the structured Query Language (SQL) is the language used for handling the RDBMS using commands i.e Creating, Inserting, Updating and Deleting the data from the databases. SQL commands are case insensitive i.e. create and create signify the same command. For any application, it is very important to store the database on a server for easy data access.

4. Python Date and Time:

In Python, date and time are not a data type of their own, but a module named datetime can be imported to work with the date as well as time. Python datetime library comes built into Python, so there is no need to install it externally. Python Datetime module supplies classes to work with date and time. These classes provide number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are manipulating objects and not string or timestamps.

PROPOSED MODULES:

1.HOME WINDOW:

- CREATES GRAPHIC MODULE
- ADD VEHICLE, MANAGE VEHICLE
- SHOWS HISTORY AND HOME PAGE

2.INSTALL WINDOW:

- INSTALLS THE SYSTEM
- CREATES DATABASE USERNAME AND PASSWORD
- CREATES ADMIN USERNAMR AND PASSWORD

3.LOGIN WINDOW:

- CREATES A GRAPHICAL LOGIN WINDOW
- THIS MODULE CHECKS THE LOGIN DEATAILS

4.MAIN PROGRAM:

- IMPORTS ALL PROGRAMS AND COMPILE THEM
- INDICATES PARKING SLOT

5.DATA BASE OPERATION:

- CREATES DATABASES
- USES MY SQL SERVER
- INDICATES PARKING SPACE

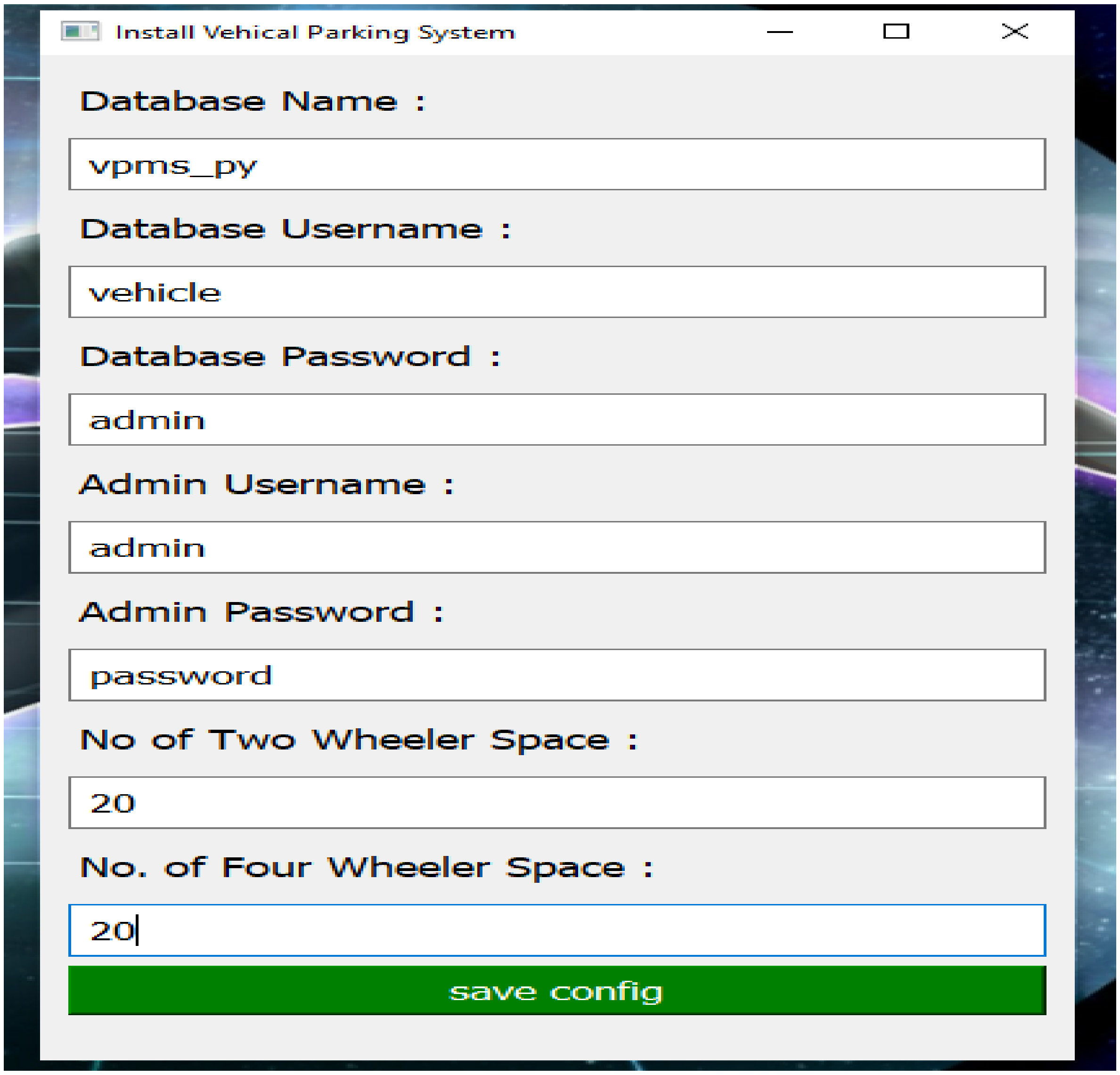
SCREENSHOTS AND WORKING:-

1.Displaying Main Page:

A home page is displayed which tells about the project topic and automatically navigates to the other window.

2.Install Window:

This Graphically Designed window starts with a database configuration page which helps us set the name of our database page which have inputs like username, admin username, passwords, No. of two wheelers and four wheelers. Designed using Pyqt module for GUI and My sql for the database part



Install Vehical Parking System

Database Name :

vpms_py

Database Username :

vehicle

Database Password :

admin

Admin Username :

admin

Admin Password :

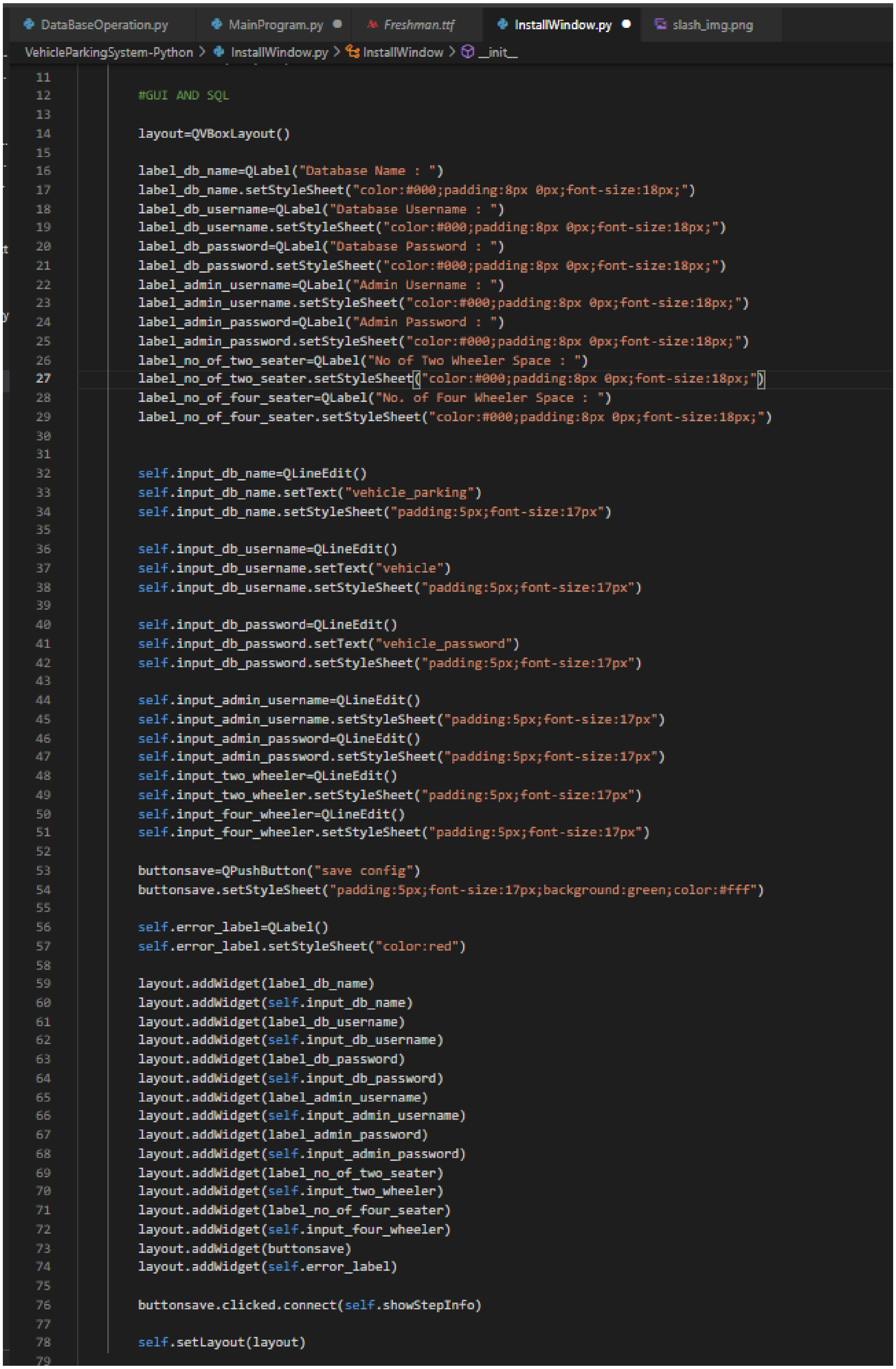
password

No of Two Wheeler Space :

20

No. of Four Wheeler Space :

20

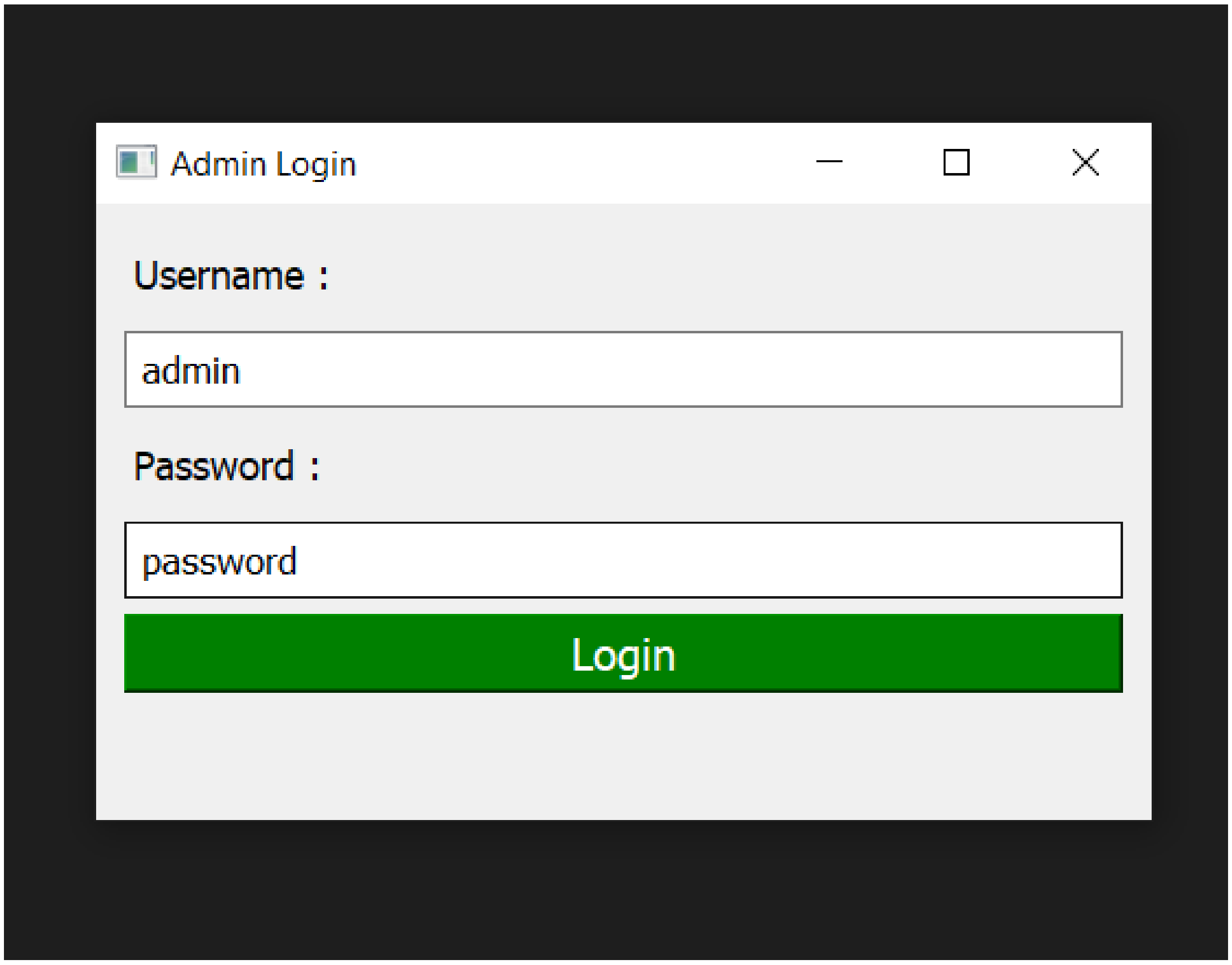

DataBaseOperation.py   MainProgram.py   Freshman.ttf   InstallWindow.py   slash_img.png

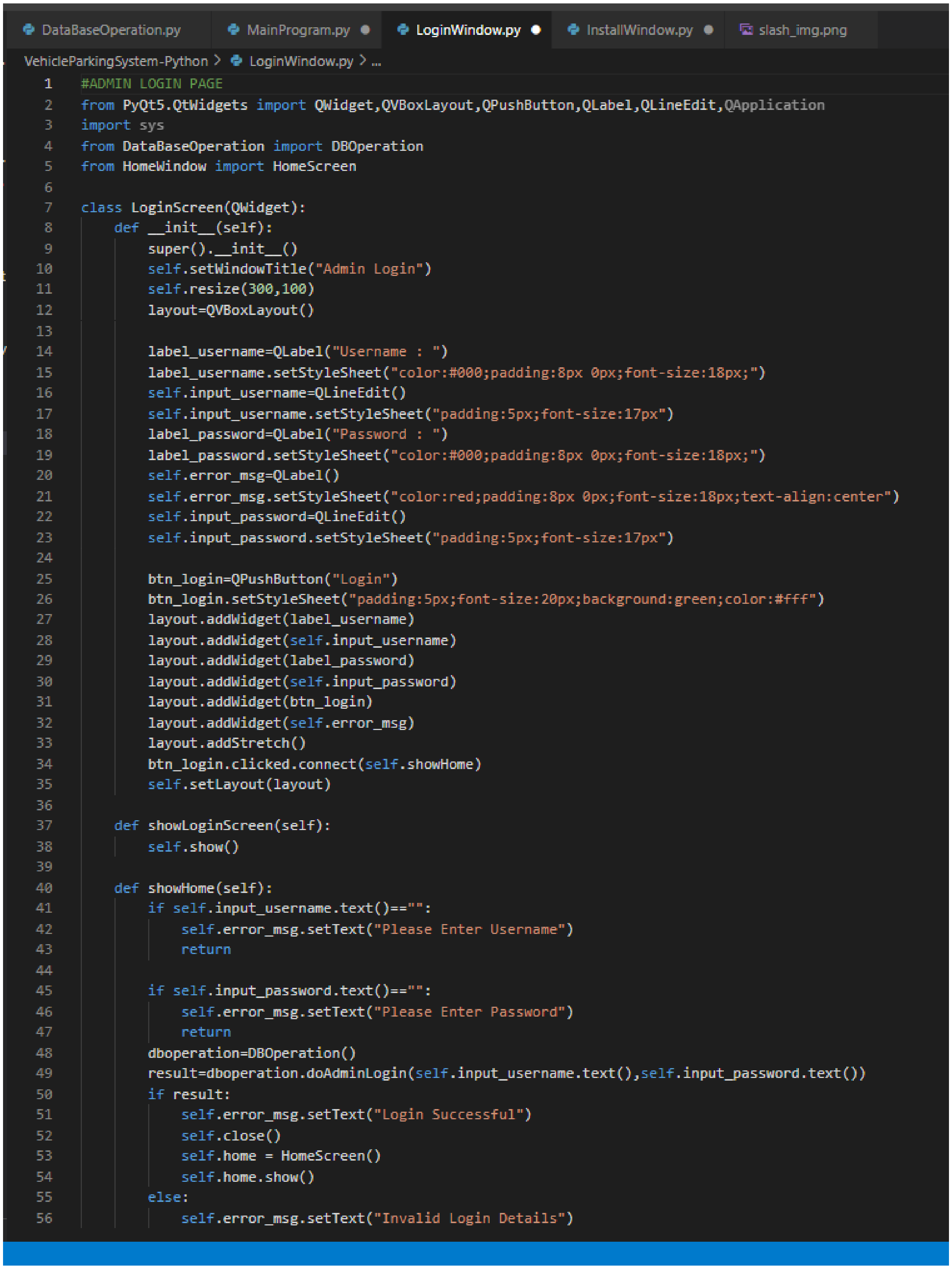
VehicleParkingSystem-Python > InstallWindow.py > InstallWindow > __init__

```python
        #GUI AND SQL

        layout=QVBoxLayout()

        label_db_name=QLabel("Database Name : ")
        label_db_name.setStyleSheet("color:#000;padding:8px 0px;font-size:18px;")
        label_db_username=QLabel("Database Username : ")
        label_db_username.setStyleSheet("color:#000;padding:8px 0px;font-size:18px;")
        label_db_password=QLabel("Database Password : ")
        label_db_password.setStyleSheet("color:#000;padding:8px 0px;font-size:18px;")
        label_admin_username=QLabel("Admin Username : ")
        label_admin_username.setStyleSheet("color:#000;padding:8px 0px;font-size:18px;")
        label_admin_password=QLabel("Admin Password : ")
        label_admin_password.setStyleSheet("color:#000;padding:8px 0px;font-size:18px;")
        label_no_of_two_seater=QLabel("No of Two Wheeler Space : ")
        label_no_of_two_seater.setStyleSheet("color:#000;padding:8px 0px;font-size:18px;")
        label_no_of_four_seater=QLabel("No. of Four Wheeler Space : ")
        label_no_of_four_seater.setStyleSheet("color:#000;padding:8px 0px;font-size:18px;")


        self.input_db_name=QLineEdit()
        self.input_db_name.setText("vehicle_parking")
        self.input_db_name.setStyleSheet("padding:5px;font-size:17px")

        self.input_db_username=QLineEdit()
        self.input_db_username.setText("vehicle")
        self.input_db_username.setStyleSheet("padding:5px;font-size:17px")

        self.input_db_password=QLineEdit()
        self.input_db_password.setText("vehicle_password")
        self.input_db_password.setStyleSheet("padding:5px;font-size:17px")

        self.input_admin_username=QLineEdit()
        self.input_admin_username.setStyleSheet("padding:5px;font-size:17px")
        self.input_admin_password=QLineEdit()
        self.input_admin_password.setStyleSheet("padding:5px;font-size:17px")
        self.input_two_wheeler=QLineEdit()
        self.input_two_wheeler.setStyleSheet("padding:5px;font-size:17px")
        self.input_four_wheeler=QLineEdit()
        self.input_four_wheeler.setStyleSheet("padding:5px;font-size:17px")

        buttonsave=QPushButton("save config")
        buttonsave.setStyleSheet("padding:5px;font-size:17px;background:green;color:#fff")

        self.error_label=QLabel()
        self.error_label.setStyleSheet("color:red")

        layout.addWidget(label_db_name)
        layout.addWidget(self.input_db_name)
        layout.addWidget(label_db_username)
        layout.addWidget(self.input_db_username)
        layout.addWidget(label_db_password)
        layout.addWidget(self.input_db_password)
        layout.addWidget(label_admin_username)
        layout.addWidget(self.input_admin_username)
        layout.addWidget(label_admin_password)
        layout.addWidget(self.input_admin_password)
        layout.addWidget(label_no_of_two_seater)
        layout.addWidget(self.input_two_wheeler)
        layout.addWidget(label_no_of_four_seater)
        layout.addWidget(self.input_four_wheeler)
        layout.addWidget(buttonsave)
        layout.addWidget(self.error_label)

        buttonsave.clicked.connect(self.showStepInfo)

        self.setLayout(layout)
```

3.Admin Login Window:

Just after creating a database and generating password and saving the configuration we get to the admin sign in page where we can check our password and hence let us to the home page if credential are correct and display message of **Invalid login** if password is wrong.

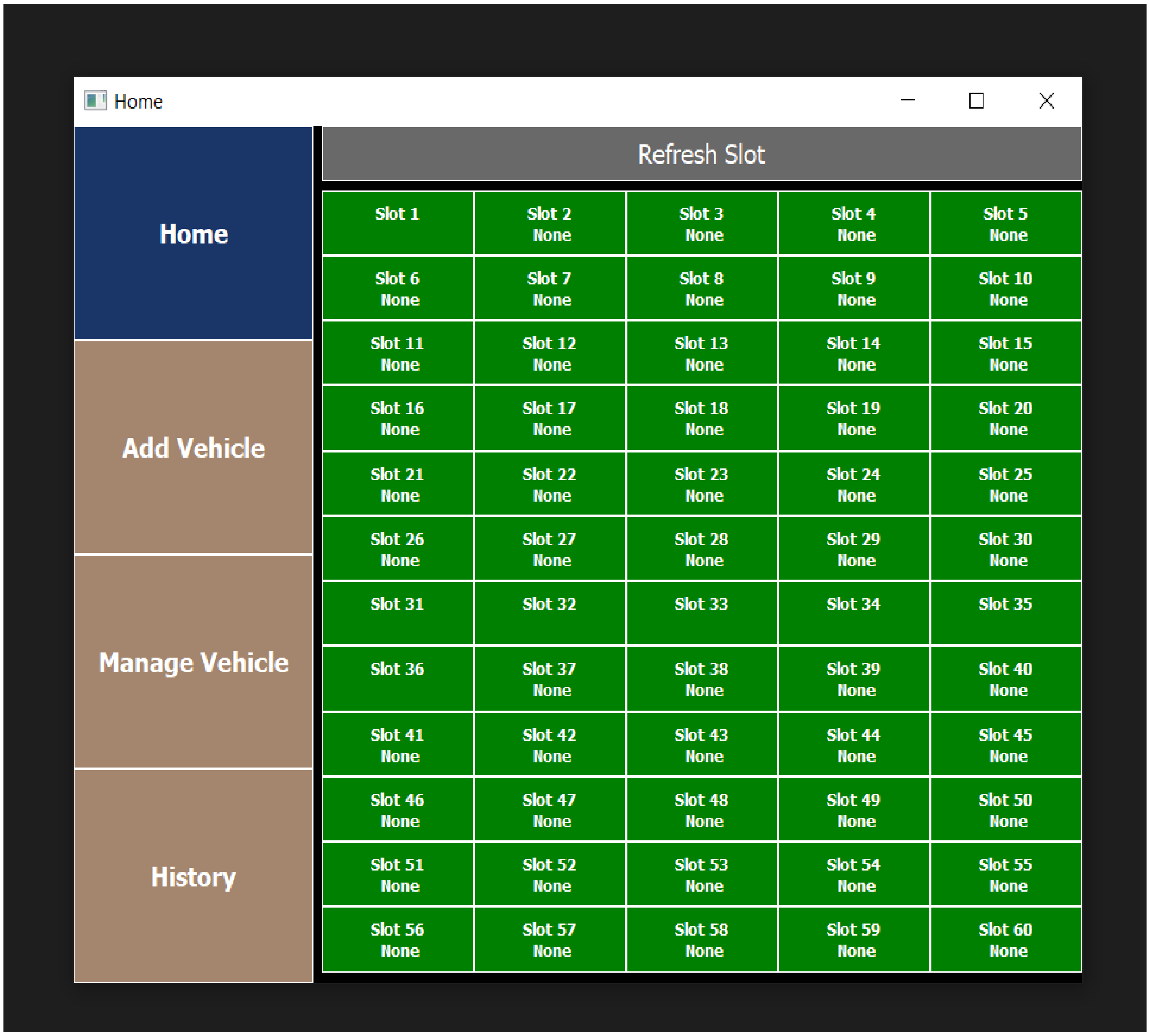




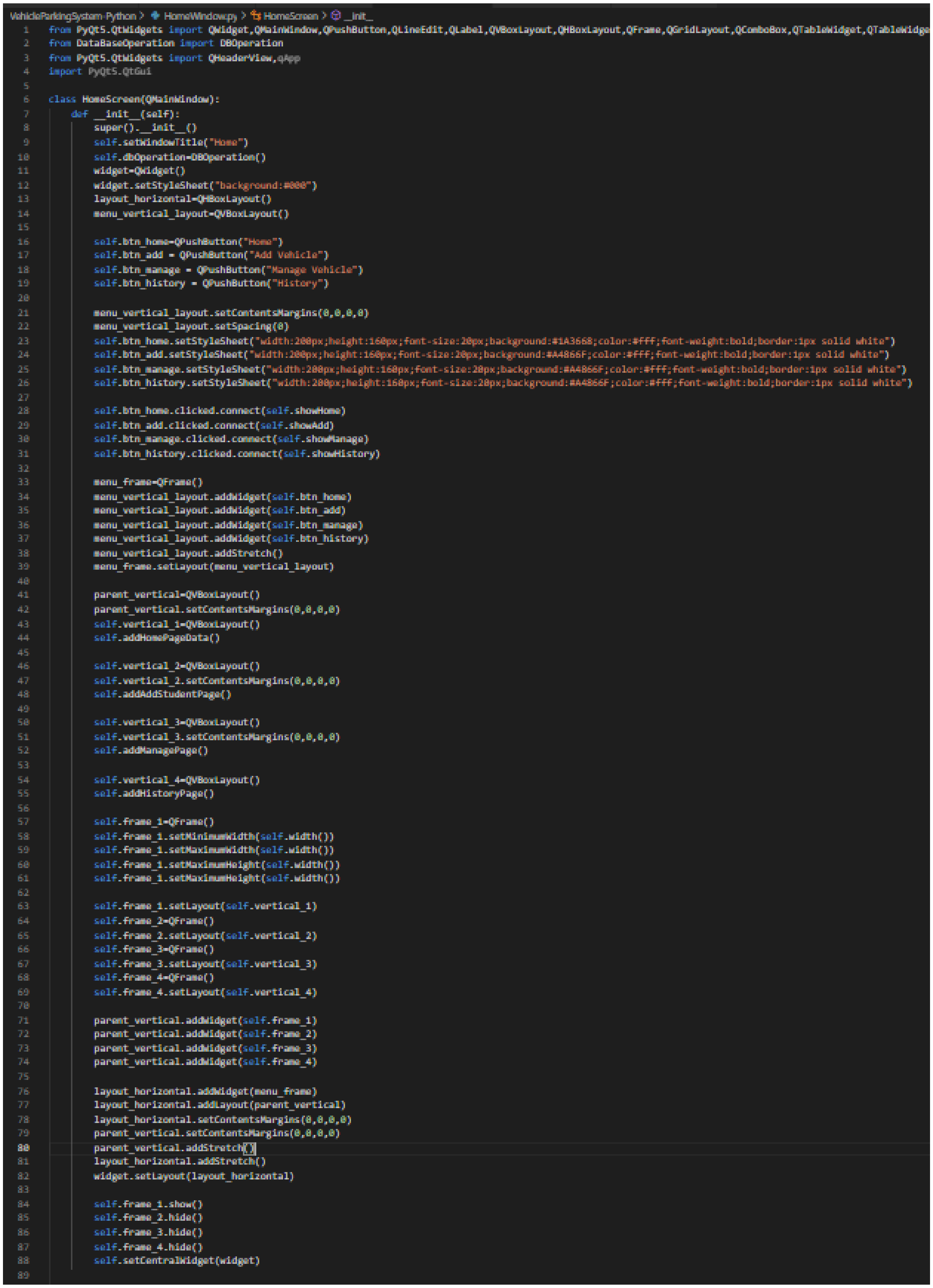
```python
#ADMIN LOGIN PAGE
from PyQt5.QtWidgets import QWidget,QVBoxLayout,QPushButton,QLabel,QLineEdit,QApplication
import sys
from DataBaseOperation import DBOperation
from HomeWindow import HomeScreen

class LoginScreen(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Admin Login")
        self.resize(300,100)
        layout=QVBoxLayout()

        label_username=QLabel("Username : ")
        label_username.setStyleSheet("color:#000;padding:8px 0px;font-size:18px;")
        self.input_username=QLineEdit()
        self.input_username.setStyleSheet("padding:5px;font-size:17px")
        label_password=QLabel("Password : ")
        label_password.setStyleSheet("color:#000;padding:8px 0px;font-size:18px;")
        self.error_msg=QLabel()
        self.error_msg.setStyleSheet("color:red;padding:8px 0px;font-size:18px;text-align:center")
        self.input_password=QLineEdit()
        self.input_password.setStyleSheet("padding:5px;font-size:17px")

        btn_login=QPushButton("Login")
        btn_login.setStyleSheet("padding:5px;font-size:20px;background:green;color:#fff")
        layout.addWidget(label_username)
        layout.addWidget(self.input_username)
        layout.addWidget(label_password)
        layout.addWidget(self.input_password)
        layout.addWidget(btn_login)
        layout.addWidget(self.error_msg)
        layout.addStretch()
        btn_login.clicked.connect(self.showHome)
        self.setLayout(layout)

    def showLoginScreen(self):
        self.show()

    def showHome(self):
        if self.input_username.text()=="":
            self.error_msg.setText("Please Enter Username")
            return

        if self.input_password.text()=="":
            self.error_msg.setText("Please Enter Password")
            return
        dboperation=DBOperation()
        result=dboperation.doAdminLogin(self.input_username.text(),self.input_password.text())
        if result:
            self.error_msg.setText("Login Successful")
            self.close()
            self.home = HomeScreen()
            self.home.show()
        else:
            self.error_msg.setText("Invalid Login Details")
```

4.Home Window:

We are now directed to the home page of the program which is completely designed using Graphic modules like PyQt5 and the database part is done using My SQL. It contains of 40 slots for 2 wheeler and 4 wheeler vehicles.



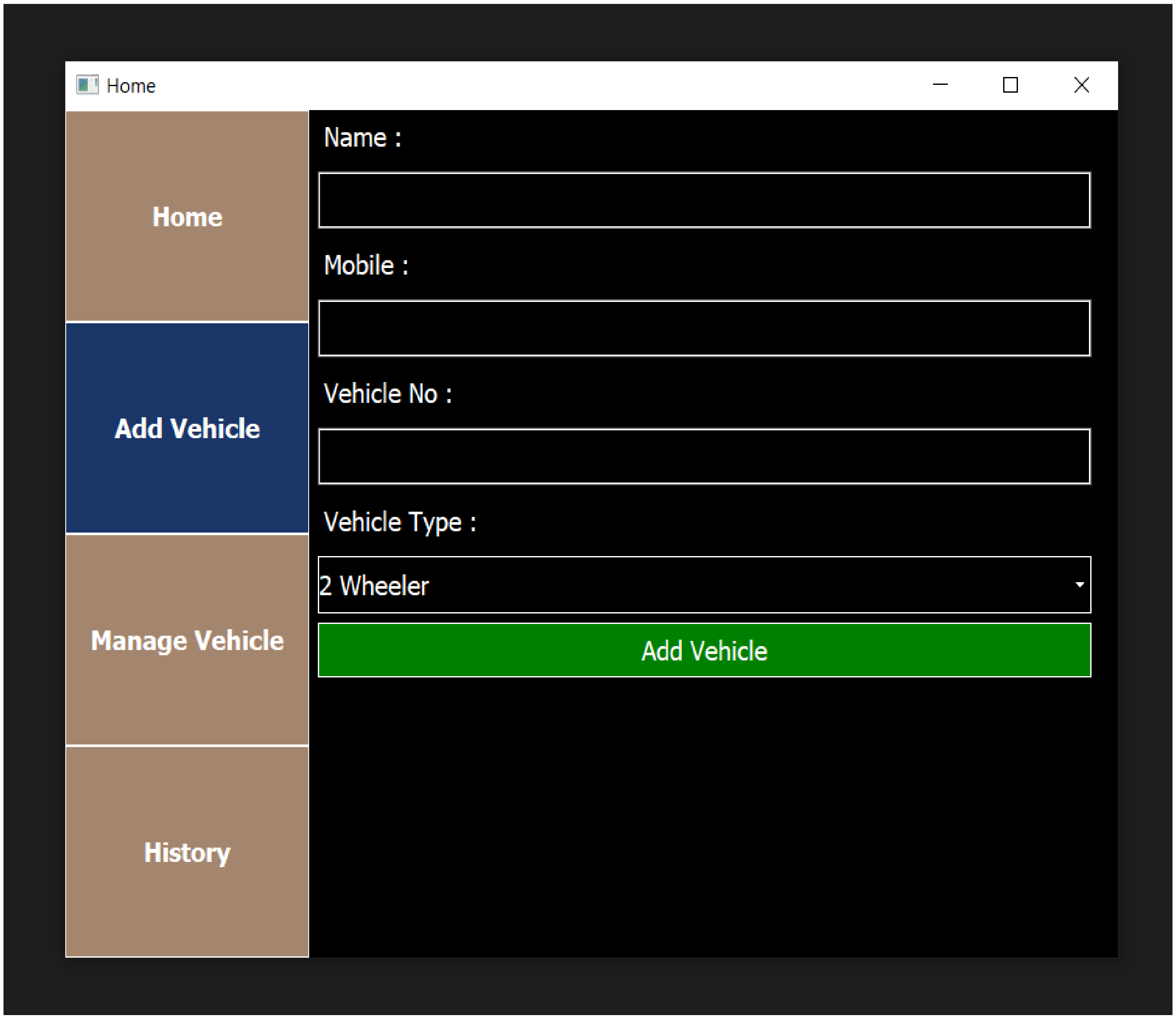
Slot 1	Slot 2 None	Slot 3 None	Slot 4 None	Slot 5 None
Slot 6 None	Slot 7 None	Slot 8 None	Slot 9 None	Slot 10 None
Slot 11 None	Slot 12 None	Slot 13 None	Slot 14 None	Slot 15 None
Slot 16 None	Slot 17 None	Slot 18 None	Slot 19 None	Slot 20 None
Slot 21 None	Slot 22 None	Slot 23 None	Slot 24 None	Slot 25 None
Slot 26 None	Slot 27 None	Slot 28 None	Slot 29 None	Slot 30 None
Slot 31	Slot 32	Slot 33	Slot 34	Slot 35
Slot 36	Slot 37 None	Slot 38 None	Slot 39 None	Slot 40 None
Slot 41 None	Slot 42 None	Slot 43 None	Slot 44 None	Slot 45 None
Slot 46 None	Slot 47 None	Slot 48 None	Slot 49 None	Slot 50 None
Slot 51 None	Slot 52 None	Slot 53 None	Slot 54 None	Slot 55 None
Slot 56 None	Slot 57 None	Slot 58 None	Slot 59 None	Slot 60 None


VehicleParkingSystem-Python > HomeWindow.py > HomeScreen > __init__

```python
from PyQt5.QtWidgets import QWidget,QMainWindow,QPushButton,QLineEdit,QLabel,QVBoxLayout,QHBoxLayout,QFrame,QGridLayout,QComboBox,QTableWidget,QTableWidge
from DataBaseOperation import DBOperation
from PyQt5.QtWidgets import QHeaderView,qApp
import PyQt5.QtGui

class HomeScreen(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Home")
        self.dbOperation=DBOperation()
        widget=QWidget()
        widget.setStyleSheet("background:#000")
        layout_horizontal=QHBoxLayout()
        menu_vertical_layout=QVBoxLayout()

        self.btn_home=QPushButton("Home")
        self.btn_add = QPushButton("Add Vehicle")
        self.btn_manage = QPushButton("Manage Vehicle")
        self.btn_history = QPushButton("History")

        menu_vertical_layout.setContentsMargins(0,0,0,0)
        menu_vertical_layout.setSpacing(0)
        self.btn_home.setStyleSheet("width:200px;height:160px;font-size:20px;background:#1A3668;color:#fff;font-weight:bold;border:1px solid white")
        self.btn_add.setStyleSheet("width:200px;height:160px;font-size:20px;background:#A4866F;color:#fff;font-weight:bold;border:1px solid white")
        self.btn_manage.setStyleSheet("width:200px;height:160px;font-size:20px;background:#A4866F;color:#fff;font-weight:bold;border:1px solid white")
        self.btn_history.setStyleSheet("width:200px;height:160px;font-size:20px;background:#A4866F;color:#fff;font-weight:bold;border:1px solid white")

        self.btn_home.clicked.connect(self.showHome)
        self.btn_add.clicked.connect(self.showAdd)
        self.btn_manage.clicked.connect(self.showManage)
        self.btn_history.clicked.connect(self.showHistory)

        menu_frame=QFrame()
        menu_vertical_layout.addWidget(self.btn_home)
        menu_vertical_layout.addWidget(self.btn_add)
        menu_vertical_layout.addWidget(self.btn_manage)
        menu_vertical_layout.addWidget(self.btn_history)
        menu_vertical_layout.addStretch()
        menu_frame.setLayout(menu_vertical_layout)

        parent_vertical=QVBoxLayout()
        parent_vertical.setContentsMargins(0,0,0,0)
        self.vertical_1=QVBoxLayout()
        self.addHomePageData()

        self.vertical_2=QVBoxLayout()
        self.vertical_2.setContentsMargins(0,0,0,0)
        self.addAddStudentPage()

        self.vertical_3=QVBoxLayout()
        self.vertical_3.setContentsMargins(0,0,0,0)
        self.addManagePage()

        self.vertical_4=QVBoxLayout()
        self.addHistoryPage()

        self.frame_1=QFrame()
        self.frame_1.setMinimumWidth(self.width())
        self.frame_1.setMaximumWidth(self.width())
        self.frame_1.setMaximumHeight(self.width())
        self.frame_1.setMaximumHeight(self.width())

        self.frame_1.setLayout(self.vertical_1)
        self.frame_2=QFrame()
        self.frame_2.setLayout(self.vertical_2)
        self.frame_3=QFrame()
        self.frame_3.setLayout(self.vertical_3)
        self.frame_4=QFrame()
        self.frame_4.setLayout(self.vertical_4)

        parent_vertical.addWidget(self.frame_1)
        parent_vertical.addWidget(self.frame_2)
        parent_vertical.addWidget(self.frame_3)
        parent_vertical.addWidget(self.frame_4)

        layout_horizontal.addWidget(menu_frame)
        layout_horizontal.addLayout(parent_vertical)
        layout_horizontal.setContentsMargins(0,0,0,0)
        parent_vertical.setContentsMargins(0,0,0,0)
        parent_vertical.addStretch()
        layout_horizontal.addStretch()
        widget.setLayout(layout_horizontal)

        self.frame_1.show()
        self.frame_2.hide()
        self.frame_3.hide()
        self.frame_4.hide()
        self.setCentralWidget(widget)

```

5. Add Vehicle:

This window take the input of customers Name, Mobile, vehicle number and the type of vehicles, through this window only we could register our vehicle for the slot numbers, where slot is generated automatically.



Name :

Mobile :

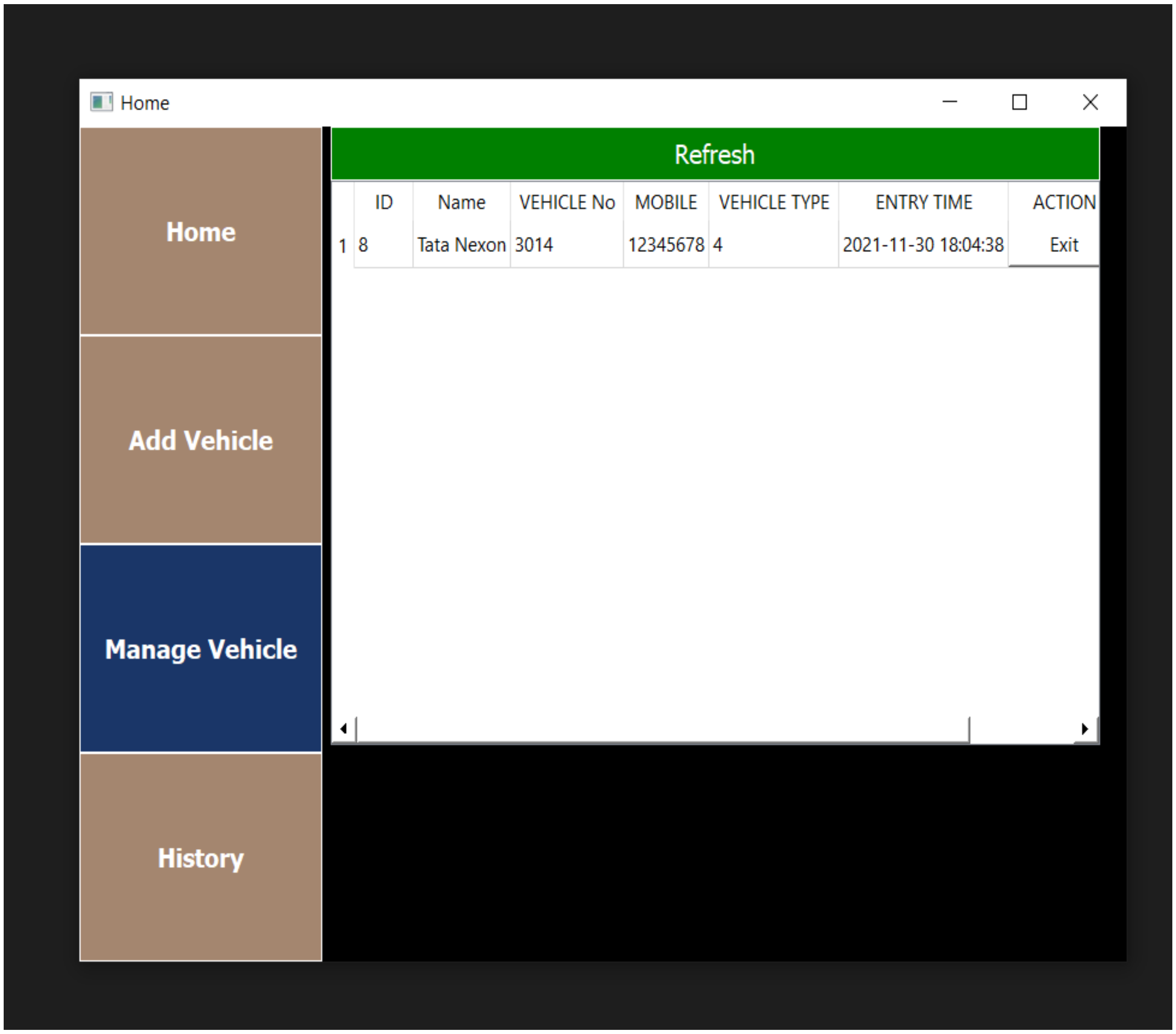
Vehicle No :

Vehicle Type :

2 Wheeler

6. Manage Vehicle:

This window indicates the registered vehicle for the respective slots, it also indicates the date and time of the vehicle when registered. We can also add and delete vehicle through this window.

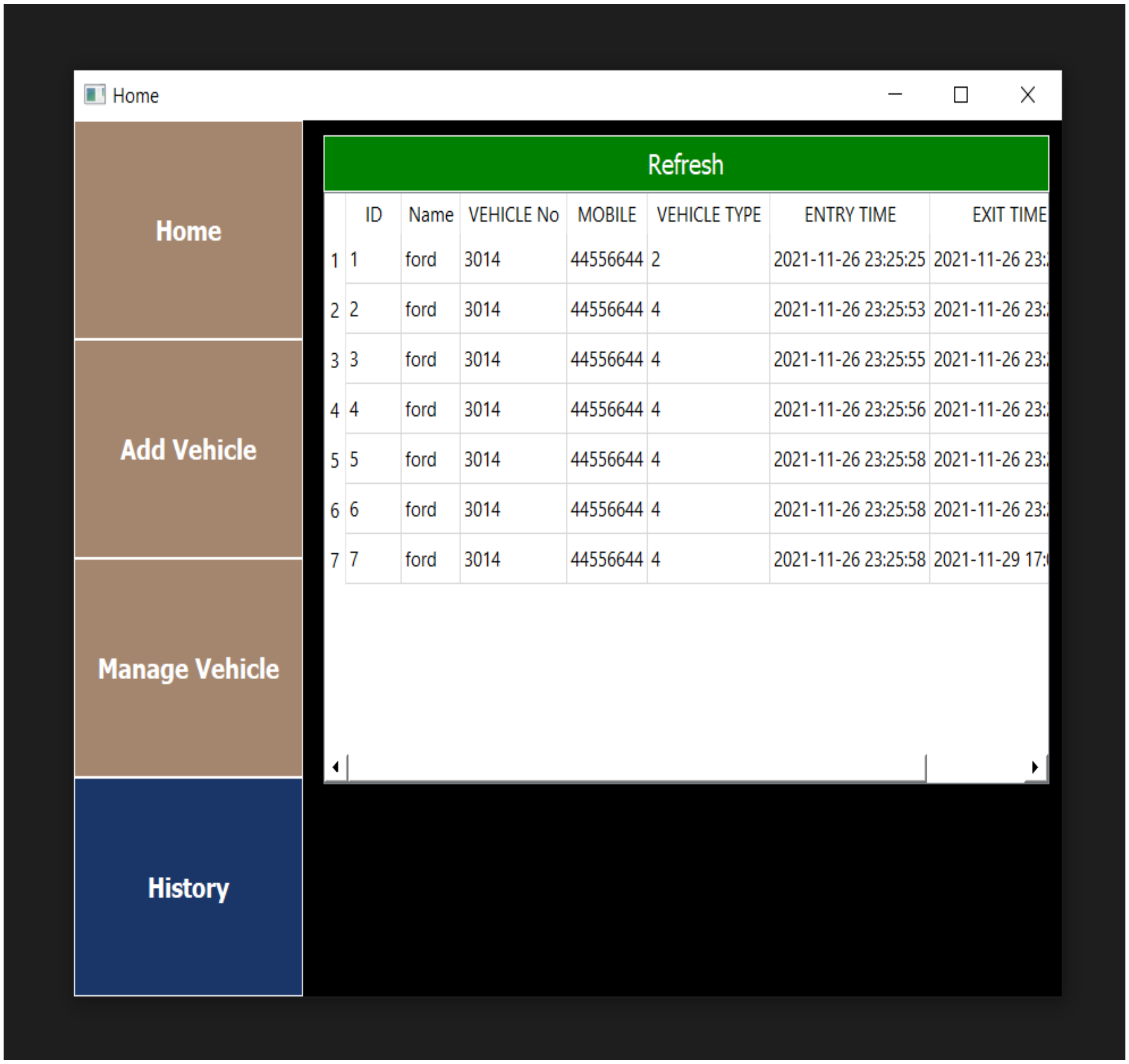


	ID	Name	VEHICLE No	MOBILE	VEHICLE TYPE	ENTRY TIME	ACTION
1	8	Tata Nexon	3014	12345678	4	2021-11-30 18:04:38	Exit

7.

7. History Window:

This window indicates the the History of the vehicles registered in past and the slot allotted to the we can anytime clear this history the navigation looks like this:-



	ID	Name	VEHICLE No	MOBILE	VEHICLE TYPE	ENTRY TIME	EXIT TIME
1	1	ford	3014	44556644	2	2021-11-26 23:25:25	2021-11-26 23:
2	2	ford	3014	44556644	4	2021-11-26 23:25:53	2021-11-26 23:
3	3	ford	3014	44556644	4	2021-11-26 23:25:55	2021-11-26 23:
4	4	ford	3014	44556644	4	2021-11-26 23:25:56	2021-11-26 23:
5	5	ford	3014	44556644	4	2021-11-26 23:25:58	2021-11-26 23:
6	6	ford	3014	44556644	4	2021-11-26 23:25:58	2021-11-26 23:
7	7	ford	3014	44556644	4	2021-11-26 23:25:58	2021-11-29 17:

Working and Importance: -

A simple project based on Vehicle Parking System which uses python language with PyQt for GUI. Following Python with PyQT project contains all the important features. It has features that will allow all the users to interact with their vehicle parking regarding their details and slot numbers. This system as well as the python application's concept is all clear, it's the same as real-life scenarios and well-implemented on it.

Moving on, this vehicle parking system project in Python focuses mainly on dealing with customer's parking details with their number, and slot. Also, the system allows inserting details of vehicle owners including their contact number, vehicle number, and vehicle category. But here, the system automatically sets a slot for reservation after inserting vehicle details until the vehicle gets out. Talking about parking slots, the system indicates empty and occupied slots with green and red color respectively. In an overview of this app, the system displays all the parked vehicles under the manage vehicles section where the user can cancel the parking once it's done.

In addition, the system displays all the parking history to date for both two and four-wheelers. Besides, the admin must set the total number of parking slots for both two and four-wheelers during the installation. On the other hand, the system captures the current time while inserting the vehicle record for parking, and then calculates the total time after cancelling the parking records. With all these time stamps, the system displays it under the history section with information such as customer's name, contact info, vehicle number, and parking dates.

Last but not least, a clean and simple GUI is presented with simple color combinations for greater user experience while using this vehicle parking management system in Python. For its UI elements, a cross-platform GUI toolkit Qt; PyQT is on board. Presenting a new vehicle parking system in Python project which includes a user panel that contains all the essential features to follow up, and a knowledgeable resource for learning purposes.

Available Features: -

- Add Vehicle Records
- Manage Vehicle Parking
- Available and Empty Parking Indicator
- List parking history
- Vehicle Category
- Total Parking Time Stamp

Instructions How to Run?

- Go to URL "http://localhost/phpmyadmin" (If you prefer using MySQL with XAMPP).
- Open the project folder where you'll find "DataBaseOperation.py" where you'll have to configure the database connection.
- Replace "localhost", "user", "passwd", "database" fields with your own.
- ***OR*** Create a Database with the given default name and import the database file (.sql) which is provided under the folder naming "DATABASE FILE". • ***Else*** run "MainProgram.py" file and fill up your configuration settings.

CONCLUSION:

Parking Management System lessens the time and boosts the efficiency of the current parking management. In overpopulated cosmopolitan zones, parking strategies must be well-implemented for efficient vehicle management.

Adopting parking management system significantly reduces the amount of time consumed in seeking the parking space, renders valuable data upon the availability of the parking area, accurate mapping of the parking space, offers guidance and suggestion for proper vehicle parking.

REFRENCES:-

Sites: -

1.https://doc.qt.io/qtforpython/overviews/stylesheet-examples.html

2.https://pythonbasics.org/pyqt-qpixmap/

3.Python | Introduction to PyQt5 - GeeksforGeeks

4.https://www.tutorialspoint.com/python/python_database_access.htm

5. datetime2 · PyPI

Books: -

- **INTRODUCTION TO PROGRAMMING USING PYTHON by Y. DANIEL LIANG, PEARSON.**
- **PYTHON PROGRAMMING: USING PROBLEM SOLVING APPROACH by REEMA THAREJA, OXFORD UNIVERSITY PRE**