|  |
| --- |
| SMART PARKING SYSTEM |
|  |
| January 27  PROJRCT: SMART PARKING SYSTEM  MEMBERS:  WALEED KHAN (2023147)  BURHAN UD DIN (2023115)  MEHTAB KHAN (2023126)  GitHub profile link:  https://github.com/waleedkhan2023147/2nd-semester-project.git  https://github.com/bu2023115/my-lab-task-5 |
|  |

PROJECT DETAILS AND TECHNOLOGY

FEATURES:

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

# PROJECT OUTPUT

INTERFACE:



This code defines a Python class named `DB Operation` that encapsulates various operations related to a parking management system. Below is a brief description of the class and its methods.

Algorithm:

Class: `DB Operation`

Constructor `\_\_init\_\_(self)`

- Initializes the class and establishes a connection to a MySQL database using credentials from a JSON configuration file (`config.json`).

Methods:

1.CreateTables(self).

- Creates three tables (`admin`, `slots`, `vehicles`) in the connected database.

- Defines the schema for each table with specific columns and data types.

2.InsertOneTimeData(self, space\_for\_two, space\_for\_four).

- Inserts initial data into the `slots` table, representing available parking spaces for two-wheelers and four-wheelers.

- Uses the `space\_for\_two` and `space\_for\_four` parameters to determine the number of spaces to insert.

3.InsertAdmin(self, username, password).

- Inserts an admin entry into the `admin` table with the specified `username` and `password`.

4.doAdminLogin(self, username, password).

- Checks the credentials provided (`username` and `password`) against entries in the `admin` table.

- Returns `True` if the credentials match an entry; otherwise, returns `False`.

5.getSlotSpace(self).

- Retrieves information about all parking slots from the `slots` table.

- Returns the fetched data.

6.getCurrentVehicle(self).

- Retrieves information about vehicles currently parked (not exited) from the `vehicles` table.

- Returns the fetched data.

7.getAllVehicle(self).

- Retrieves information about all vehicles that have exited from the parking lot from the `vehicles` table.

- Returns the fetched data.

8.AddVehicles(self, name, vehicle\_no, mobile, vehicle\_type).

- Adds a new vehicle entry to the `vehicles` table with details such as `name`, `vehicle\_no`, `mobile`, and `vehicle\_type`.

- Checks for available parking space based on `vehicle\_type`.

- Updates the corresponding slot and vehicle information.

- Returns `True` on success; otherwise, returns a message indicating no available space.

9.spaceAvailable(self, v\_type).

- Checks for available parking space of a specific type (`v\_type`) in the `slots` table.

- Returns the ID of the available slot if space is found; otherwise, returns `False`.

10.exitVehicle(self, id).

- Marks a vehicle as exited in the `vehicles` table based on the provided `id`.

- Updates the corresponding slot information in the `slots` table.

Note:

- The code interacts with a MySQL database using the `MySQL. Connector` module.

- It uses a configuration file (`config.json`) to store database connection details.

- Date and time operations are handled using the `datetime` module.