SELF SERVICE PARKING ENTRY AND EXIT MANAGEMENT SYSTEM

-Priyam Jain

PERSONAL PROJECT

INTRODUCTION

I have developed this personal project called the "**Self Service Parking Management System**" to gain practical experience and enhance my programming skills.

This Project Aim is to enhance the process of parking entry and exit with more security and efficient data management for highly important and secured areas such as government buildings or military areas or religiously or culturally reserved areas and many more . The aim is not to exchange the jobs but to enhance the security of these places moreover it can only work with coordination of security personals and managing personals. With this system, parking attendants can easily register entries, verify exits, manage the records, and generate accurate bills for the parked vehicles. It eliminates the need for manual paperwork and reduces the chances of errors, resulting in smoother operations and improved customer service.

Through this project, I have honed my programming abilities, particularly in areas such as **data handling, file management, user input validation, and time/date calculations**. By using essential programming concepts and libraries like *datetime, pickle, csv, and os,* I have implemented various functionalities that contribute to the overall efficiency of the system.

Moreover, this project has provided me with a practical understanding of how to design and develop a software application from scratch. It has allowed me to apply my knowledge of programming principles and problem-solving techniques in a meaningful way.

In summary, the Parking Bill Management System showcases my commitment to expanding my programming skills and acquiring practical knowledge. It serves as evidence of my ability to develop functional applications that automate complex processes and contribute to increased efficiency.

I would like to express my heartfelt appreciation to **Mr. Ritesh Sahu** for his invaluable guidance in teaching me the Python programming language. His comprehensive instruction and insightful explanations have equipped me with the necessary skills to independently develop this project.

Data Management and Processing in the Self Service Parking Management System:

Data Collection:

The system collects data through user input during the entry and exit processes. Users provide their name, phone number, and other relevant details. The system validates the phone number to ensure its accuracy.

Data Storage:

The collected data is stored in different files for efficient management. The entry records are stored in a binary file named "parking.dat," while the exit records are stored in another binary file named "exiting.dat." Additionally, a CSV file named "sheet.csv" is used to store records for further analysis or reporting. The data is organized using a sequential structure within these files.

Data Utilization:

The stored data is utilized in various ways within the system. For example, when generating parking bills, the system retrieves the necessary details such as the user's name, verified phone number, date, entry time, and exit time from the corresponding entry and exit records. This data is then used to generate a formatted receipt.

Data Structures and Algorithms:

The Parking Bill Management System primarily utilizes file-based data storage and retrieval. It employs the pickle module in Python to serialize and deserialize data objects when reading from or writing to binary files. This allows for efficient storage and retrieval of complex data structures, such as lists containing multiple attributes for each entry or exit record.

Additionally, the system utilizes the CSV module in Python to handle data stored in the "sheet.csv" file. This module provides functions to read and write data in CSV format, allowing for easy integration with spreadsheet applications and analysis tools.

Overall, the data management and processing in the Parking Bill Management System leverage file-based storage, serialization, and the CSV format to effectively collect, store, and utilize parking-related data. The chosen data structures and algorithms ensure efficient handling and retrieval of records for various operations within the system.

FLOW OF THE PROGRAM

The Program follows a structured flow, beginning with the main menu screen. The **main menu** presents different options based on the user's needs.

ENTRY SCREEN: Choosing the **"Entry"** option leads to the entry screen, where users input their name and phone number. The system verifies the phone number, records the entry time, and assigns a unique serial number for each entry. Once the entry is validated, a permit is granted to the user.

EXIT SCREEN: On the other hand, selecting the **"Exit"** option displays the exit screen. Users can verify their serial number, record the exit time, and generate a parking bill receipt. The receipt includes crucial details like the serial number, name, verified phone number, date, entry time, and exit time.

MANGER: Additionally, the system offers a **"Manager**" option for parking attendants or administrators to oversee various aspects of the parking system. The manager screen provides functionalities such as viewing all previous parking records (both entry and exit), manually adding an entry record, updating existing records (prior to exit), deleting records (prior to exit), viewing present entries, backing up data, and removing empty lists from the entry records.

In summary, the Parking Bill Management System provides a clear flow where users can enter and exit the parking lot, while managers have access to comprehensive management features. This streamlined approach ensures efficient parking management and facilitates the smooth operation of the system.

FUNCTIONS

The provided code is an implementation of a self-service parking management system. It allows users to enter and exit a parking lot and generates bills for their parking duration. Here's an explanation of the code:

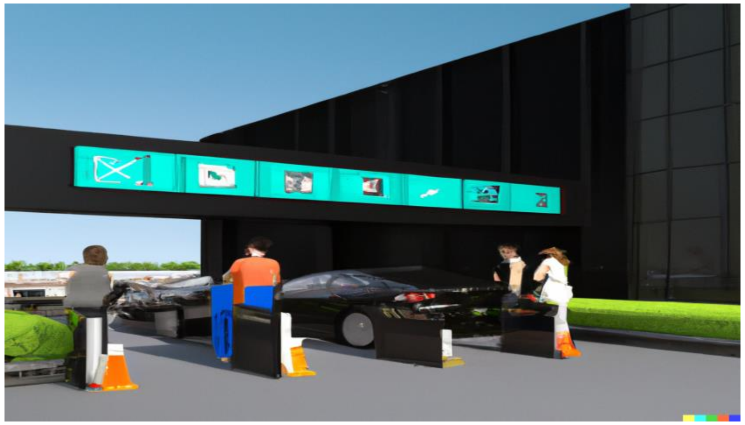
The code begins with importing necessary modules such as time, pickle, csv, datetime, and os. These modules provide functionality for time manipulation, data serialization, CSV file handling, and operating system-related operations.

The code defines several functions:

1. **date():** Retrieves the current date and returns it as a string.
2. **ptime():** Retrieves the current time and returns it as a string.
3. ***main():*** Displays the main menu and handles user input for different operations (entry, exit, managing, quit).
4. **write_heading()**: Writes the header for three files (entry records, exit records, bill records).
5. ***entry():*** Handles the entry process, collects user information (name, phone number), and stores the entry record.
6. ***exitn()*****:** Handles the exit process, verifies the user's entry record, collects the exit time, and generates a bill.
7. **deco():** Prints a decorative line separator.
8. **manager():** Displays the management menu and handles different management operations (view records, add entry manually, update record, delete record, view present entries, backup data, remove empty lists, return to main menu).
9. **sno():** Generates a unique serial number for the entry record.
10. **checks(s):** Checks if the given serial number exists in the entry records and verifies that the user has not already exited

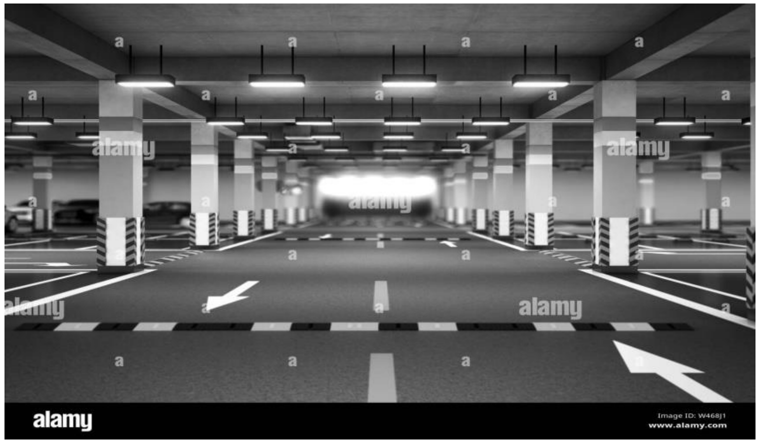
11. **read()**: Reads and displays all previous parking records (entry and exit).
12. **write()**: Manually adds an entry record by collecting user information and storing it.
13. **backup():** Creates a backup of the entry records.
14. **delete(sno)**: Deletes a specific record by serial number from the entry records.
15. **delete_empty_list():** Deletes any empty lists from the entry records.
16. **update()**: Updates a specific record in the entry records with new information.
17. **readp():** Reads and displays all present entry records.

Representation of self service parking management system



PARKING ENTRY

USER REGISTERING FOR ENTRY



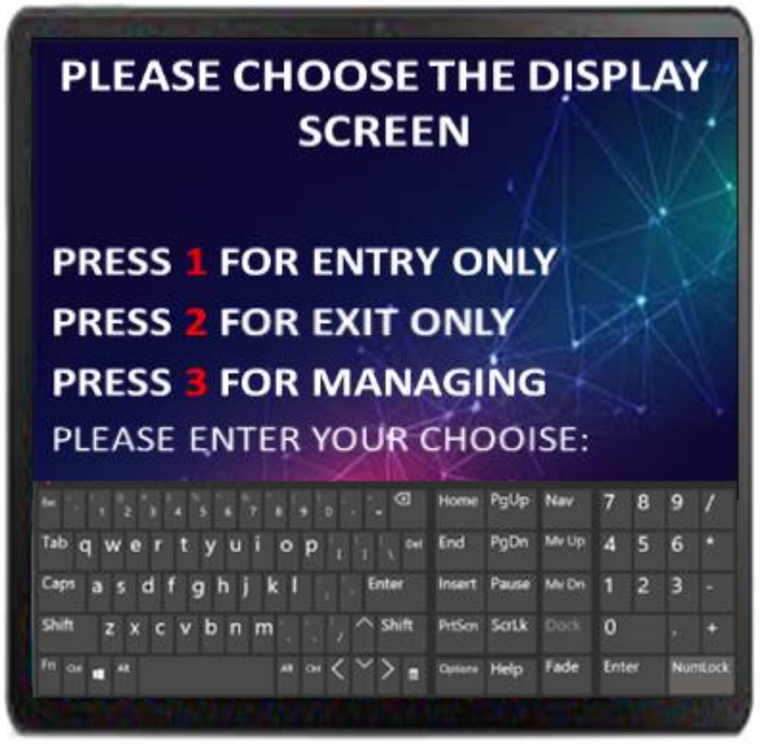
PARKING AREA

USER **EXIT**ING THE PARKING AND COLLECTING RECEIPT

MAIN MENU

WITH GRAPHICAL USER INTERFACE

CONSOLE VIEW



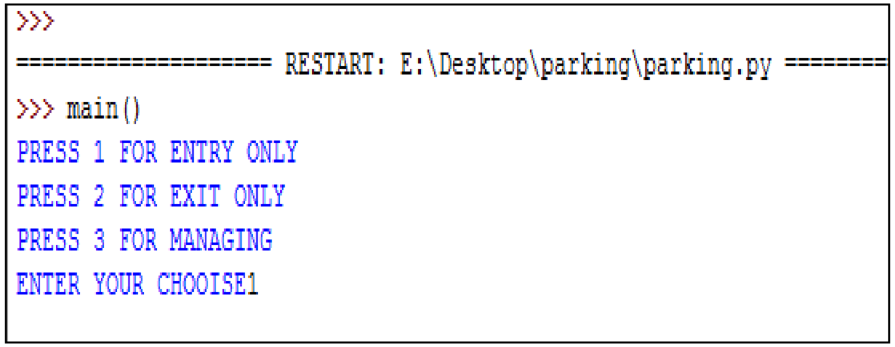
PLEASE CHOOSE THE DISPLAY SCREEN

PRESS 1 FOR ENTRY ONLY

PRESS 2 FOR EXIT ONLY

PRESS 3 FOR MANAGING

PLEASE ENTER YOUR CHOOISE:



```
>>>
==================== RESTART: E:\Desktop\parking\parking.py ========
>>> main()
PRESS 1 FOR ENTRY ONLY
PRESS 2 FOR EXIT ONLY
PRESS 3 FOR MANAGING
ENTER YOUR CHOOISE1
```

main(): Displays the main menu and handles user input for different operations (entry, exit, managing, quit).

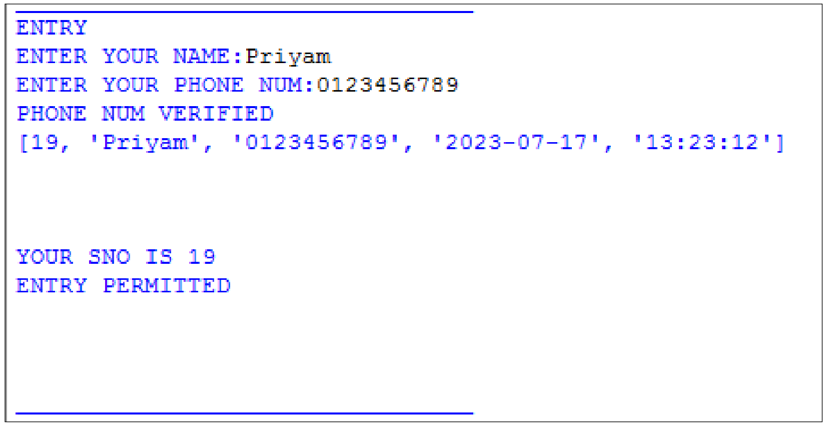
ENTRY SCREEN

PARKING ENTRY

Name:

Phone No:

Please enter you name and phone number to get the entry.



```
ENTRY
ENTER YOUR NAME:Priyam
ENTER YOUR PHONE NUM:0123456789
PHONE NUM VERIFIED
[19, 'Priyam', '0123456789', '2023-07-17', '13:23:12']


YOUR SNO IS 19
ENTRY PERMITTED
```

PARKING ENTRY

PHONE NO VERIFIED

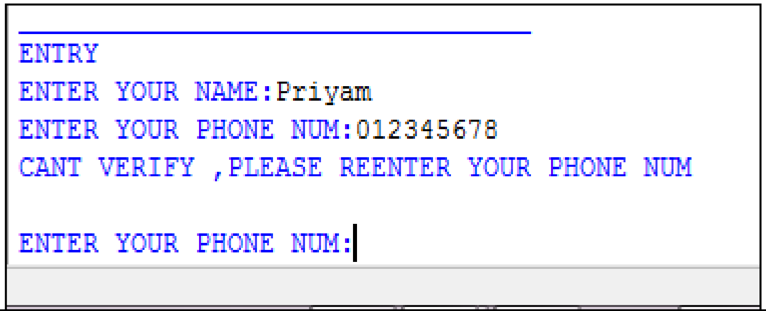
Name: xyz

Phone No: abc

Your serial number is **01**

Serial Num Is Required During Exit

ENTRY PERMITTED

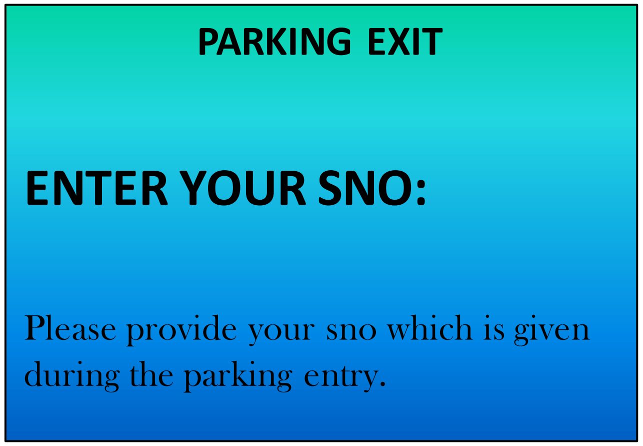


```
ENTRY
ENTER YOUR NAME:Priyam
ENTER YOUR PHONE NUM:012345678
CANT VERIFY ,PLEASE REENTER YOUR PHONE NUM

ENTER YOUR PHONE NUM:
```

if the phone num doesn’t exist or banned for entry

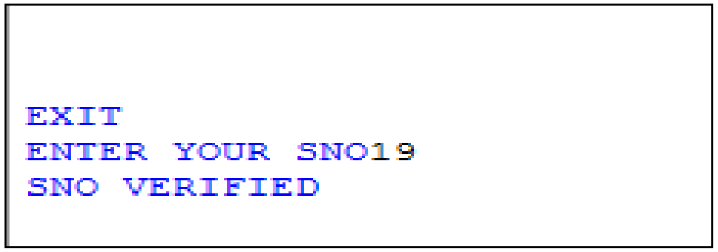
ENTRY SCREEN: Choosing the **"Entry"** option leads to the entry screen, where users input their name and phone number. The system verifies the phone number, records the entry time, and assigns a unique serial number for each entry. Once the entry is validated, a permit is granted to the user.

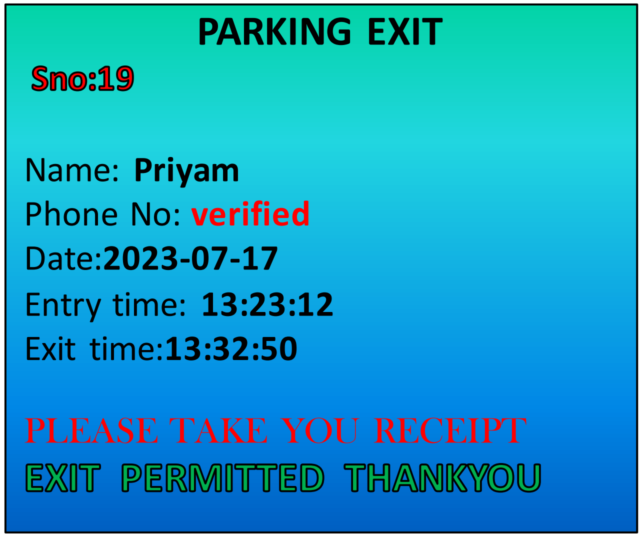
EXIT SCREEN


PARKING EXIT

ENTER YOUR SNO:

Please provide your sno which is given during the parking entry.

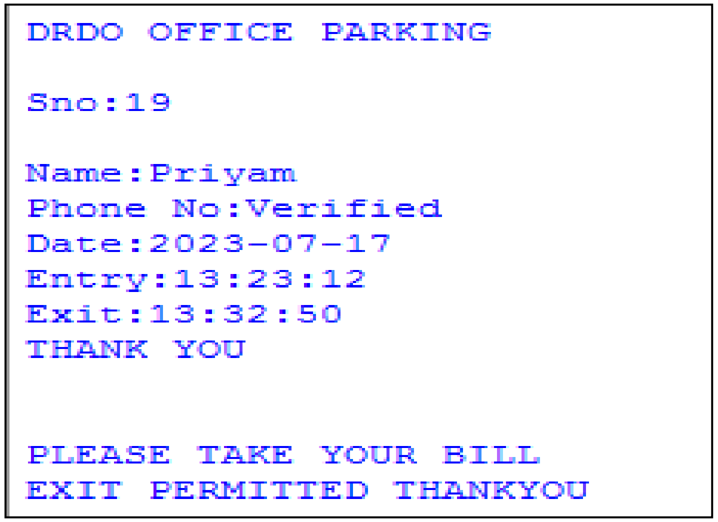

```
EXIT
ENTER YOUR SNO19
SNO VERIFIED
```


PARKING EXIT

Sno:19

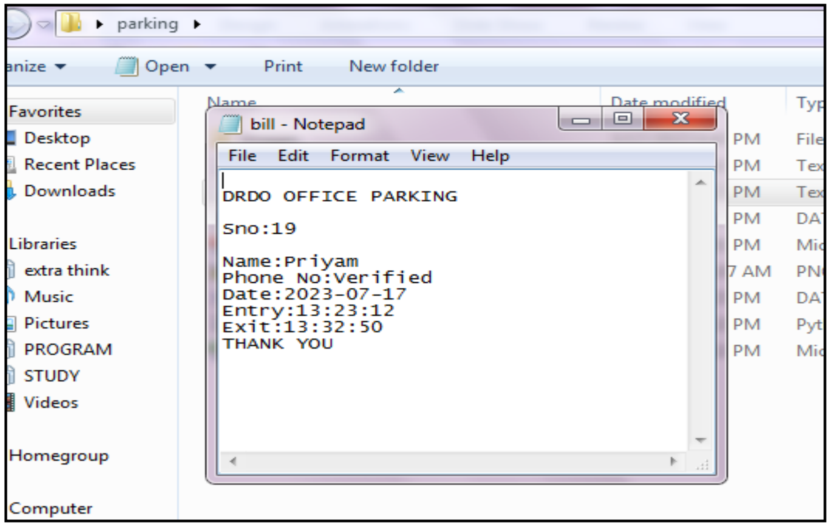
Name: Priyam
Phone No: verified
Date:2023-07-17
Entry time: 13:23:12
Exit time:13:32:50

PLEASE TAKE YOU RECEIPT
EXIT PERMITTED THANKYOU


```
DRDO OFFICE PARKING

Sno:19

Name:Priyam
Phone No:Verified
Date:2023-07-17
Entry:13:23:12
Exit:13:32:50
THANK YOU


PLEASE TAKE YOUR BILL
EXIT PERMITTED THANKYOU
```

EXIT SCREEN: On the other hand, selecting the **"Exit"** option displays the exit screen. Users can verify their serial number, record the exit time, and generate a parking bill receipt. The receipt includes crucial details like the serial number, name, verified phone number, date, entry time, and exit time.

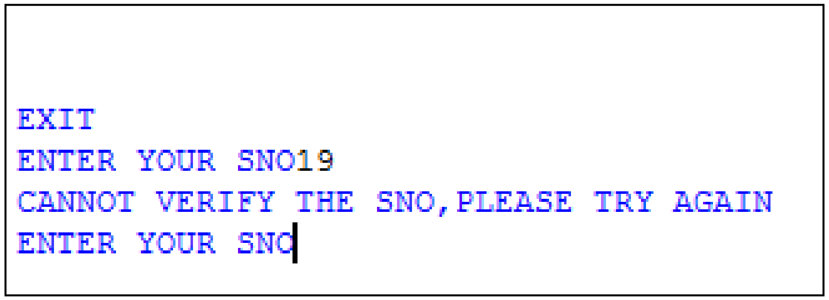
RECEIPT PRINTING



```
DRDO OFFICE PARKING

Sno:19

Name:Priyam
Phone No:Verified
Date:2023-07-17
Entry:13:23:12
Exit:13:32:50
THANK YOU
```

With the help of f.write() function ,The bill will get saved as a text file and then printed in the form of receipt .

SECURITY



```
EXIT
ENTER YOUR SNO19
CANNOT VERIFY THE SNO,PLEASE TRY AGAIN
ENTER YOUR SNO
```

same serial number cannot be used twice to exit , for trial the serial numbers are given simple.

Managing screen

MANAGING SCREEN

PRESS 1 FOR READING RECORDS

PRESS 2 FOR WRITING MANUALLY

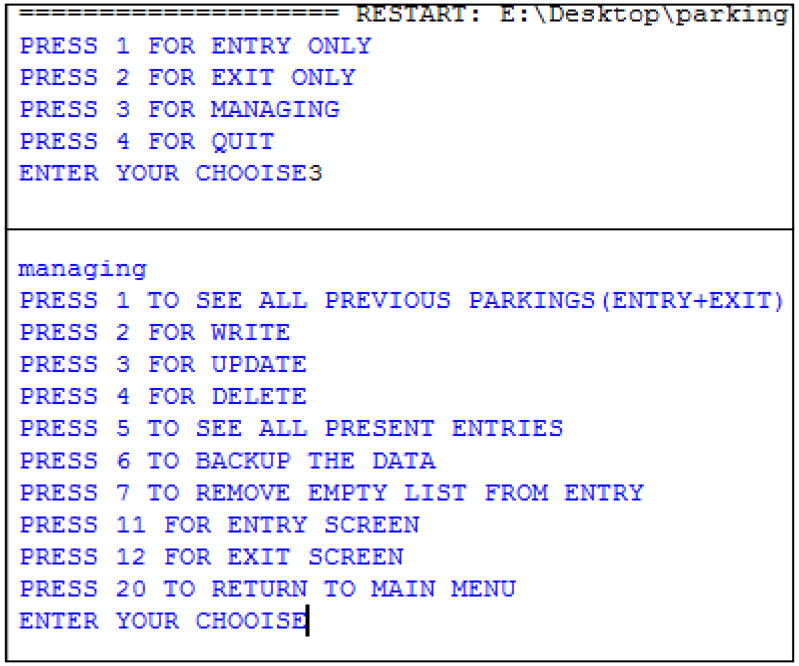
PRESS 3 FOR UPDATING

PRESS 4 FOR DELETING

PRESS 5 TO OPEN ENTRY SCREEN

PRESS 6 TO OPEN EXIT SCREEN

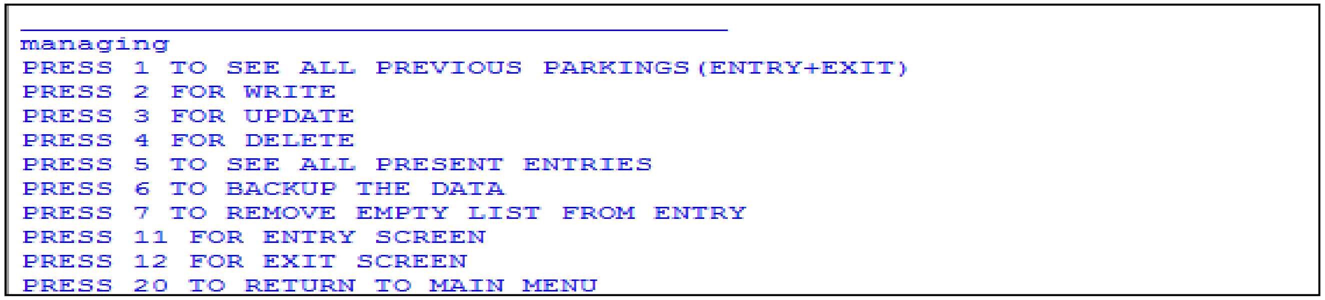
PLEASE ENTER YOUR CHOOISE:



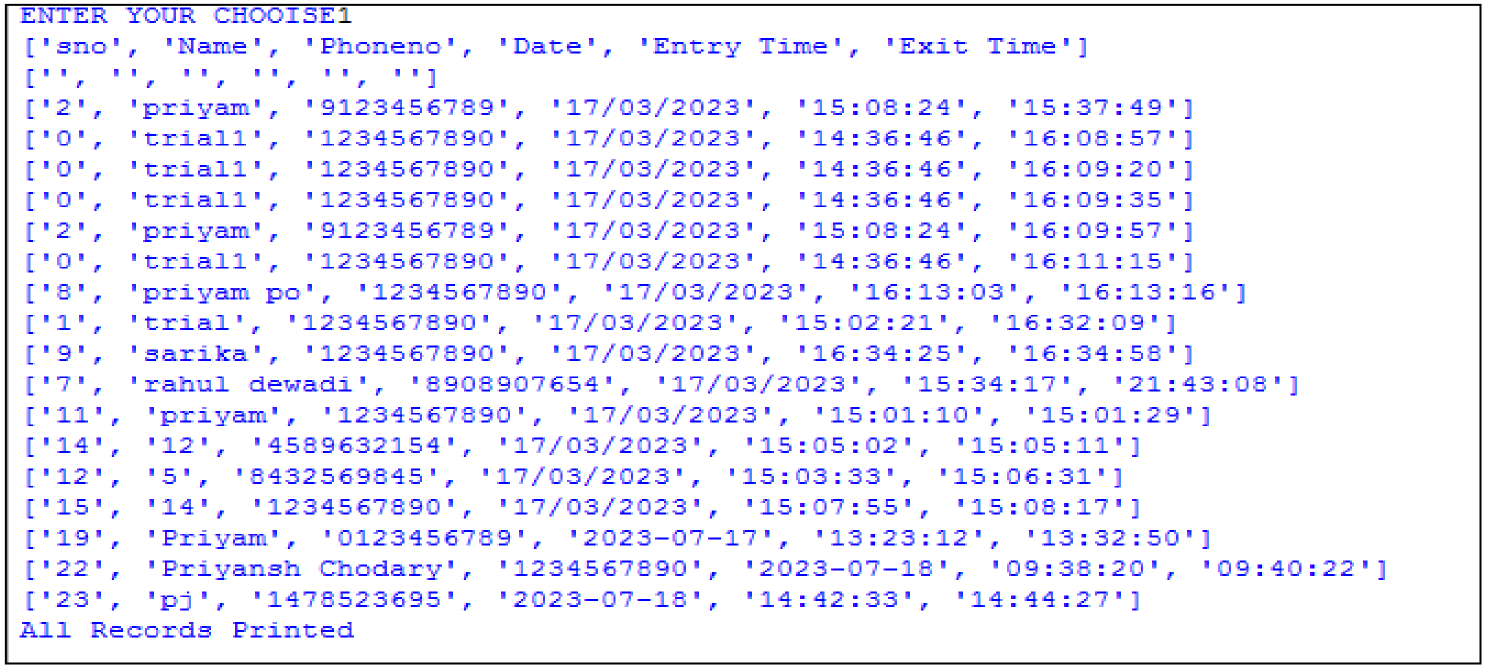
```
==================== RESTART: E:\Desktop\parking
PRESS 1 FOR ENTRY ONLY
PRESS 2 FOR EXIT ONLY
PRESS 3 FOR MANAGING
PRESS 4 FOR QUIT
ENTER YOUR CHOOISE3

managing
PRESS 1 TO SEE ALL PREVIOUS PARKINGS(ENTRY+EXIT)
PRESS 2 FOR WRITE
PRESS 3 FOR UPDATE
PRESS 4 FOR DELETE
PRESS 5 TO SEE ALL PRESENT ENTRIES
PRESS 6 TO BACKUP THE DATA
PRESS 7 TO REMOVE EMPTY LIST FROM ENTRY
PRESS 11 FOR ENTRY SCREEN
PRESS 12 FOR EXIT SCREEN
PRESS 20 TO RETURN TO MAIN MENU
ENTER YOUR CHOOISE
```

MANGER: Additionally, the system offers a "**Manager**" option for parking attendants or administrators to oversee various aspects of the parking system. The manager screen provides functionalities such as viewing all previous parking records (both entry and exit), manually adding an entry record, updating existing records (prior to exit), deleting records (prior to exit), viewing present entries, backing up data, and removing empty lists from the entry records.

Reading Records



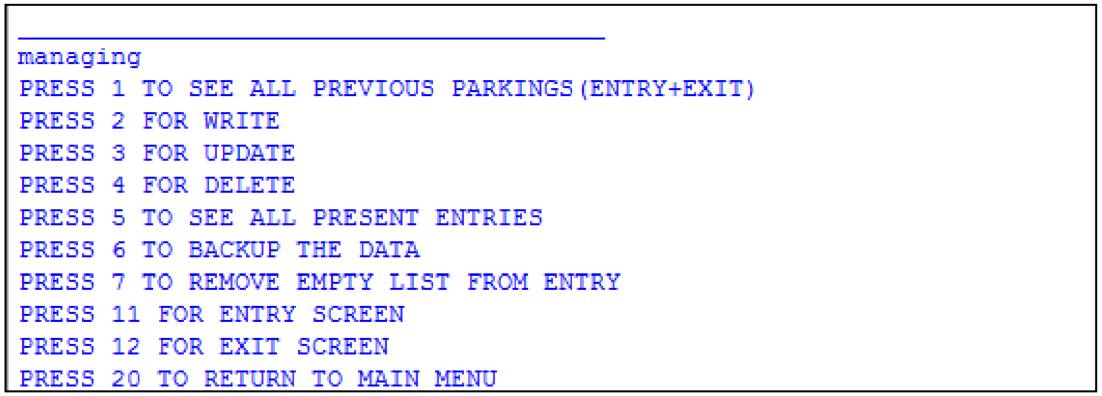
```
managing
PRESS 1 TO SEE ALL PREVIOUS PARKINGS(ENTRY+EXIT)
PRESS 2 FOR WRITE
PRESS 3 FOR UPDATE
PRESS 4 FOR DELETE
PRESS 5 TO SEE ALL PRESENT ENTRIES
PRESS 6 TO BACKUP THE DATA
PRESS 7 TO REMOVE EMPTY LIST FROM ENTRY
PRESS 11 FOR ENTRY SCREEN
PRESS 12 FOR EXIT SCREEN
PRESS 20 TO RETURN TO MAIN MENU
```



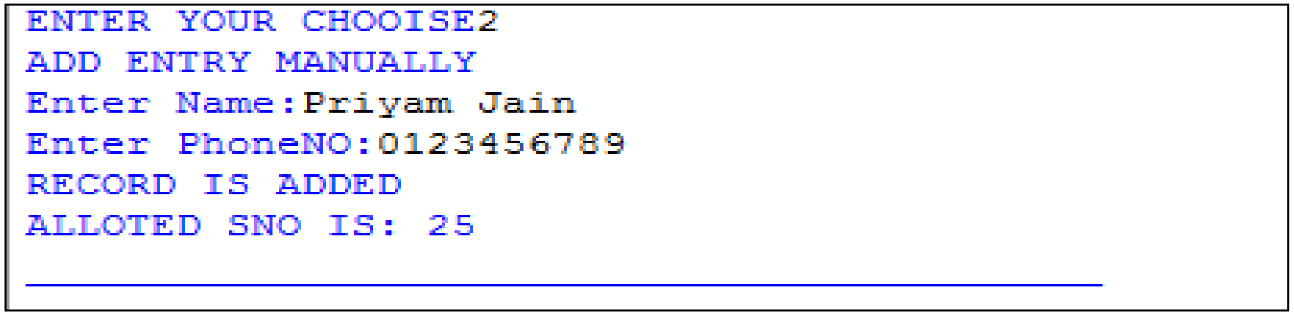
```
ENTER YOUR CHOOISE1
['sno', 'Name', 'Phoneno', 'Date', 'Entry Time', 'Exit Time']
['', '', '', '', '', '']
['2', 'priyam', '9123456789', '17/03/2023', '15:08:24', '15:37:49']
['0', 'trial1', '1234567890', '17/03/2023', '14:36:46', '16:08:57']
['0', 'trial1', '1234567890', '17/03/2023', '14:36:46', '16:09:20']
['0', 'trial1', '1234567890', '17/03/2023', '14:36:46', '16:09:35']
['2', 'priyam', '9123456789', '17/03/2023', '15:08:24', '16:09:57']
['0', 'trial1', '1234567890', '17/03/2023', '14:36:46', '16:11:15']
['8', 'priyam po', '1234567890', '17/03/2023', '16:13:03', '16:13:16']
['1', 'trial', '1234567890', '17/03/2023', '15:02:21', '16:32:09']
['9', 'sarika', '1234567890', '17/03/2023', '16:34:25', '16:34:58']
['7', 'rahul dewadi', '8908907654', '17/03/2023', '15:34:17', '21:43:08']
['11', 'priyam', '1234567890', '17/03/2023', '15:01:10', '15:01:29']
['14', '12', '4589632154', '17/03/2023', '15:05:02', '15:05:11']
['12', '5', '8432569845', '17/03/2023', '15:03:33', '15:06:31']
['15', '14', '1234567890', '17/03/2023', '15:07:55', '15:08:17']
['19', 'Priyam', '0123456789', '2023-07-17', '13:23:12', '13:32:50']
['22', 'Priyansh Chodary', '1234567890', '2023-07-18', '09:38:20', '09:40:22']
['23', 'pj', '1478523695', '2023-07-18', '14:42:33', '14:44:27']
All Records Printed
```

Reads and displays all previous parking records (entry and exit).

Writing Records Manually



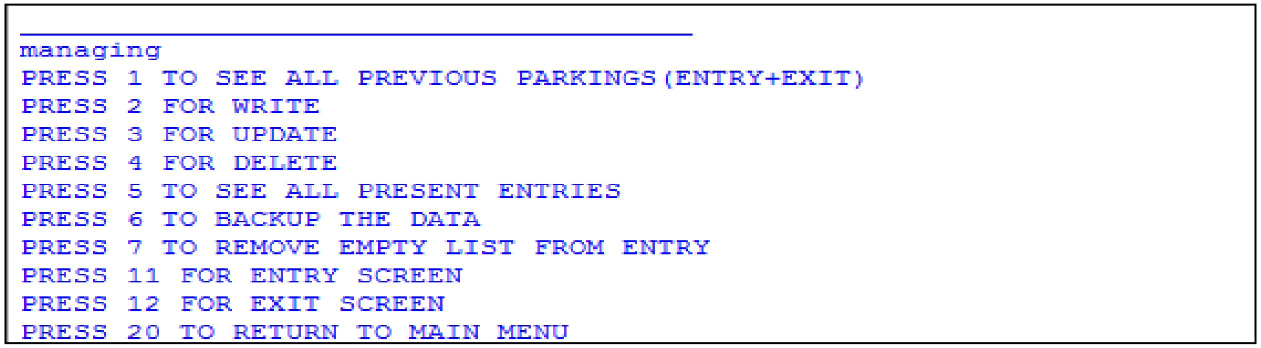
```
___________________________________________
managing
PRESS 1 TO SEE ALL PREVIOUS PARKINGS(ENTRY+EXIT)
PRESS 2 FOR WRITE
PRESS 3 FOR UPDATE
PRESS 4 FOR DELETE
PRESS 5 TO SEE ALL PRESENT ENTRIES
PRESS 6 TO BACKUP THE DATA
PRESS 7 TO REMOVE EMPTY LIST FROM ENTRY
PRESS 11 FOR ENTRY SCREEN
PRESS 12 FOR EXIT SCREEN
PRESS 20 TO RETURN TO MAIN MENU
```



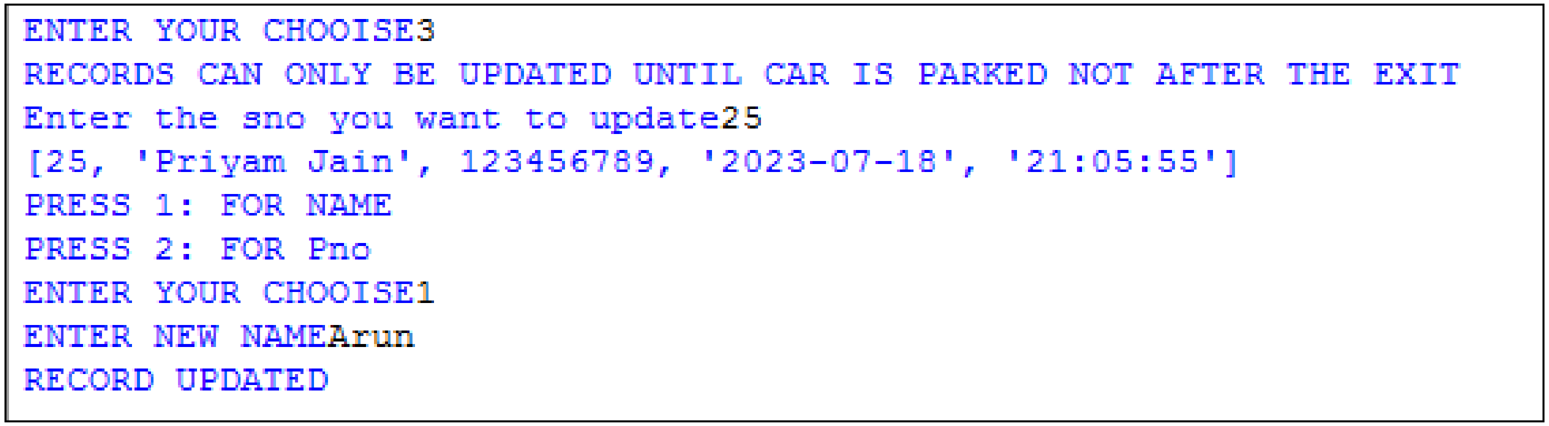
```
ENTER YOUR CHOOISE2
ADD ENTRY MANUALLY
Enter Name:Priyam Jain
Enter PhoneNO:0123456789
RECORD IS ADDED
ALLOTED SNO IS: 25

___________________________________________________________
```

Manually adds an entry record by collecting user information and storing it.

Updating the records



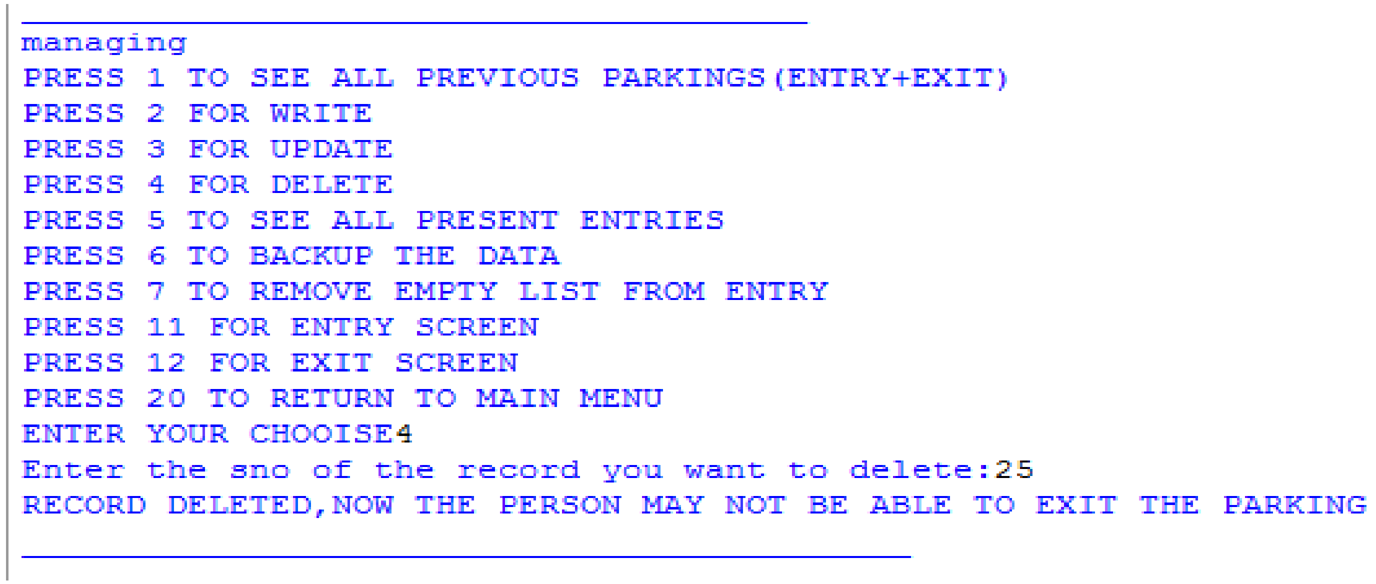
```
managing
PRESS 1 TO SEE ALL PREVIOUS PARKINGS(ENTRY+EXIT)
PRESS 2 FOR WRITE
PRESS 3 FOR UPDATE
PRESS 4 FOR DELETE
PRESS 5 TO SEE ALL PRESENT ENTRIES
PRESS 6 TO BACKUP THE DATA
PRESS 7 TO REMOVE EMPTY LIST FROM ENTRY
PRESS 11 FOR ENTRY SCREEN
PRESS 12 FOR EXIT SCREEN
PRESS 20 TO RETURN TO MAIN MENU
```



```
ENTER YOUR CHOOISE3
RECORDS CAN ONLY BE UPDATED UNTIL CAR IS PARKED NOT AFTER THE EXIT
Enter the sno you want to update25
[25, 'Priyam Jain', 123456789, '2023-07-18', '21:05:55']
PRESS 1: FOR NAME
PRESS 2: FOR Pno
ENTER YOUR CHOOISE1
ENTER NEW NAMEArun
RECORD UPDATED
```

Updates a specific record in the entry records with new information.

For deleting the record



```
managing
PRESS 1 TO SEE ALL PREVIOUS PARKINGS(ENTRY+EXIT)
PRESS 2 FOR WRITE
PRESS 3 FOR UPDATE
PRESS 4 FOR DELETE
PRESS 5 TO SEE ALL PRESENT ENTRIES
PRESS 6 TO BACKUP THE DATA
PRESS 7 TO REMOVE EMPTY LIST FROM ENTRY
PRESS 11 FOR ENTRY SCREEN
PRESS 12 FOR EXIT SCREEN
PRESS 20 TO RETURN TO MAIN MENU
ENTER YOUR CHOOISE4
Enter the sno of the record you want to delete:25
RECORD DELETED,NOW THE PERSON MAY NOT BE ABLE TO EXIT THE PARKING
```

Deletes a specific record by serial number from the entry records.

Other options on managing screen

- **backup():** Creates a backup of the entry records.
- **delete_empty_list():** Deletes any empty lists from the entry records
- ***entry():*** Handles the entry process, collects user information (name, phone number), and stores the entry record, from manager screen.
- ***exitn()*:** Handles the exit process, verifies the user's entry record, collects the exit time, and generates a bill,from manager screen.

To run the code

- To run the code get the code from here, you can copy it into a Python IDE or text editor and run it from there. Make sure you have the necessary dependencies installed, such as pickle, csv, and datetime.
- However, please note that running the code as-is may result in errors if the required file paths (fw, fb, fbe, fc, ft) are not valid or if the directories and files mentioned in the code do not exist in your local system.
- It's recommended to review the code, ensure the file paths are correctly set according to your system's directory structure, and make any necessary modifications or adjustments before running it.
- If you encounter any specific errors or have further questions about the code, please let me know, and I'll do my best to assist you.

Limitations and Considerations

Although I have provided a graphical view of the front end in my project, I want to clarify that, as a beginner, the current implementation will primarily generate outputs displayed in the console or terminal. This is because I have focused on developing the backend functionality and core features of the project.

I understand the importance of a visually appealing user interface and plan to incorporate it as my skills progress. However, I assure you that despite the console or terminal-based outputs, I am dedicated to delivering a functional and efficient project.
If you have any suggestions or feedback on how to enhance the user experience within these limitations, I would greatly appreciate your input.

Although the program's features, such as phone number verification, are mentioned in the project description, it's important to note that as a beginner, the implementation of advanced functionalities like phone number verification may not be fully developed in the code. However, the project showcases your understanding of the concept and your intent to incorporate such features in a real-world scenario.

As a beginner, it is natural to focus on learning and applying fundamental programming concepts rather than implementing complex functionalities. The project serves as a stepping stone in your programming journey, allowing you to gain practical experience and develop a strong foundation in software development.

It takes a lot of time and efforts to explain the project efficiently ,to reduce it , I took the help from **Chat Gpt** it guided me and helped to explain this project more efficiently

Thank you for your understanding and support.