

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590 018



A Mini -Project Work on

“Driving School Management System”

A Dissertation work submitted in partial fulfillment of the requirement
for the award of the degree

Bachelor of Engineering
In
Information Science & Engineering

Submitted by

Deeksha Sudheer Nayak
H R Prathigna

1AY20IS029
1AY20IS039

Under the guidance of

Dr. Umapathi G R

Associate Professor



DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
ACHARYA INSTITUTE OF TECHNOLOGY

(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI. APPROVED BY AICTE, NEW DELHI,
ACCREDITED BY NAAC, NEW DELHI)

Acharya Dr. Sarvepalli Radhakrishnan Road, Soldevanahalli, Bengaluru-560107

2022-23

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING ACHARYA INSTITUTE OF TECHNOLOGY

(AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI. APPROVED BY AICTE, NEW DELHI, ACCREDITED BY NAAC, NEW DELHI)

Acharya Dr. Sarvepalli Radhakrishnan Road, Soldevanahalli, Bengaluru-560107



Certificate

This is to certify that the **File Structure Laboratory with Mini Project (18ISL67)** entitled **"Driving School Management System"** is a bonafide work carried out by **Deeksha Sudheer Nayak (1AY20IS029)** and **H R Prathigna (1AY20IS039)** in partial fulfillment for the award of the degree of **Bachelor of Engineering in Information Science and Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2022-23. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The mini project has been approved as it satisfies the academic requirements in respect of mini project work prescribed for the Bachelor of Engineering Degree.

Dr. Umapathi G R
Associate Professor,
Department of ISE

Prof. Kala Venugopal
HOD-ISE

Name of the Examiners

1. _____
2. _____

Signature with date

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this mini-project would be incomplete without the mention of the people who made it possible through constant guidance and encouragement.

We would take this opportunity to express our heartfelt gratitude to **Sri. B. Premnath Reddy**, Chairman, Acharya Institutes and **Dr. Rajath Hegde M M**, Principal, and **Prof. C K Marigowda**, Vice Principal, Acharya Institute of Technology for providing the necessary infrastructure to complete this mini-project.

We wish to express our deepest gratitude and thanks to **Prof. Kala Venugopal**, Head of the Department, Information Science and Engineering.

We wish to express sincere thanks to my guide **Dr. Umapathi G R**, Associate Professor, Department of Information Science and Engineering for helping us throughout and guiding us from time to time.

A warm thanks to all the faculty of Department of Information Science and Engineering, who have helped us with their views and encouraging ideas.

ABSTRACT

The driving school management system using file structure is a software solution designed to revolutionize the management of administrative tasks and data in a driving school environment. This system leverages a hierarchical file structure organization to streamline operations, enhance data management, and improve the overall effectiveness of driver education.

By adopting a file structure-based approach, the driving school management system enables efficient storage, retrieval, and modification of various types of data. It encompasses student information, instructor details, scheduling data, payment records, course materials, and other important documents. This hierarchical organization ensures easy access and seamless handling of information, eliminating the hassles of manual paperwork.

Key features of the system include comprehensive student management, dynamic scheduling and calendar functionality, streamlined instructor management, accurate billing and payment tracking, efficient course management, seamless communication and notifications, insightful reporting and analytics, and secure document storage. These features empower driving schools to efficiently handle student enrolments, monitor student progress, schedule driving lessons and theory classes, assign instructors, generate invoices, process payments, and store essential documents such as contracts and licenses.

Implementing the driving school management system using file structure brings numerous advantages. It enhances organization, simplifies data management, and streamlines workflows. The system eliminates the risk of data loss or misplacement, offering quick and convenient access to pertinent information.

Furthermore, the system provides communication and notification capabilities to facilitate seamless interaction among driving schools, instructors, and students. It generates comprehensive reports and analytics, empowering driving schools to monitor performance, evaluate instructor effectiveness, track student progress, and make data-driven decisions for continuous improvement. The driving school management system using file structure is scalable and customizable, allowing driving schools to adapt it to their specific requirements and integrate it with other tools and systems as needed. By embracing this innovative solution, driving schools can effectively manage their operations, optimize data management, and deliver exceptional driver education services.

TABLE OF CONTENTS

Acknowledgement	i
Abstract	ii
1. Introduction	1
1.1 Introduction	1
1.2 Aim of the project	1
1.3 Main Purpose	2
1.4 Goals and Objectives	2
2. Getting Started	3
2.1 Problem Statement	3
2.2 Proposed System	3
2.3 Advantages	4
2.4 Project Applications	4
3. System Requirements	5
3.1 Hardware Requirements	5
3.2 Software Requirements	5
4. System Design	6
4.1 Flow Chart	6
4.2 Data Flow Diagram	7
4.3 Class Diagram	8
5. Technology / Methodology	9
5.1 Technique Followed	9
6. Implementation	10
6.1 Source Code	10
7. Results	15
Conclusion & Future Enhancements	21
Bibliography	22

LIST OF FIGURES

Sl. No.	Figure No.& Name	Page No.
4.1	Flow chart	6
4.2	Data flow diagram	7
4.3	Class diagram	8
7.1	Snapshot of main menu	15
7.2	Snapshot of customer entry page	15
7.3	Snapshot of adding a member	16
7.4	Snapshot of deleting a member	16
7.5	Snapshot of session management	17
7.6	Snapshot of filter by day and time	17
7.7	Snapshot of print database	18
7.8	Snapshot of clear data	18
7.9	Snapshot of car management	19
7.10	Snapshot of adding new car	19

CHAPTER -1

INTRODUCTION

1.1 Introduction

We have developed driving school management system to get rid from manual entry and store it in a file which can be easily available. Here we perform various operations like inserting a record, deleting a record, search a record, updating a record and displaying a record. For this purpose, need the software which could easily and conveniently maintain the student details. The record of student can be stored in a single file. This project “driving school management system” includes some facilities such as id, name, course, phone number, location, etc .

1.2 Aim of the Project

The aim of a driving school management system is to revolutionize the way driving schools operate by providing a comprehensive and integrated solution for managing their administrative tasks. By automating processes, streamlining operations, and centralizing data, the system aims to enhance the efficiency, organization, and effectiveness of the driving school. It aims to optimize resource allocation, improve scheduling accuracy, and reduce manual work, ultimately saving time and effort for driving school administrators and staff. Additionally, the system aims to enhance communication and provide a better experience for students, instructors, and administrators through features like notifications, reminders, and messaging capabilities. With its reporting and analytics functionalities, the system enables data-driven decision-making, allowing administrators to analyze student progress, instructor performance, and financial data. Ultimately, the aim is to create a more streamlined, organized, and customer-centric driving school environment that maximizes operational efficiency and enhances the overall experience for all stakeholders involved.

1.3Main Purpose

The main purpose of a driving school management system is to streamline and automate various administrative tasks involved in running a driving school. This comprehensive software solution is designed to enhance efficiency, organization, and overall effectiveness in managing the operations of a driving school.

One of the primary functions of a driving school management system is to simplify the process of managing student registrations and scheduling driving lessons. It allows the school to maintain a centralized database of students, their contact information, and their progress throughout the course. The system can generate student profiles, track their attendance, and manage their driving lesson schedules, ensuring that each student receives the appropriate training at the right time.

Additionally, the driving school management system facilitates the management of driving instructors. It enables the school to maintain a database of instructors, including their qualifications, availability, and assigned students. The system assists in scheduling instructors for lessons, ensuring efficient utilization of resources and minimizing conflicts in scheduling.

1.4 Goals and Objectives

- Automate administrative tasks, reducing manual work and paperwork.
- Maintain a centralized database for organized data management.
- Improve communication and collaboration between students, instructors, and administrators.
- Optimize resource allocation, ensuring efficient utilization of instructors, vehicles, and classrooms.
- Track student progress and performance, providing personalized guidance and support.
- Simplify financial management processes, including generating invoices and tracking payments.
- Provide insights and analytics through reports, enabling data-driven decision-making.
- Enhance overall efficiency, organization, and performance of the driving school.

By achieving these goals, the driving school management system aims to make administrative processes easier, improve communication and resource management, track student progress, simplify finances, and provide valuable insights for better decision-making.

CHAPTER – 2

GETTING STARTED

2.1 Problem Statement

The problem statement of a driving school management system revolves around the challenges faced in efficiently managing various aspects of a driving school. It includes the need to streamline operations, improve customer and session management, enhance communication, and ensure effective utilization of resources. The specific problem statement could be as follows: Driving schools face difficulties in managing customer information, scheduling and tracking driving sessions, coordinating with instructors, and maintaining a fleet of cars. The existing manual processes are time-consuming, prone to errors, and lack proper organization. Additionally, communication between customers, instructors, and administrators is often fragmented, leading to confusion and inefficiency. The driving school management system aims to address these challenges by providing a centralized platform to automate and streamline administrative tasks, improve session scheduling and tracking, enhance communication, and optimize resource management.

2.2 Proposed system

The following features are included in our project

Customer Management:

- Maintain customer details like names, contact information.
- Register new customers and manage their course enrollments.
- Delete customer record if necessary, with appropriate security measures.

Session Management:

- The students can choose the session which is available
- Delete session records when they are no longer relevant.

Car Management:

- Keep track of available cars for driving lessons.
- Assign cars to sessions and courses based on availability and location.
- Delete car records if required, adhering to data privacy guidelines.

Database Deletion:

- Provide an option to securely delete unnecessary data from the database.

2.3 Advantages

Streamlined Administrative Processes:

- Automation of administrative tasks reduces manual efforts and minimizes errors.
- Simplifies customer registration, enrollment, and record-keeping processes.
- Improves efficiency and saves time for staff and administrators.

Resource Optimization:

- Car inventory management and allocation for driving lessons.
- Efficient utilization of vehicles, minimizing downtime.

Cost and Time Savings:

- Automation reduces paperwork, eliminates redundancies, and saves time.
- Streamlined processes reduce administrative costs.
- Staff can focus on value-added tasks rather than repetitive manual work.

Scalability and Flexibility:

- Ability to scale the system to accommodate growth in customer base or new locations.
- Customization options to adapt the system to specific requirements.

2.4 Project Applications

Operational Optimization:

- The driving school management system optimizes and streamlines administrative processes, allowing driving schools to achieve operational efficiency and effectiveness.

Enhanced Customer Engagement:

- The system fosters enhanced customer engagement by facilitating personalized services and efficient communication channels.

Resource Allocation and Utilization:

- The driving school management system ensures optimal resource allocation and utilization, particularly in terms of instructors and cars.

Progress Tracking and Performance Monitoring:

- With its comprehensive tracking capabilities, the system enables meticulous monitoring of customer progress and driving lesson achievements.

CHAPTER – 3

SYSTEM REQUIREMENTS

3.1 Hardware Requirements

Minimum RAM :- 2GB

Processor :- Intel Pentium 5

Operating System :- Windows 10

3.2 Software Requirements

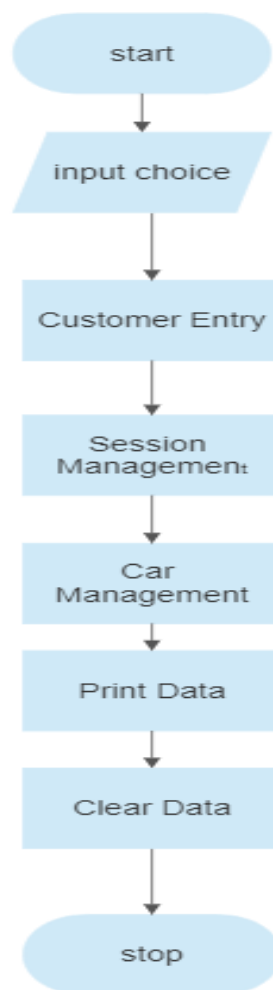
Language :- C++

Software used :- Dev C++

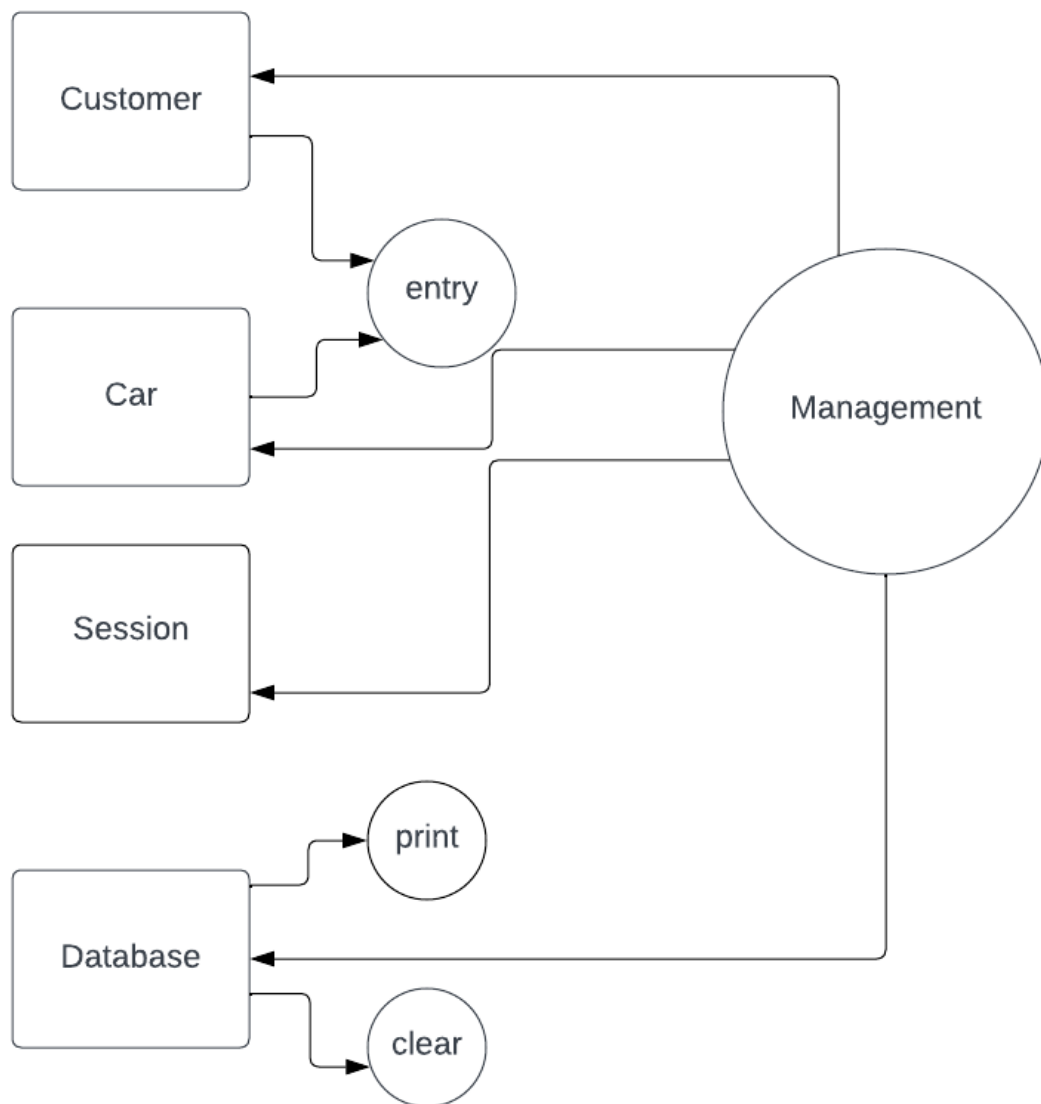
CHAPTER - 4

DESIGN

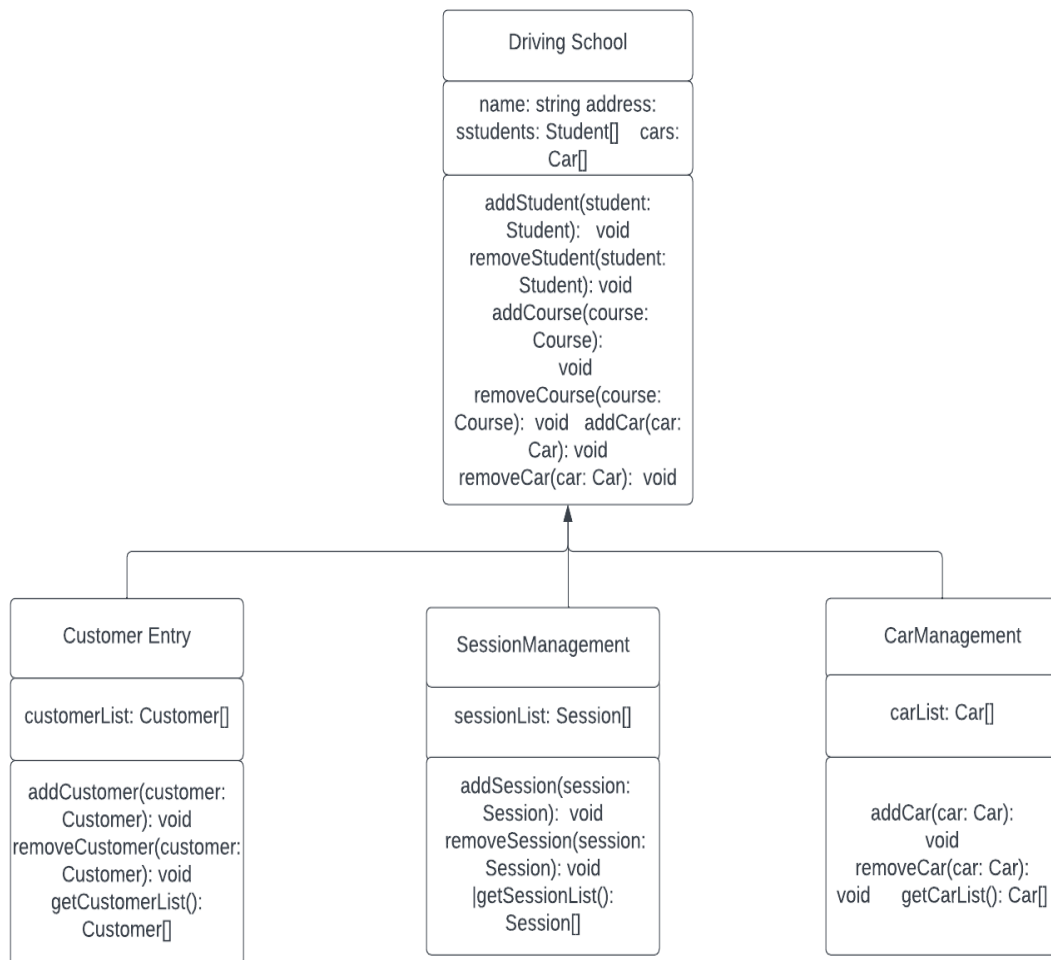
4.1 Flow Chart



4.2 Data Flow Diagram



4.3 Class Diagram



CHAPTER - 5

TECHNOLOGY / METHODOLOGY

5.1 Technique Followed

Insertion: To insert the customer and the car insertion operation is used.

- Open the appropriate file in write mode to add new data.
- Collect the required information from the user or system inputs.
- Append the new data to the file in the desired format.
- Close the file after the insertion is complete.

Deletion: To delete the customer and the car insertion operation is used.

- Open the file in read mode to access the existing data.
- Identify the data to be deleted based on specific criteria or user input.
- Create a new file to store the modified data or temporarily hold the remaining records.
- Read each record from the original file, skipping the records to be deleted.
- Write the remaining records to the new file.
- Replace the original file with the new file, effectively deleting the desired data.
- Close both the original and new files.

Searching: To search the particular record searching is used

- Open the file in read mode to access the data.
- Determine the search criteria based on user input or specific requirements.
- Read each record from the file, comparing it against the search criteria.
- If a match is found, display the relevant information to the user.
- Close the file after the search operation is completed.

Sorting: To search the particular record searching is used.

- Open the file in read mode to access the data.
- Read all the records from the file and store them in memory or an array.
- Apply the desired sorting algorithm (e.g., bubble sort, merge sort, quicksort) to sort the records based on specific criteria (e.g., customer name, session date).
- Write the sorted records back to the file, overwriting the existing data.
- Close the file after the sorting operation is completed.

CHAPTER - 6

IMPLEMENTATION

6.1 Source Code

```
void Addcar()
{
    ofstream writetofile;
    writetofile.open("cars.txt", ios::app);
    car c1;
    cout << "\t\tPlease enter the name of new car " << endl;
    cin >> c1.name;
    cout << "\t\tPlease enter the model of new car " << endl;
    cin >> c1.model;
    cout << "\t\tplease enter the color of new car " << endl;
    cin >> c1.color;
    cout << "\t\tplease enter the type of new car " << endl;
    cin >> c1.type;
    writetofile << endl << c1.name << "\t\t" << c1.model << "\t\t" << c1.color << "\t\t"
<< c1.type << endl;
    writetofile.close();
    cout << "\t\tcar has been added succesfully ";
}

bool checkId(customer arr[], int noofitems, int id)
{
    for (int i = 0; i < noofitems; i++)
    {
        if (id == arr[i].id)
            return 1;
    }
    return 0;
}
```

```
void addSession(string arr[][5], int sizeR) {
    string day, type, time; int counter = 0;
    cout << "\t\t\t-----\n";
    cout << "\t\t\tAdd Session:\n";
    cout << "\t\t\t-----\n";
    cout << "\n\t\t\tEnter your preferences \n\t\t\tDay (sat, sun, mon ...): "; // reading preferences
    from the user
    cin >> day;
    cout << "\t\t\tTime (6, 7, 8, 9): ";
    cin >> time;
    cout << "\t\t\tType (A, M): ";
    cin >> type;
    for (int i = 0; i < sizeR; i++) {        // loop for rows
        bool book = false;                // boolean condition to break the loop when booked
        for (int j = 0; j < 5; j++) {        // loop for columns
            if (arr[i][j] == day) {
                if (arr[i][j + 1] == time) {
                    if (arr[i][j + 2] == type) {
                        if (arr[i][j + 4] == "none") {
                            cout << "\n\t\t\tChosen time is available with " << arr[i][j + 3] << "\n";
                            cout << "\t\t\tDo you want to book this session? if yes type 'yes': ";
                            string opt;
                            cin >> opt;
                            if (opt == "yes") {
                                cout << "\n\t\t\tEnter Your ID: ";
                                string id;
                                cin >> id;
                                arr[i][j + 4] = id;
                                cout << "\t\t\tSession has been booked successfully :) \n";
                                book = true; // will end the loop
                            }
                        }
                    }
                }
            }
            else {
                counter++;
            }
        }
    }
}
```

```
    }
    if (counter == 2) {
        cout << "\t\t\tchosen time is fully booked \n";
        cout << "\t\t\t-----\n";
    }
}
}
}
}
}
if (book)
    break;
}
}

void filterDayType(string arr[][5], int sizeR) {
    string day, type, opt; int free = 0;
    cout << "\t\t\t-----\n";
    cout << "\t\t\tFilter by Day & Type:\n";
    cout << "\t\t\t-----\n";
    cout << "\n\t\t\tEnter your preferences \n\t\t\tDay (sat, sun, mon ...): ";
    cin >> day;
    cout << "\t\t\tType (A, M); ";
    cin >> type;
    for (int i = 0; i < sizeR; i++) {
        for (int j = 0; j < 5; j++) {
            if (arr[i][j] == day) {
                if (arr[i][j + 2] == type) {
                    if (arr[i][j + 4] == "none") {
                        cout << "\t\t\tSession at " << arr[i][j + 1] << " with " << arr[i][j + 3] << " is
available. \n";
                        free++;
                    }
                }
            }
        }
    }
}
```

```
}  
if (free >= 1) {  
    cout << "\t\tDo you want to book? if yes type 'yes': ";  
    cin >> opt;  
    if (opt == "yes") {  
        string time, cap, id;  
        cout << "\t\tEnter time: ";  
        cin >> time;  
        cout << "\t\tEnter captain: ";  
        cin >> cap;  
        cout << "\t\tEnter your ID: ";  
        cin >> id;  
        bookFree(arr, 112, day, type, time, cap, id);  
    }  
}  
else {  
    cout << "\t\tThe choosen preferences are fully booked :( \n";  
    cout << "\t\t-----\n";  
}  
}  
  
void filterTimeType(string arr[][5], int sizeR) {  
    string time, type, opt; int free = 0;  
    cout << "\t\t-----\n";  
    cout << "\t\tFilter by Time & Type:\n";  
    cout << "\t\t-----\n";  
    cout << "\n\t\tEnter your preferences \n\t\tTime (6, 7, 8, 9): ";  
    cin >> time;  
    cout << "\t\tType (A, M); ";  
    cin >> type;  
    for (int i = 0; i < sizeR; i++) {  
        for (int j = 0; j < 5; j++) {  
            if (arr[i][j] == time) {  
                if (arr[i][j + 1] == type) {  
                    if (arr[i][j + 3] == "none") {
```

```
        cout << "\t\t\tSession on " << arr[i][j - 1] << " with " << arr[i][j + 2] << " is
available. \n";
        free++;
    }
}
}
}
}
}
if (free >= 1) {
    cout << "\t\t\tDo you want to book? if yes type 'yes': ";
    cin >> opt;
    if (opt == "yes") {
        string day, cap, id;
        cout << "\t\t\tEnter day (sat, sun, mon ...): ";
        cin >> day;
        cout << "\t\t\tEnter captain: ";
        cin >> cap;
        cout << "\t\t\tEnter your ID: ";
        cin >> id;
        bookFree(arr, 112, day, type, time, cap, id);
    }
}
else {
    cout << "\t\t\tThe choosen preferences are fully booked :( \n";
    cout << "\t\t\t-----\n";
}
}

void clearData(string arr[][5], int sizeR) {
    for (int i = 0; i < sizeR; i++)
        arr[i][4] = "none";
    cout << "\t\t\t-----\n";
    cout << "\t\t\tData has been cleared successfully (all sessions are free) \n";
}
```

CHAPTER -7

RESULTS

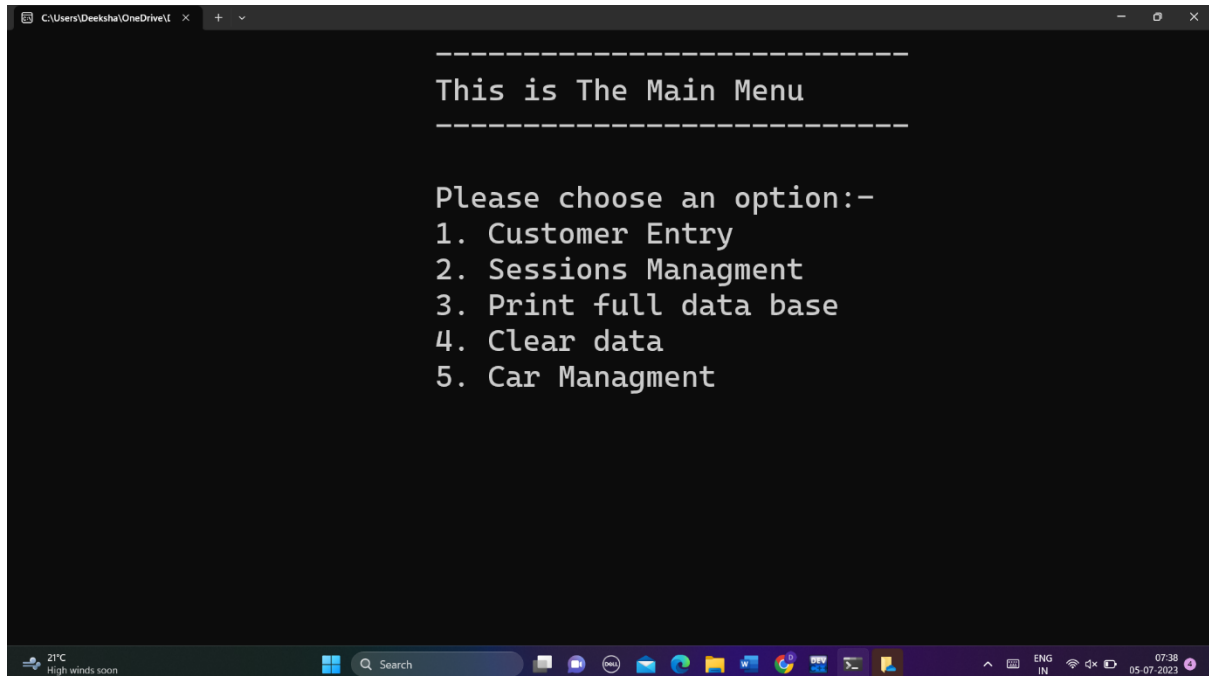


Fig-7.1: Snapshot of main menu

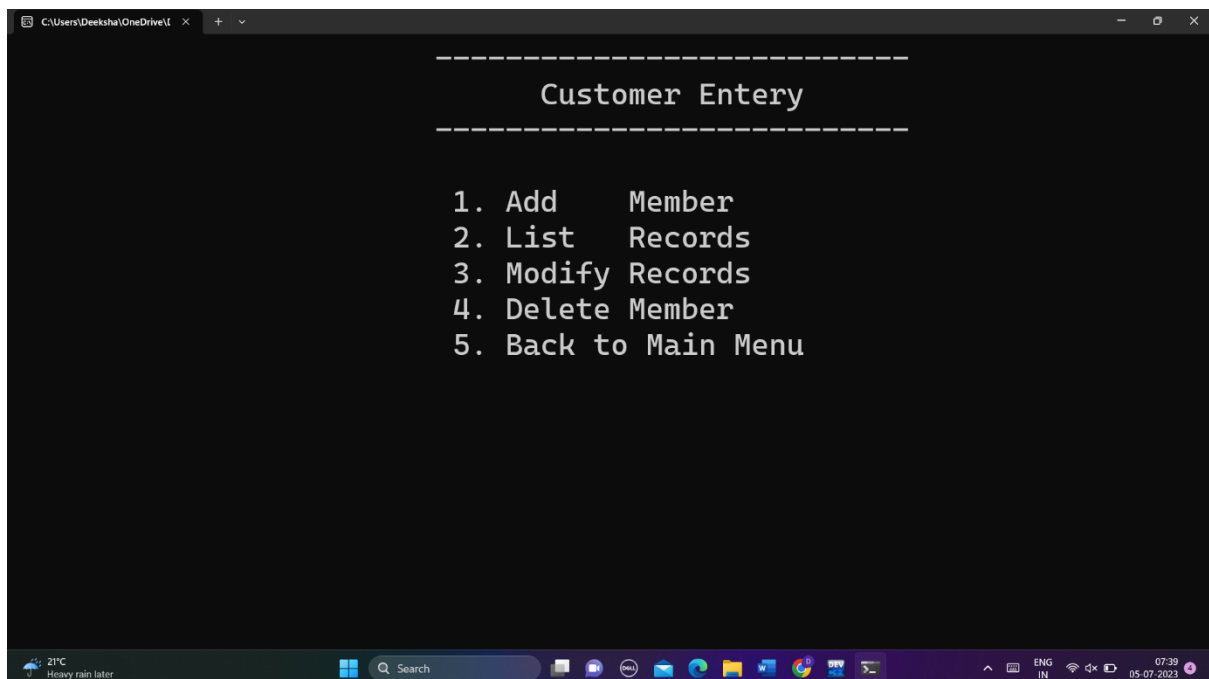


Fig-7.2: Snapshot of customer entry page

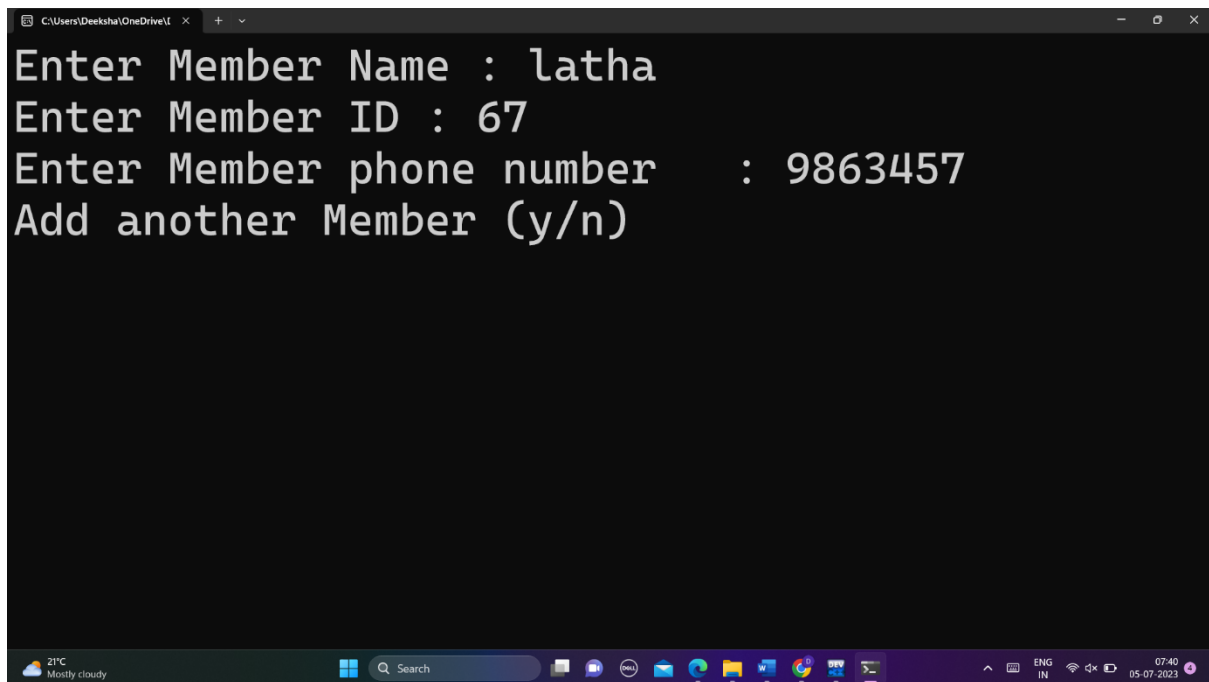


Fig-7.3: Snapshot of adding a member

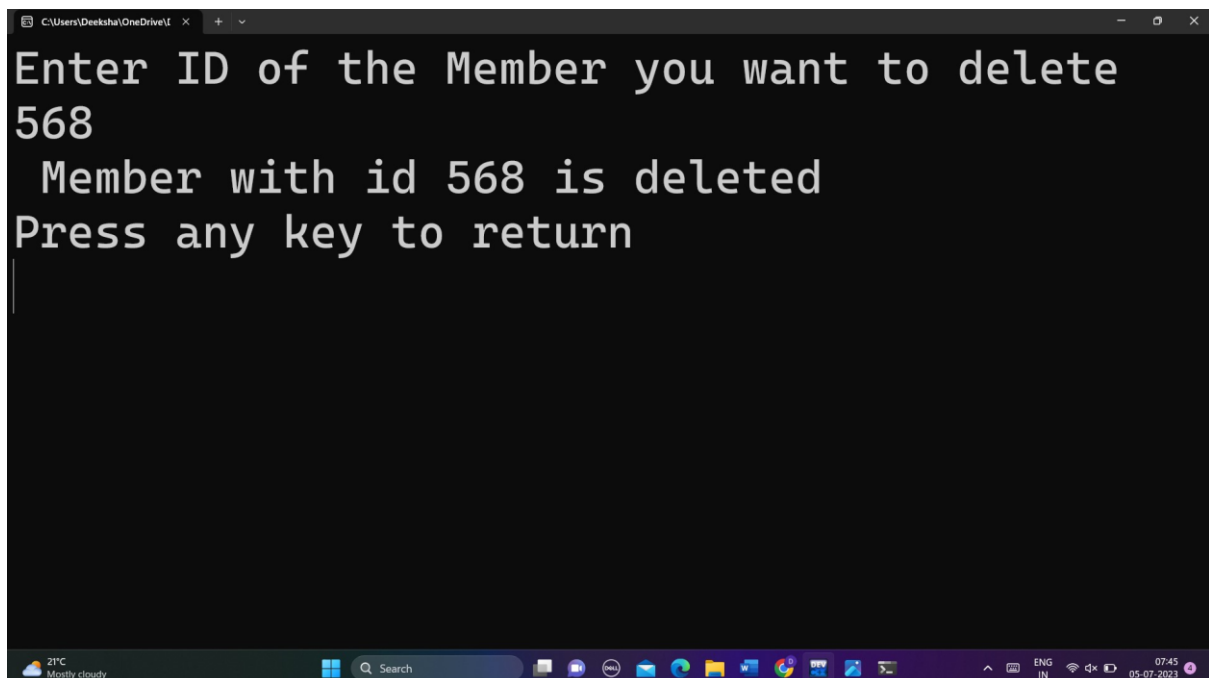


Fig-7.4: Snapshot of deleting a member

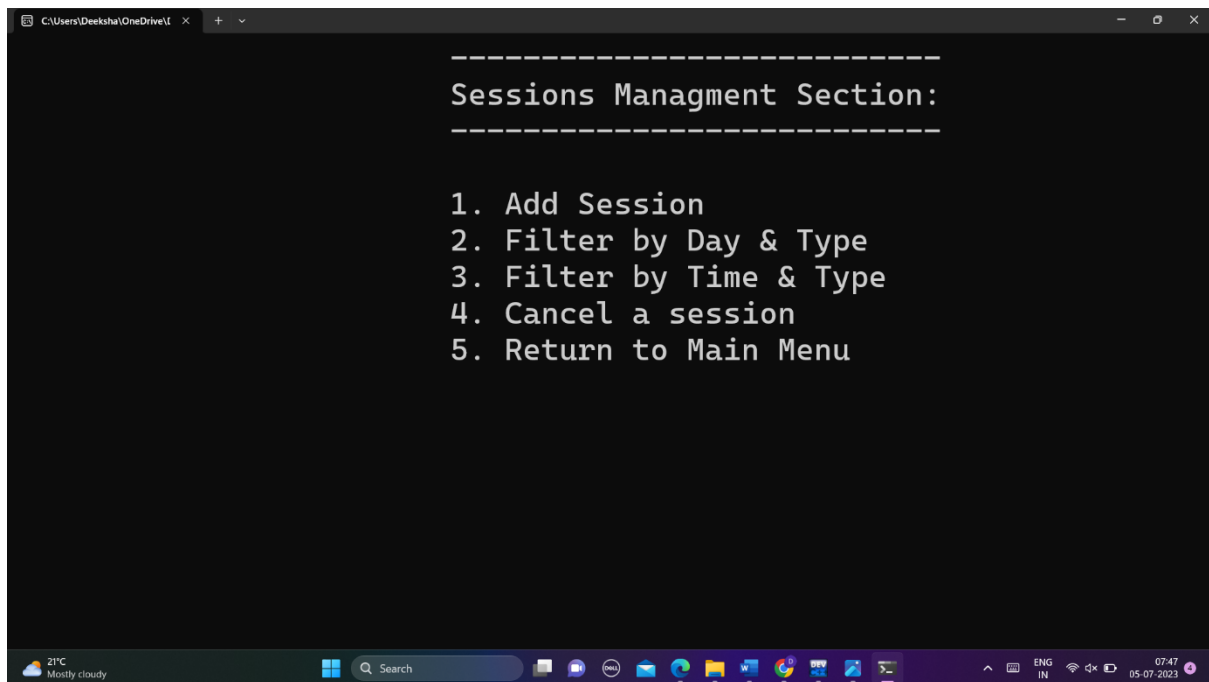


Fig-7.5: Snapshot of session management

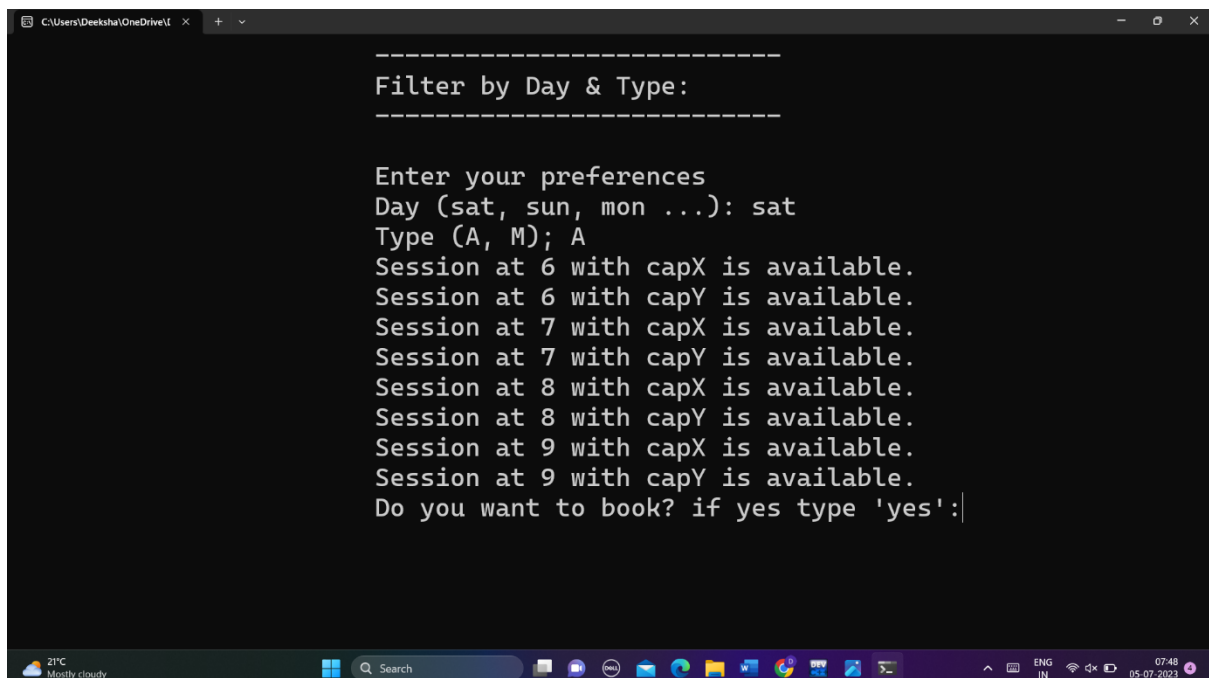
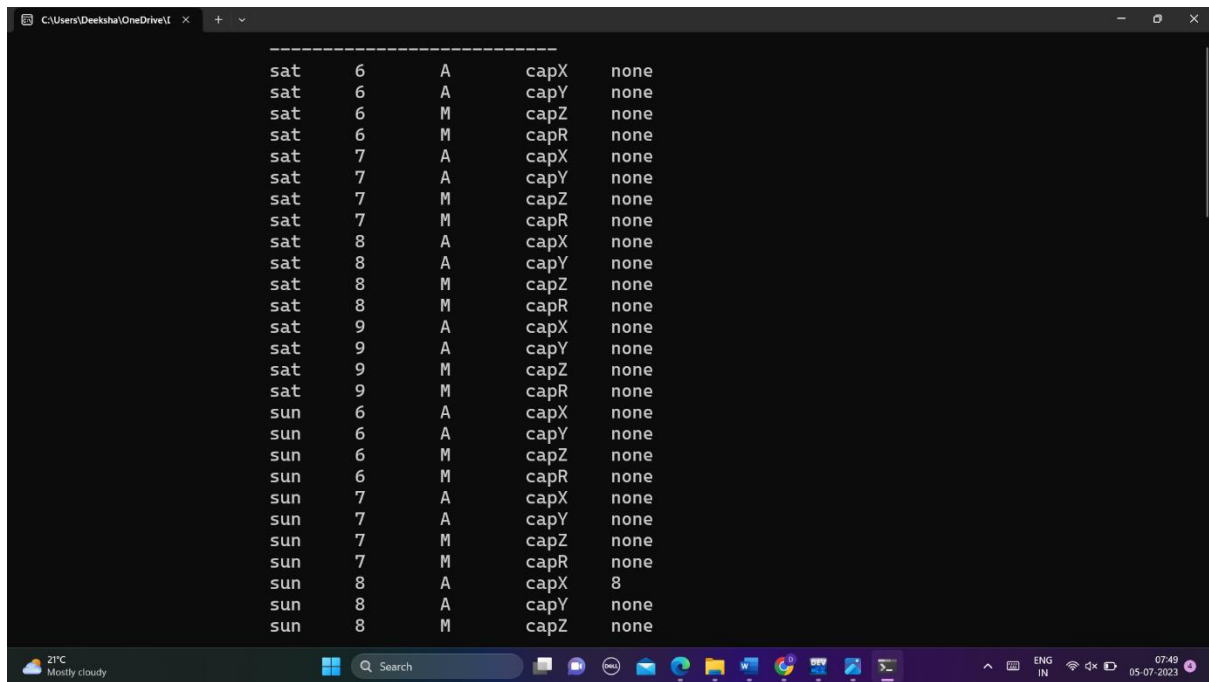
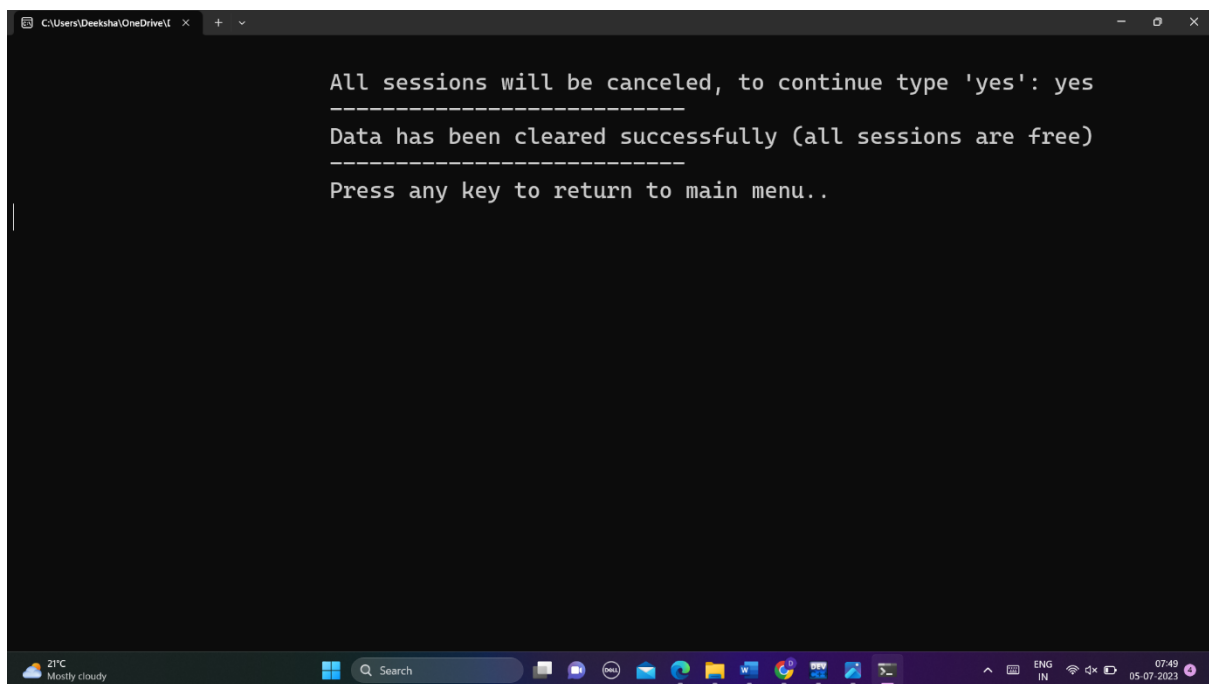


Fig-7.6: Snapshot of filter by day and time



day	number	letter	cap	status
sat	6	A	capX	none
sat	6	A	capY	none
sat	6	M	capZ	none
sat	6	M	capR	none
sat	7	A	capX	none
sat	7	A	capY	none
sat	7	M	capZ	none
sat	7	M	capR	none
sat	8	A	capX	none
sat	8	A	capY	none
sat	8	M	capZ	none
sat	8	M	capR	none
sat	9	A	capX	none
sat	9	A	capY	none
sat	9	M	capZ	none
sat	9	M	capR	none
sun	6	A	capX	none
sun	6	A	capY	none
sun	6	M	capZ	none
sun	6	M	capR	none
sun	7	A	capX	none
sun	7	A	capY	none
sun	7	M	capZ	none
sun	7	M	capR	none
sun	8	A	capX	8
sun	8	A	capY	none
sun	8	M	capZ	none

Fig-7.7: Snapshot of print database



```
All sessions will be canceled, to continue type 'yes': yes
-----
Data has been cleared successfully (all sessions are free)
-----
Press any key to return to main menu..
```

Fig-7.8: Snapshot of clear data

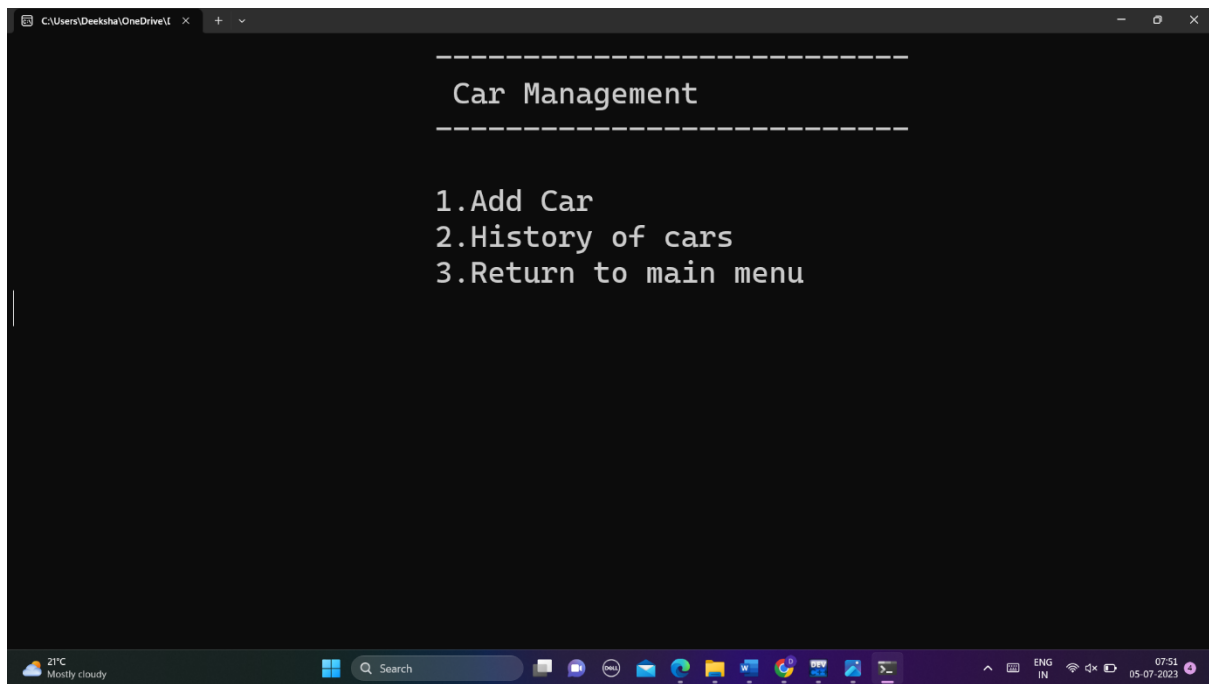


Fig-7.9: Snapshot of car management

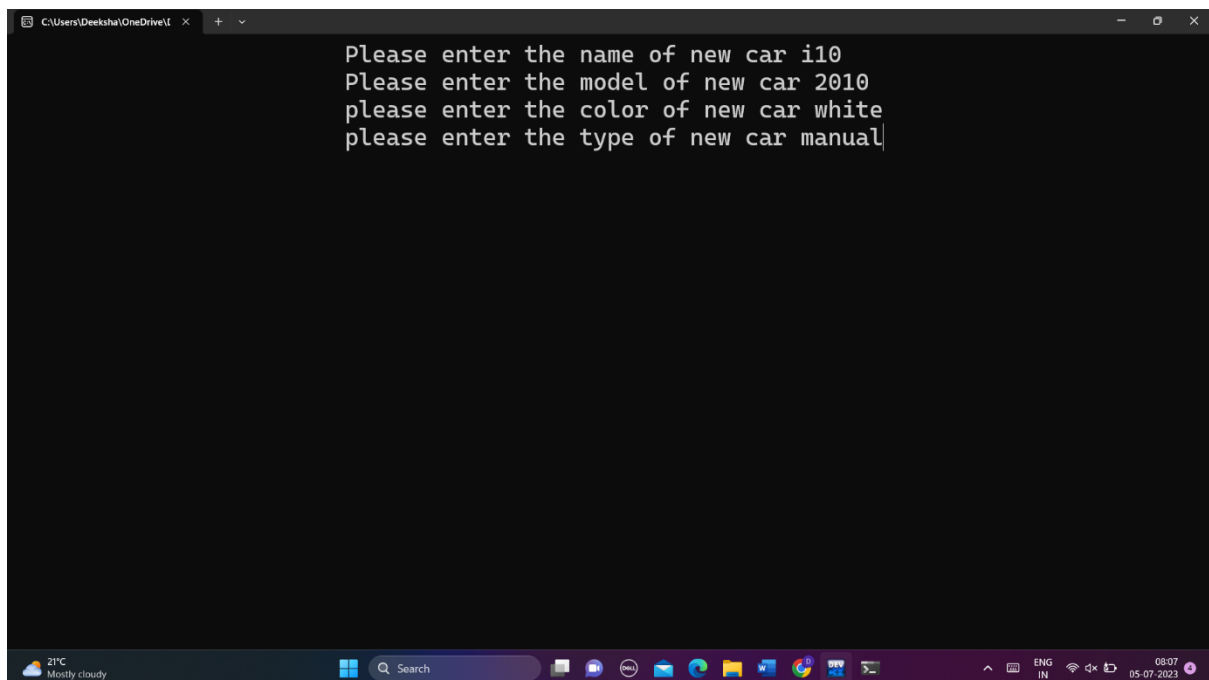


Fig-7.10: Snapshot of adding new car

CONCLUSION & FUTURE ENHANCEMENT

Conclusion

In conclusion, the driving school management system file structure mini project offers a well-structured and organized approach to managing the files and data within a driving school. By implementing a carefully designed file structure, the project ensures efficient management and easy accessibility of various types of data, including customer information, session records, and car details. The file structure provides a systematic organization of directories and subdirectories, simplifying data retrieval, updates, and maintenance. The inclusion of consistent file naming conventions further enhances the searchability and identification of files, saving time and improving productivity. With the implementation of this file structure mini project, driving schools can effectively manage their data, minimize the risk of data loss or misplacement, and lay the foundation for a streamlined and efficient driving school management system.

Future Enhancement

- **Online Booking and Payment:** Integrate an online booking and payment system to allow customers to schedule driving lessons, make payments online, and receive instant confirmation.
- **Mobile Application:** Develop a mobile application for the driving school management system, enabling customers to access their lesson schedules, receive notifications, and track their progress on their mobile devices.
- **Automated Lesson Scheduling:** Implement an intelligent algorithm to automatically generate optimized driving lesson schedules based on instructor availability, customer preferences, and traffic conditions. This ensures efficient resource utilization and minimizes scheduling conflicts.
- **Integration with Vehicle Maintenance Systems:** Connect the driving school management system with vehicle maintenance systems to automate reminders for vehicle servicing, track maintenance history, and ensure the fleet is in optimal condition for lessons.

BIBLIOGRAPHY

Book References

- [1] Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object Oriented Approach with
- [2] C++, 3rd Edition, Pearson Education, 1998.
- [3] K.R. Venugopal, K.G. Srinivas, P.M. Krishnaraj: File Structures Using C++, Tata McGraw-Hill,
- [4] Scot Robert Ladd: C++ Components and Algorithms, BPB Publications, 1993.

Web References

- [1] www.google.com
- [2] www.youtube.com
- [3] www.wikipedia.com
- [4] www.w3schools.com
- [5] <https://www.geeksforgeeks.org/>