**CHAPTER 1**

# INTRODUCTION

## PROBLEM DEFINITION

As a consequence car rental companies end up spending a lot of money acquiring cus- tomers and dealing with customer service issues. To help them we have identified the ar- eas of opportunity that could be exploited by using custom-built chatbots .

## OBJECTIVES

The objectives for this mini-project are as follows:

1. **Faster Process** – To make sure a user gets his desire car as early as possible; the car rental management system will provide a faster response to complete the process.
2. **Minimize paperwork** – As all the system is computerized, there is no need to fill any application form for renting purpose. So, the paperwork will be very less.
3. **Centralized** – All types of data and the activities related to the system will be kept in a single place so that it is easy to monitor the system and provide the customer the best service.

## METHODOLOGY TO BE FOLLOWED

* Use of array of structures to store the name, age and license number of the user from which user will be asked to enter the number of entries they want to make and accordingly they have to enter name, age and license number of people
* Use of while loops to take input and display those inputs taken from the users, do- while loops so that user can enter a choice if they want to enter for more entries or not.
* Use of conditional statements to verify the license number of the user, availability Of cars and display cars with its information

## EXPECTED OUTCOMES

1. The chatbot keeps detailed records of both the cars and the customers, the duration they rent car as well as the type of car they rent..
2. This chatbot system will have the ability to generate and print invoice for each

successful transaction.

## HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

* 1. A Personal Computer/Laptop.
  2. Intel Pentium Processor or later.
  3. RAM 512 MB or more.
  4. Storage of 128GB or more.

Software requirements:

1. Windows 7 32-bit/64-bit or above.
2. C language Compiler.

**CHAPTER 2**

# DATA STRUCTURES

A data structure is not only used for organizing the data. It is also used for processing, retrieving, and storing data. There are different basic and advanced types of data struc- tures that are used in almost every program or software system that has been devel- oped. So we must have good knowledge about data structures.

* **Linear data structure:** Data structure in which data elements are arranged sequentially or linearly, where each element is attached to its previous and next adjacent elements, is called a linear data structure.

*Examples of linear data structures are array, stack, queue, linked list, etc.*

* **Static data structure:** Static data structure has a fixed memory size. It is easier to ac- cess the elements in a static data structure.

*An example of this data structure is an array.*

* **Dynamic data structure:** In dynamic data structure, the size is not fixed. It can be ran- domly updated during the runtime which may be considered efficient concerning the memory (space) complexity of the code.

*Examples of this data structure are queue, stack, etc.*

* **Non-linear data structure:** Data structures where data elements are not placed se- quentially or linearly are called non-linear data structures. In a non-linear data struc- ture, we can’t traverse all the elements in a single run only.

*Examples of non-linear data structures are trees and graphs.*

## 1. STRUCTURES

A structure is a keyword that creates user-defined data types in C/C++. A structure cre- ates a data type that can be used to group items of possibly different types into a single type.

Syntax example of declaration of a structure:

**struct** address

{

**char** name[50]; **char** street[100]; **char** city[50]; **char** state[20]; **int** pin;

};

We can use this data type to store dates of different attributes of different data types. For example, If we want to store data on multiple patients such as patient name, age, and blood group.

Declaration of structure variables:

// A variable declaration with structure declaration.

**struct** Point

{

**int** x, y;

} p1; // The variable p1 is declared with 'Point'

// A variable declaration like basic data types

**struct** Point

{

**int** x, y;

};

**int** main()

{

**struct** Point p1; // The variable p1 is declared like a normal variable

}

A structure variable can either be declared with structure declaration or as a separate declaration like basic types.Structure members are accessed using dot (.) operator.

#include <stdio.h>

**struct** Point {

**int** x, y;

};

**int** main()

{

**struct** Point p1 = { 0, 1 };

// Accessing members of point p1 p1.x = 20;

**printf**("x = %d, y = %d", p1.x, p1.y);

**return** 0;

}

**2.1.1 ARRAY OF STRUCTURES**

A structure is a data type in **C/C++** that allows a group of related variables to be treated as a single unit instead of separate entities. A structure may contain elements of different data types – int, char, float, double, etc.

It may also contain an array as its member. Such an array is called an array within a struc- ture. An array within a structure is a member of the structure and can be accessed just as we access other elements of the structure.

# ARRAYS

* + - An array is used to store a collection of variables of the same type. A specific element in an array is accessed by an index.
    - The array elements are stored in contiguous memory locations. The first element

is stored at the lowest address and the last element is stored at the highest ad- dress

An array is a finite collection of similar elements stored in adjacent memory locations. The number of elements to be used are mentioned as index usually ranging from o to N-1 where ‘0’ is lower bound and N-1 is upper bound.

Arrays can be 1-D, 2-D or multi-dimensional. Syntax- Data-type array name[index] Eg: int A[10], char B[20] etc. Representation: if A[5]={10,20,30,40,50} then in the array it is shown as A= A[0] A[1] A[2] A[3] A[4]

**CHAPTER 3**

# DESIGN

## 1. DESIGN GOALS

1. User-Friendliness: The chatbot should be easy to use and understand for the end-user. This can be achieved by using simple language, providing clear and concise responses, and utilizing an intuitive conversational ﬂow.
2. Efficiency: The chatbot should be able to quickly and accurately assist users in finding the information they need, such as available rental properties, rental rates, and availability. This can be achieved through the use of natural language processing and machine learning techniques.
3. Personalization: The chatbot should be able to tailor its responses to the individual user's needs and preferences. This can be achieved by using data such as the user's search history and location to provide more relevant recommendations.
4. Accessibility: The chatbot should be designed to be accessible to users with diﬀerent levels of technical proficiency and diﬀerent abilities. This can be achieved by providing diﬀerent options for interacting with the chatbot, such as voice commands or text-based input.
5. Scalability: The chatbot should be able to handle a large volume of user requests and be able to adapt to changes in the rental market.
6. Integration: The chatbot should be able to integrate with other tools and systems used by the rental service, such as property management software, payment systems, and scheduling tools.

## 2. ALGORITHM / PSEUDOCODE

The system uses two structs, "car" and "passDet", to store information about the cars and the passengers respectively. The system also includes a function called "initCars" that initializes the car data, and a function called "bookCar" that is used to book a car for a certain number of days.

The algorithm for the car rental system using chatbot is as follows:

* Initialize the car data by calling the "initCars" function
* Take input from the user for the car number and number of days they want to book the car for
* Use a for loop to iterate through the "cars" array and find the car with the matching car number
* Check if the car is available for booking (carAvailability == 1)
* If the car is available, set the carAvailability to 0, print a confirmation message with the booking details, and return 1
* If the car is not available, print a message saying the car is not available, and return 0

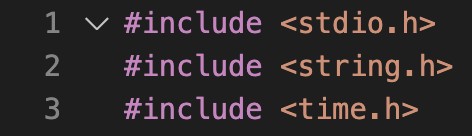
The above is the basic ﬂow of the algorithm, there might be more features that can be added such as storing the booking details and the customer details in a database, the ability to cancel a booking, the ability to search for cars based on diﬀerent criteria such as car type, car color, car model etc.

CHATBOT FOR CAR RENTAL

**CHAPTER 4**

# IMPLEMENTATION

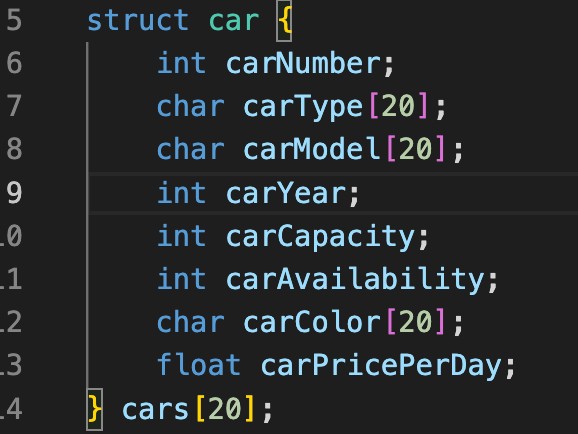
## MODULE 1 FUNCTIONALITY (LIBRARIES)



These are three C standard library headers that are included in this module.

* <stdio.h> is the header for the standard input/output library. It provides functions for reading and writing data, such as printf() and scanf().
* <string.h> is the header for the string manipulation library. It provides functions for working with strings, such as strcpy() and strlen().
* <time.h> is the header for the time manipulation library. It provides functions for working with time, such as time() and localtime().

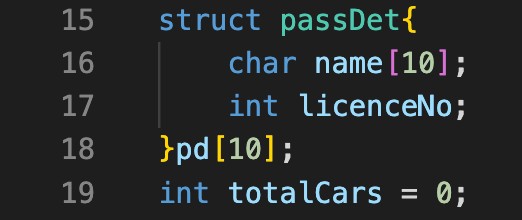
## MODULE 2 FUNCTIONALITY(DECLARING STRUCTURES)



This is a C struct that defines a type called "car". It is an array of 20 "car" structs, each with the following fields:

* carNumber: an integer value representing the unique number of the car
* carType: a character array of length 20, representing the type of the car (e.g. "Sedan", "SUV", "Truck")
* carModel: a character array of length 20, representing the model of the car (e.g. "Camry", "F-150", "X5")
* carYear: an integer value representing the year the car was manufactured
* carCapacity: an integer value representing the number of people the car can seat
* carAvailability: an integer value representing whether the car is available (1) or not (0)
* carColor: a character array of length 20, representing the color of the car
* carPricePerDay: a ﬂoat value representing the price of renting the car per day

This struct can be used to store information about multiple cars, with each element in the array of "car" structs representing a single car.



This is a C struct that defines a type called "passDet" and an array of 10 "passDet" structs, each with the following fields:

* name: a character array of length 10, representing the name of the passenger.
* licenceNo: an integer value representing the licence number of the passenger.

This struct can be used to store information about multiple passengers, with each element in the array of "passDet" structs representing a single passenger.

Additionally, there's a global variable totalCars initialized with 0, which could be used to keep track of total number of cars.

## MODULE 3 FUNCTIONALITY(VOID FUNCTION)



This is a C function called "initCars()" that initializes the data for an array of

"car" structs defined earlier.

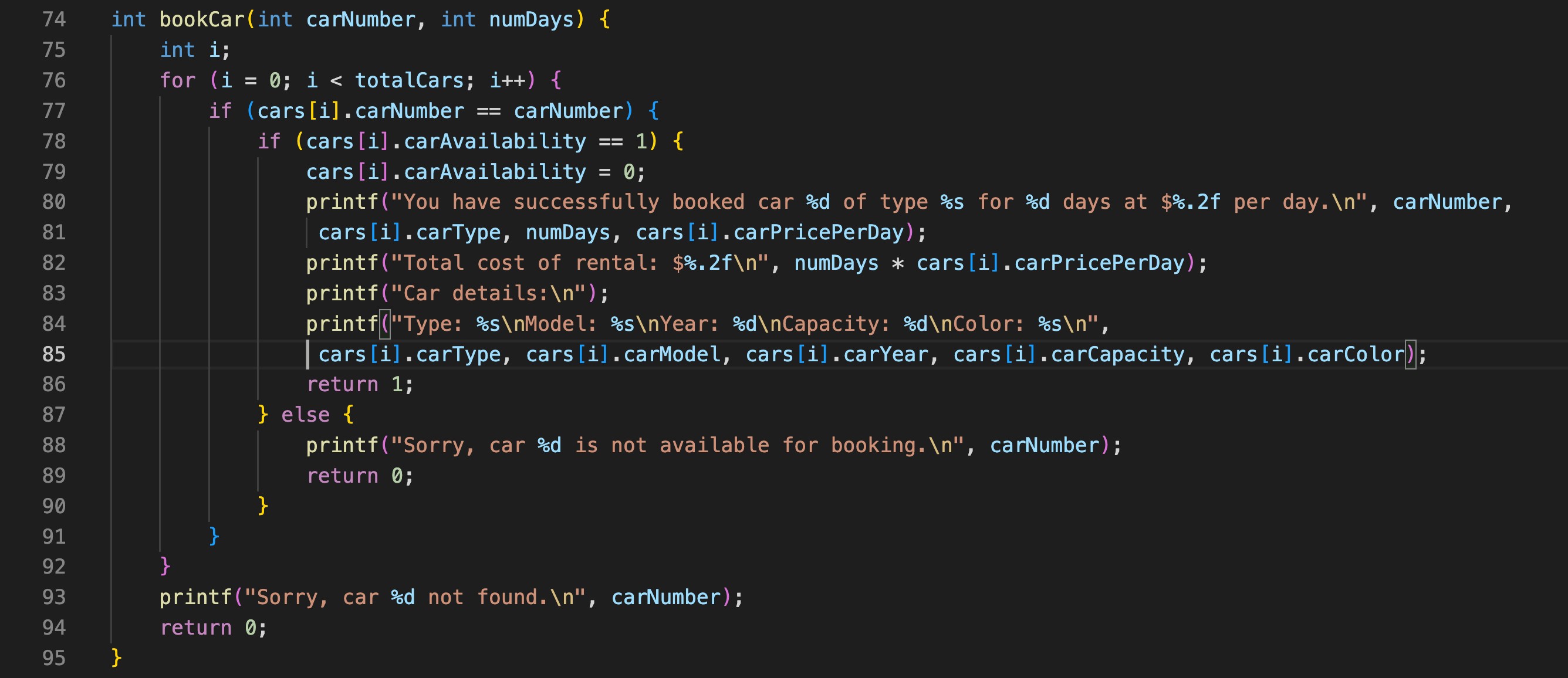
It uses the C standard library function strcpy() to copy string values (e.g. carType, carModel, carColor) into the corresponding fields of the structs.

It also assigns values to the other fields of the struct (e.g. carNumber, carYear, carCapacity, carAvailability, carPricePerDay).

It also increments the global variable totalCars after each car is added.

This function can be called to initialize the data for a set of cars, which can then be used in other parts of the program.

## MODULE 4 FUNCTIONALITY(DRIVER CODE)



This code defines a function called "bookCar" that takes in two parameters:

"carNumber" and "numDays". The function then uses a for loop to iterate through each element of the "cars" array, checking if the carNumber of the current element matches the "carNumber" parameter passed into the function. If a match is found, the code checks if the car is available (if carAvailability is equal to 1). If the car is available, the code sets carAvailability to 0, indicating that the car has been booked. It then prints a message to the console to confirm the booking, including the total cost of the rental, and displays the car details. If the car is not available, the code prints a message to the console that the car is not available for booking. If no match is found, it prints a message that the car is not found. Finally, the function returns 0 or 1, indicating whether the booking was successful or not.

## MODULE 4 FUNCTIONALITY(VOID FUNCTION)

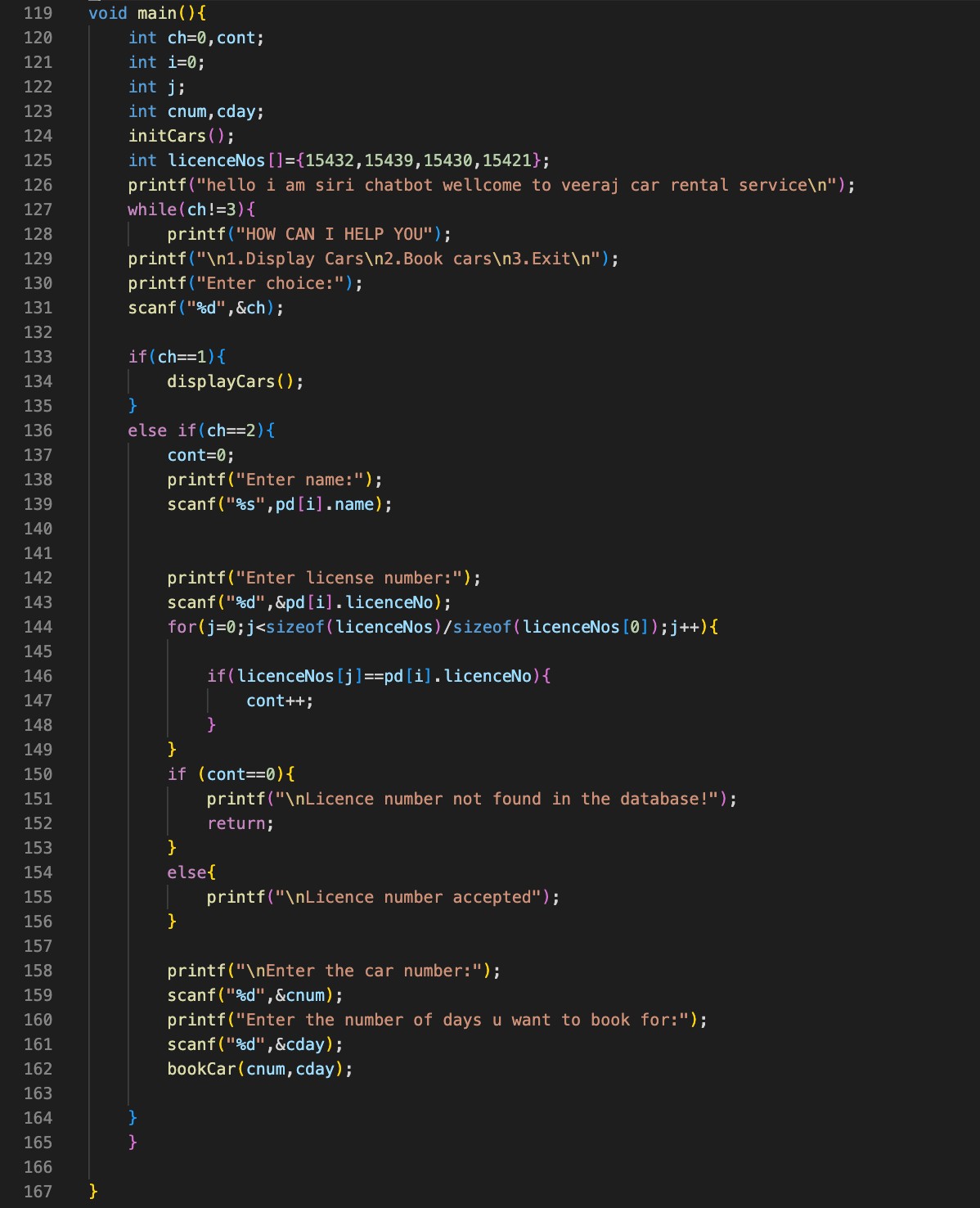


**T**his code defines a function called "displayCars" which iterates through each

element of the "cars" array, and for each iteration it prints out the car's details to the console. The details include Car Number, Type, Model, Year, Capacity, Color, Price Per Day, and Availability. It uses a for loop to iterate over each element of the "cars" array and uses printf function to print the details of the current element of the array. If cars[i].carAvailability is 1, it prints Availability as Yes, otherwise as No.

It also uses the "\n" characters to print the output in a neat format with new lines separating the details of each car. This function likely serves as a way to display all the cars and their details in a readable format.

## MODULE 5 FUNCTIONALITY(VOID FUNCTION)



This code defines the main function of the program. It starts by initializing an array of cars using the "initCars" function and also declare an array of integers called licenceNos. It then prints a welcome message to the user. It then enters a while loop that continues until the user chooses to exit by entering 3. Inside the loop, it presents the user with a menu of options (1. Display Cars, 2. Book Cars, 3. Exit) and prompts the user to enter their choice. If the user chooses option 1, the program calls the "displayCars" function to display all the cars and their details. If the user chooses option 2, the program prompts the user to enter their name and license number, and it also checks whether the license number entered is present in the licenceNos array or not. If it's not present, it prints an error message and terminates the program otherwise it continues by prompting the user to enter the car number and number of days for which the user wants to book the car. It then calls the bookCar function passing the car number and the number of days as arguments.

Finally, it exits the program when the user enters 3.

**CHAPTER 5**

# RESULTS

## chatbot interacting with user:

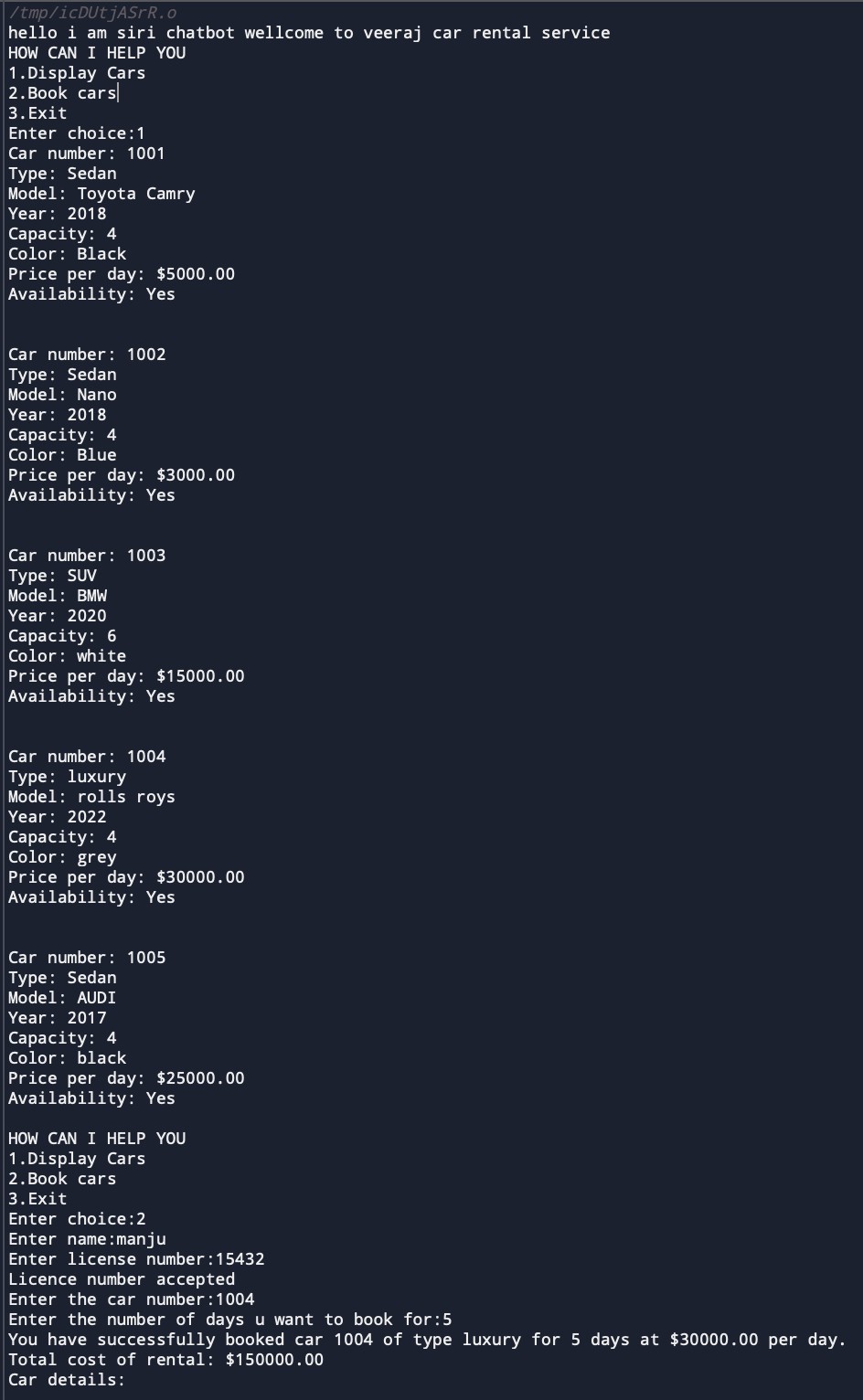
This module appears to be a chatbot designed for a car rental service. The chatbot oﬀers three options for the user: to display cars, to book a car, or to exit the program. The user can enter a choice (1, 2, or 3) to proceed.

In this scenario, the user has chosen to book a car (choice 2). The chatbot then prompts the user to enter their name, license number, car number, and number of days they would like to rent the car for. After the user enters this information, the chatbot confirms that the license number has been accepted, and then displays the total cost of the rental and the details of the car that the user has booked.

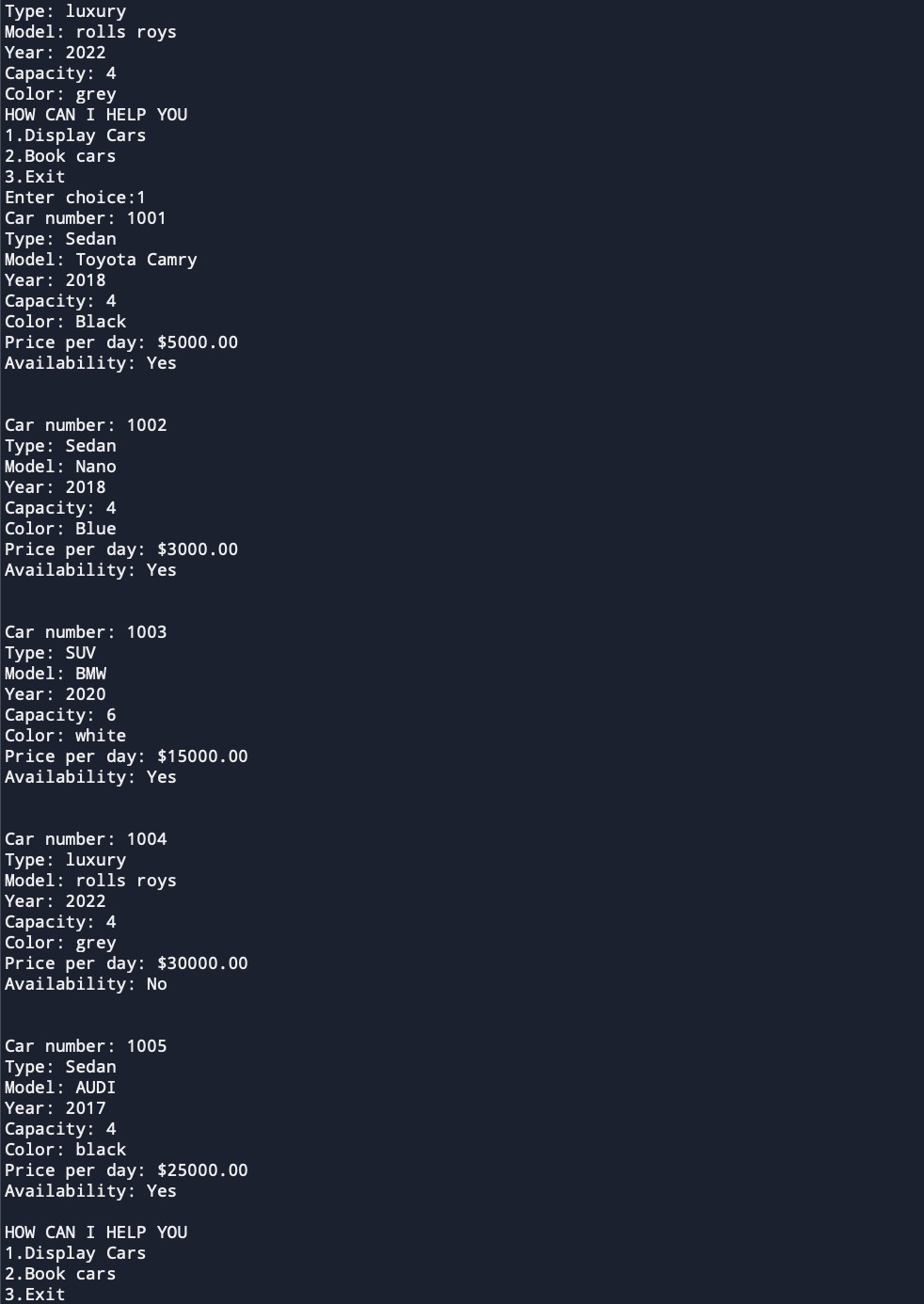
After the user has booked a car, the chatbot then prompts the user to choose between displaying cars, booking cars or exiting the program. This time, the user has chosen to display cars. The chatbot displays the list of cars available for rental, including the car number, type, model, year, capacity, color, price per day, and availability. However, the chatbot now shows that the car number 1004, which was booked by the user, is not available.

The chatbot continues to prompt the user with the options to display cars, book cars or exit the process

## Snapshots of chatbot interacting with user:



CHATBOT FOR CAR RENTAL



CHATBOT FOR CAR RENTAL

**CHAPTER 6**

# CONCLUSION

1. The chatbot designed for a car rental service is a useful tool for customers to easily browse and book cars for rental.
2. The chatbot allows customers to view the available cars, their specifications and pricing, and book a car by providing necessary details like name, license number, car number, and rental period.
3. The chatbot can also indicate the availability of cars which can help customers to select the car of their choice.
4. The chatbot also helps the customer to have a seamless experience by providing options to either display the cars, book the cars or exit the program.
5. The chatbot can also help the car rental service to keep track of the rented cars and their availability, resulting in better inventory management.
6. The chatbot can help to improve the customer experience and increase efficiency for the car rental service.

# REFERENCES

* + Dialogﬂow (previously known as API.AI): A conversational platform that makes it easy to design and integrate a chatbot into your application. It provides a web-based interface for building, training, and deploying chatbots. <https://dialogflow.com/>
  + Botkit: An open-source framework for building chatbots using Node.js. It provides an easy-to-use interface for creating chatbots and integrating them with popular messaging platforms like Slack and Facebook Messenger. <https://botkit.ai/>
  + Microsoft Bot Framework: A comprehensive framework for building chatbots and conversational AI applications. It supports multiple channels, including Skype, Facebook Messenger, and Slack. [https://](https://dev.botframework.com/) [dev.botframework.com/](https://dev.botframework.com/)
  + IBM Watson Assistant: A cloud-based platform for building, training, and deploying chatbots. It provides a variety of features, including natural language understanding and pre-built integrations with popular messaging platforms. <https://www.ibm.com/cloud/watson-assistant>
  + Rasa: Open source conversational AI framework that helps you build contextual assistants. [https://](https://rasa.com/) [rasa.com/](https://rasa.com/)