**CAR RENTAL MANAGEMENT SYSTEM**

**MINI PROJECT REPORT**

**Submitted by**

**ABHIKHYA DEEKSHA BAKKAMANTHALA RA2211003011486**

**KALPESH BONDE RA2211003011502**

**GITANSH PISE RA2211003011504**

**Under the Guidance of**

**Dr. S SHANMUGAM**

**Assistant Professor, Department of Computing Technologies**

***In partial satisfaction of the requirements for the degree of***

## BACHELOR OF TECHNOLOGY

**In**

## COMPUTER SCIENCE ENGINEERING

## Logo, company name Description automatically generated

**SCHOOL OF COMPUTING**

# COLLEGE OF ENGINEERING AND TECHNOLOGY

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR – 603203**

**MAY 2023**



|  |  |  |
| --- | --- | --- |
| S.No | CONTENTS | PAGE NO |
| 1. | Problem Statement | 4 |
| 2. | Modules of Project | 4 |
| 3. | Diagrams | 6 |
|  | 1. Use case Diagram | 6 |
|  | 1. Class Diagram | 7 |
|  | 1. Sequence Diagram | 8 |
|  | 1. Collaboration Diagram | 9 |
|  | 1. State Chart Diagram | 10 |
|  | 1. Activity Diagram | 11 |
|  | 1. Package Diagram | 12 |
|  | 1. Component Diagram | 13 |
|  | 1. Deployment Diagram | 14 |
| 4. | Code/Output Screenshots | 15 |
| 5. | Conclusion and Results | 24 |
| 6. | References | 24 |

1. **Problem Statement**

Car rental companies often face difficulties managing their fleet of vehicles, bookings, and customer data. These challenges can lead to inefficiencies, delayed responses to customer inquiries, and a poor customer experience. The goal of this project is to develop a car rental management system to help car rental companies overcome these challenges and improve their overall operations.

1. **Modules of Project :-**

The Car Rental Management System is developed using C++ programming language. The system consists of the following modules:

**a. Customer Class**

This class is responsible for storing and managing customer data. It has the following attributes:

* Customer name
* Car model
* Car number
* Rental days

**b. Rent Class**

This class is inherited from the Customer class and is responsible for calculating the rental fee for the customer. It has the following methods:

* Data(): This method collects data from the user, such as the customer's name, car model, car number, and rental days.
* Calculate(): This method calculates the rental fee based on the car model and rental days.
* ShowRent(): This method displays the rental fee and customer data in an invoice format.

**<iostream>:**

iostream stands for standard input-output stream. This header file contains

definitions of objects like cin, cout, cerr, etc.You use operators or iostream

member functions to insert data into a stream (output) or extract data from

a stream (input), and to control the format of data that you insert or

extract.

**<fstream>:**

<fstream> library provides functions for files, and we should simply add

#include<fstream> directives at the start of our program. To open a file, a

file stream object should first be created. This is either an of stream object

for writing, or an if stream object for reading.

**<cctype>**

In C++, the cctype module is a standard library header that provides

functions for character handling and classification. It contains functions

that allow the programmer to perform various operations on characters,

including determining whether a character is a letter, digit, or punctuation

mark, and converting characters to uppercase or lowercase.

**<iomanip>**

In C++, the iomanip library is a standard library header that provides

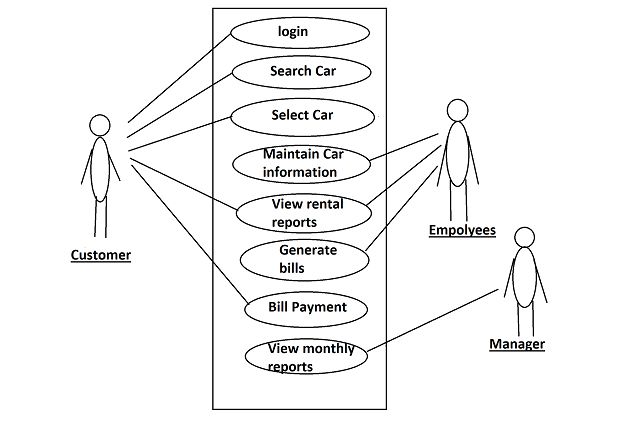
functions to manipulate input/output operations. It contains functions that

allow the programmer to set various properties for output formatting, such

as the width and precision of numbers, the alignment of output.

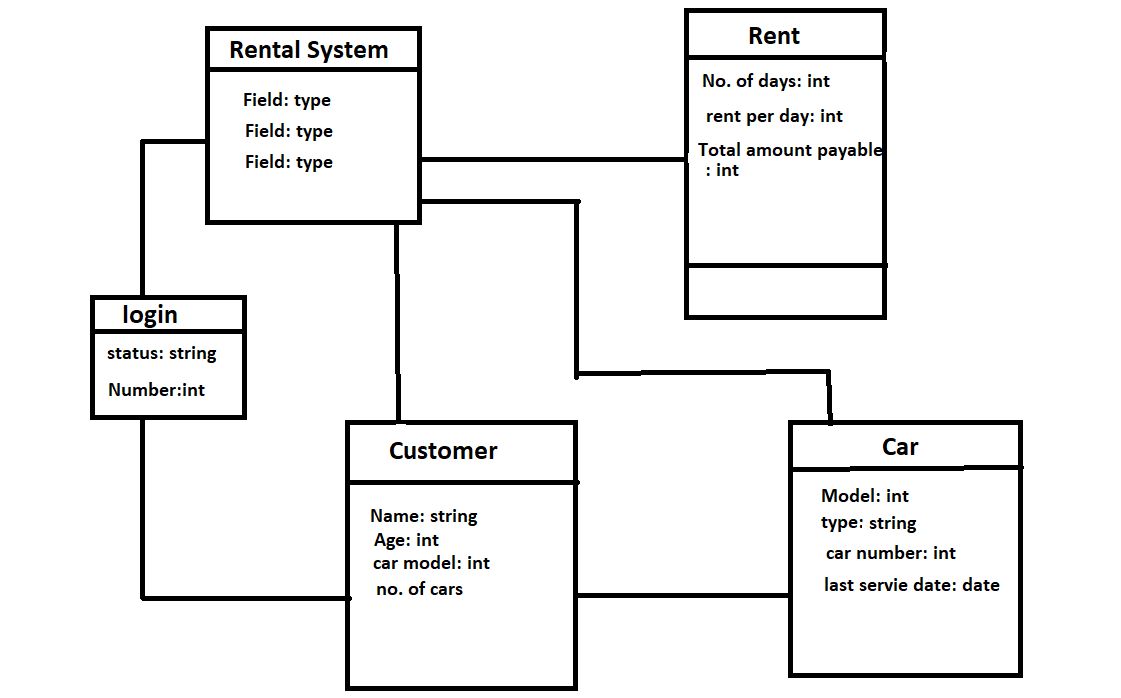
1. **Diagrams :-**

1. **Use case Diagram :-**



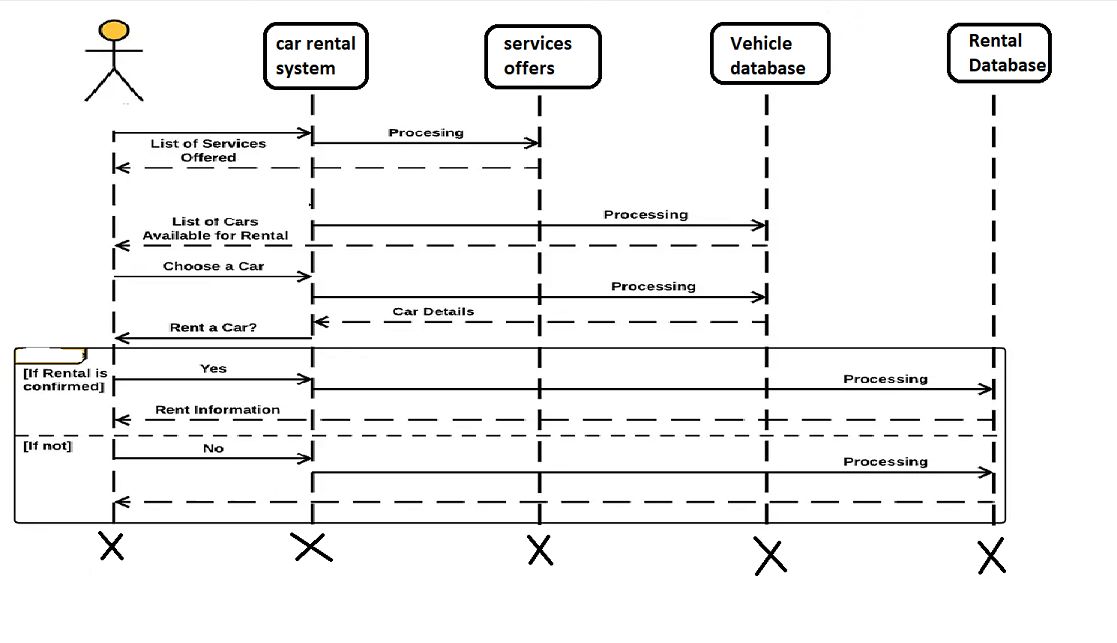
In this diagram, the User interacts with the System by performing different use cases such as Login, Rent a car, Calculate rental fee, and Show customer invoice. The Login use case allows the user to login to the system, and the Rent a car use case allows the user to select a car and provide rental details such as the number of days to rent the car. The Calculate rental fee use case calculates the rental fee based on the selected car and the number of days. Finally, the Show customer invoice use case shows the customer invoice with all the rental details.

1. **Class Diagram :-**



In the diagram, the customer class is shown at the top with its member variables. The rent class is derived from the customer class and is shown at the bottom. It has its own member variables and member functions, which are denoted by the plus sign (+) in front of their names. The arrows between the two classes represent inheritance, indicating that the rent class is derived from the customer class.

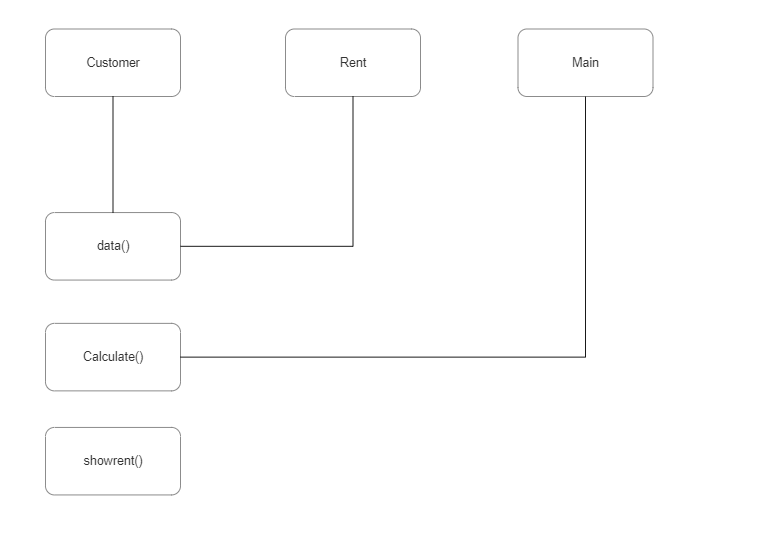
1. **Sequence Diagram :-**



In the above sequence diagram, we have two objects - rent and customer. The rent object is an instance of the rent class, which is inherited from the customer class. The customer object is an instance of the customer class.

The rent object calls the data() method, which takes input from the user and stores it in the object's member variables. Then, the rent object calls the calculate() method, which calculates the rental fee based on the car model and the number of days. Finally, the rent object calls the showrent() method, which displays the invoice to the user.

1. **Collaboration** **Diagram**



The program consists of three classes: customer, rent, and main.

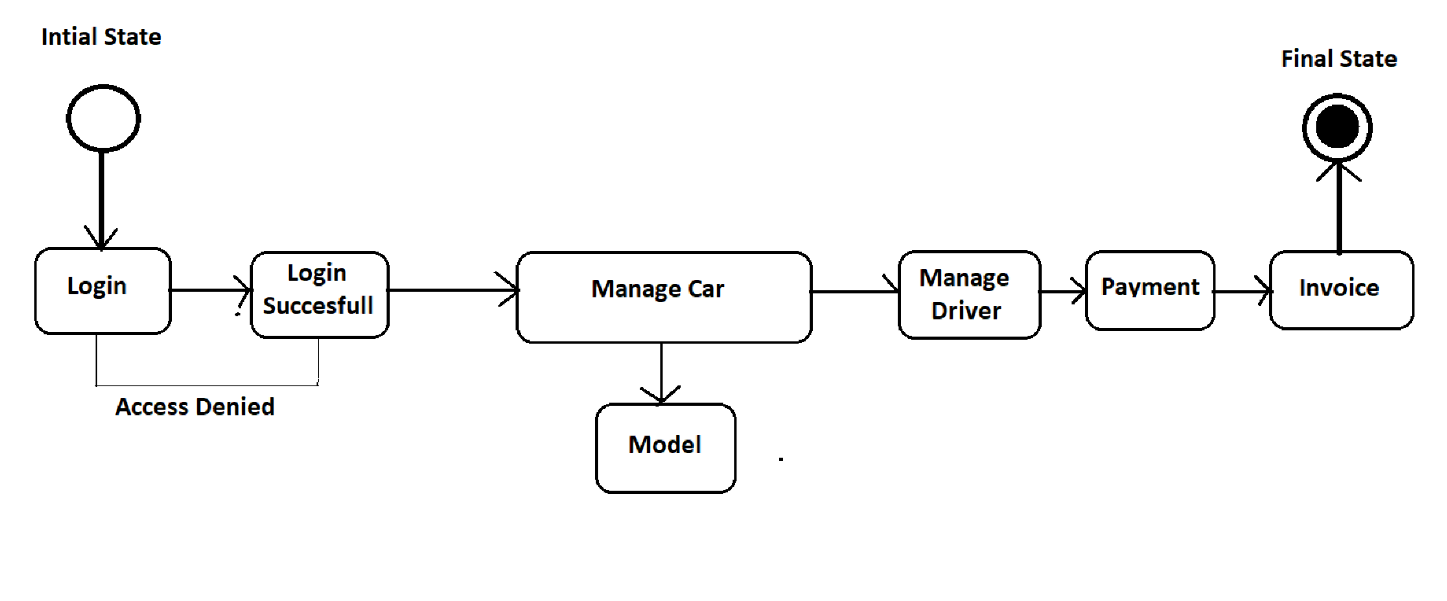
main is the class containing the main() function, which serves as the entry point to the program.

The customer class has a single data member, which is a string called customername.

The rent class inherits from the customer class and has additional data members such as days, rentalfee, and carmodel. It also has three member functions: data(), calculate(), and showrent().

* The **data**() function takes input from the user, including the customer name, car model, and number of days the car will be rented.
* The **calculate()** function calculates the rental fee based on the car model and number of days.
* The **showrent**() fuction prints total rent.

1. **State Chart Diagram :-**



A state diagram, also known as a state machine diagram, is used to model the behavior of an object or system in terms of the states it can be in and the transitions between those states.

In the given code, there are no clear states or transitions defined, but we can assume some states and transitions to explain the concept of a state diagram.

Assuming the state diagram for a car rental system, the possible states and transitions can be as follows:

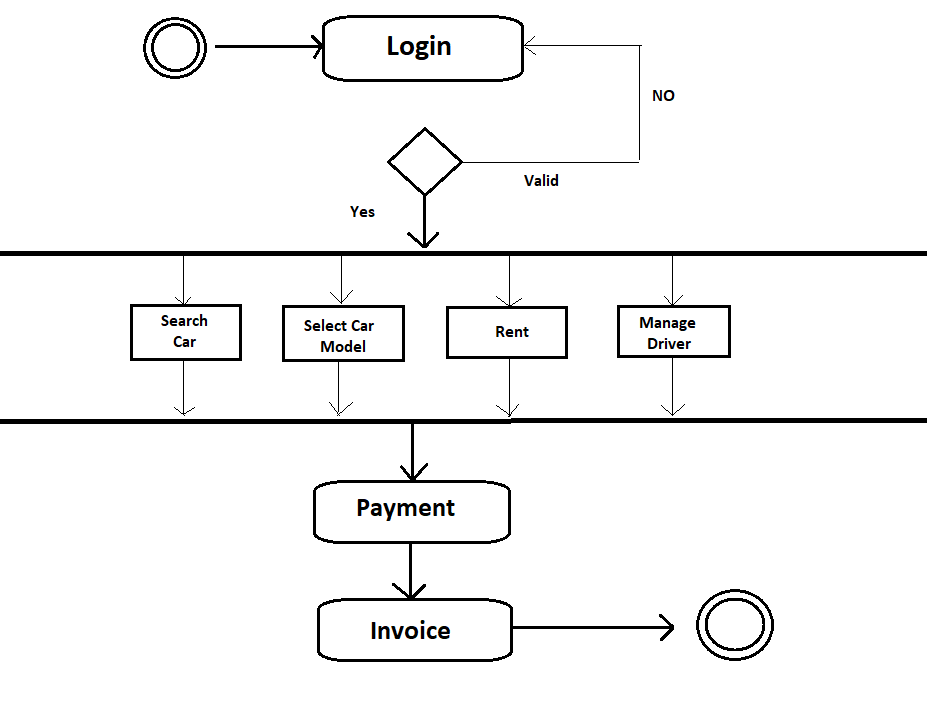
States:

* Initial state: the starting state of the system
* Login state: the state where the user is prompted to login
* Car selection state: the state where the user selects the desired car
* Rental information state: the state where the user provides rental information
* Calculation state: the state where the rental fee is calculated
* Invoice state: the final state where the invoice is generated

Transitions:

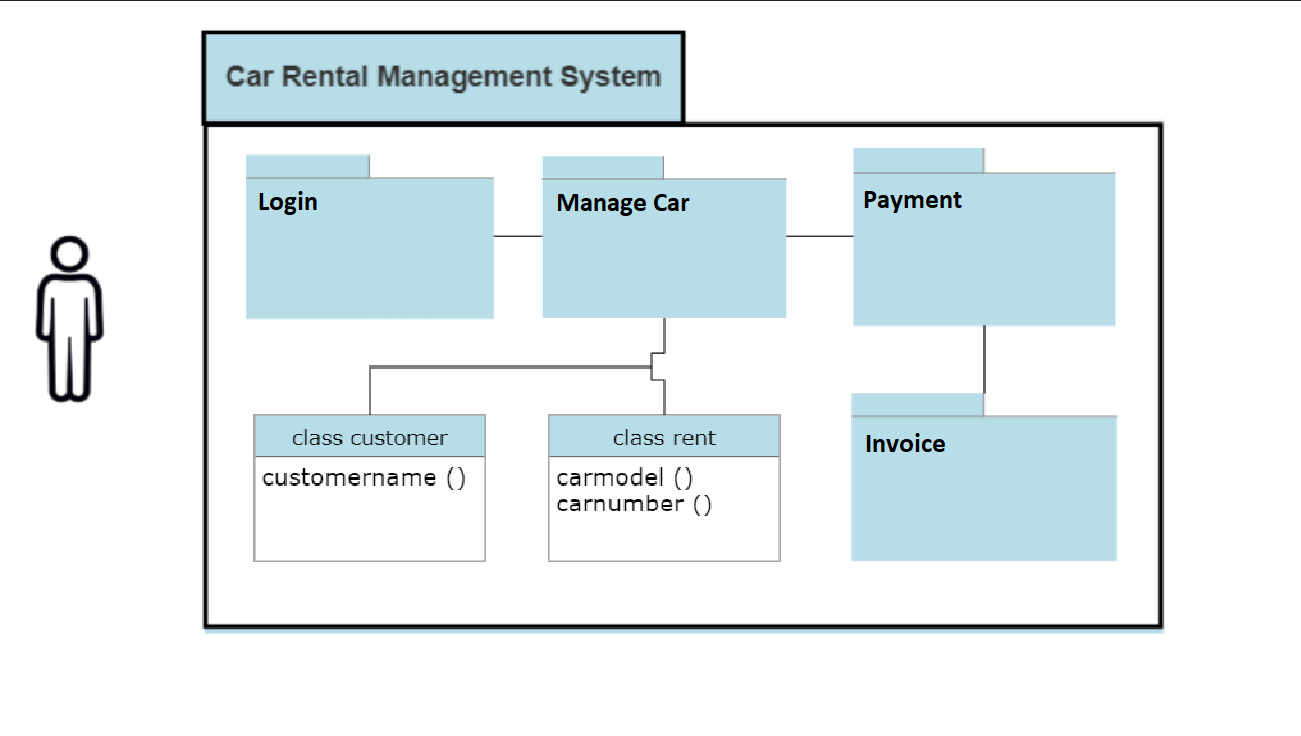
* Login transition: from initial state to login state when the user chooses to login
* Car selection transition: from login state to car selection state when the user successfully logs in
* Rental information transition: from car selection state to rental information state when the user selects a car
* Calculation transition: from rental information state to calculation state when the user provides rental information
* Invoice transition: from calculation state to invoice state when the rental fee is calculated

1. **Activity Diagram :-**



The activity diagram for the given code starts with the user entering their name, selecting a car model from three different options, entering the number of cars to be rented and the number of days they want to rent them for. After this, the program calculates the rental fee based on the selected car model and the number of days rented. Finally, the program displays a customer invoice showing the details of the rental.

**G. Package** **Diagram :-**

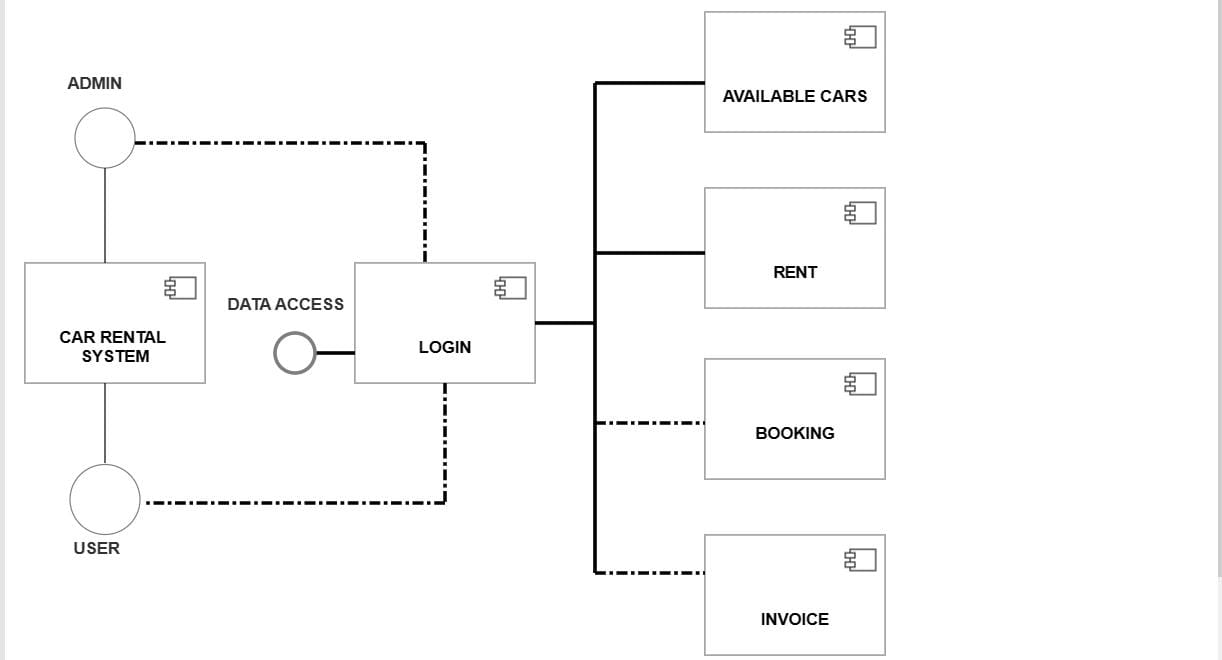


A package diagram in UML provides an overview of the packages and their dependencies in a system. A package is a grouping of related elements, such as classes, interfaces, and other packages, that form a single logical unit.

In this code, there are no packages defined explicitly. However, we can create a package diagram based on the classes that are defined in the code.The diagram will have two packages, one for the standard library classes (iostream, fstream, conio, stdlib, unistd, and dos), and another for the classes defined in the code (customer and rent). The rent class inherits from the customer class.

The diagram shows the two packages and their contents. The standard library package contains the six standard library classes, and the code classes package contains the customer and rent classes. The rent class inherits from the customer class.

**H. Component Diagram :-**

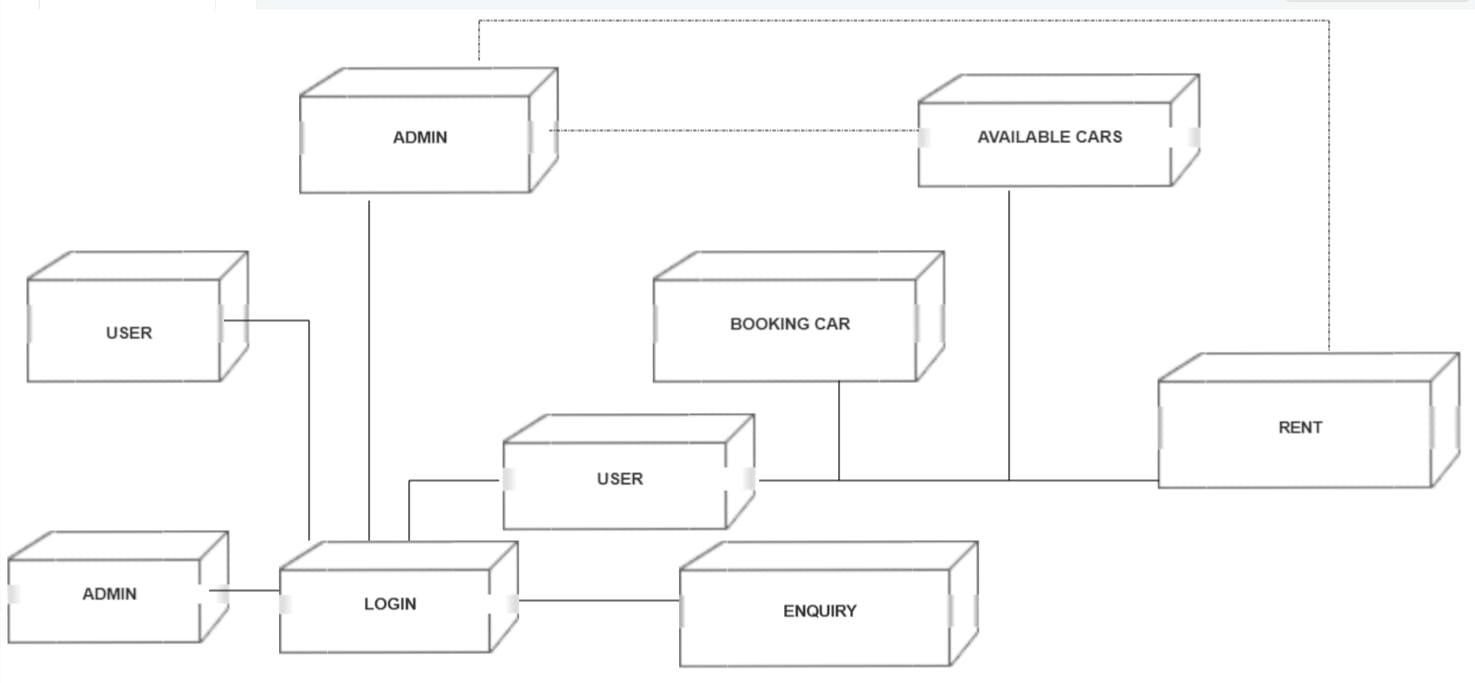


The diagram shows two components, Rental and Customer. The Rental component has three public methods, data(), calculate(), and showrent(), and three private data members, rentalfee, carmodel, and carnumber, as well as a private days data member. The Customer component has four public data members, customername, carmodel, carnumber, and days, and one public method, data(). The Rental component extends the Customer component, indicating that it inherits all the public data members and methods from the Customer component.

The Rental component depends on the Customer component, as it inherits from it and uses its public data members. The Rental component also has a private login() method, which is not shown in the diagram, but could be represented by a separate component. The Rental component also has dependencies on the iostream, fstream, conio, stdlib, unistd, dos, and iomanip libraries, which are not shown in the diagram.

Overall, the component diagram shows the different components that make up the system and their relationships, which can help in understanding the overall architecture of the system.

**I.Deployment Diagram :-**



The diagram shows that the console application runs on a computer, which hosts an operating system and the required libraries to execute the application. The source code and files required to build the application are stored separately and can be deployed to the computer for execution.

1. **Code/Output Screenshots**

#include <iostream>

#include <fstream>

#include <conio.h>

#include <stdlib.h>

#include <unistd.h>

#include <dos.h>

#include <iomanip>

using namespace std;

class customer // customer class

{

private:

public:

string customername;

string carmodel;

string carnumber;

char data;

// variables defined in this class in public mode.

};

class rent : public customer // inhereted class from customer class

{

public:

int days=0,rentalfee=0; // additional int variables defined

void data()

{

int login();

login();

cout << "\t\t\t\tPlease Enter your Name: "; //taking data from the user

cin >> customername;

cout<<endl;

{

cout <<"\t\t\t\tPlease Select a Car"<<endl; //giving user a choice to select among three different models

cout<<"\t\t\t\tEnter 'A' for Hatchback "<<endl;

cout<<"\t\t\t\tEnter 'B' for Sedan "<<endl;

cout<<"\t\t\t\tEnter 'C' for SUV"<<endl;

cout<<endl;

cout<<"\t\t\t\tChoose a Car from the above options: ";

cin >>carmodel;

cout<<endl;

cout<<"--------------------------------------------------------------------------"<<endl;

if(carmodel=="A")

{

system("CLS");

cout<<"You have choosed Hatchback model"<<endl;

ifstream inA("A.txt"); //displaying details of model A

char str[200];

while(inA) {

inA.getline(str, 200);

if(inA) cout << str << endl;

}

sleep(2);

}

if(carmodel=="B")

{

system("CLS");

cout<<"You have choosed Sedan model "<<endl;

ifstream inB("B.txt"); //displaying details of model B

char str[200];

while(inB) {

inB.getline(str, 200);

if(inB) cout << str << endl;

}

sleep(2);

}

if(carmodel=="C")

{

system("CLS");

cout<<"You have choosed SUV model "<<endl;

ifstream inC("C.txt"); //displaying details of model C

char str[200];

while(inC) {

inC.getline(str, 200);

if(inC) cout << str << endl;

}

sleep(2);

}

if(carmodel !="A" && carmodel !="B" && carmodel !="C" )

cout<<"Invaild Car Model. Please try again!"<<endl;

}

while(carmodel !="A" && carmodel !="B" && carmodel !="C" );

cout<<"--------------------------------------------------------------------------"<<endl;

cout << "Please provide following information: "<<endl;

//getting data from user related to rental service

cout<<"Please select Number of Cars to be rented : ";

cin >> carnumber;

cout<<"Number of days you wish to rent the car : ";

cin >> days;

cout<<endl;

}

void calculate()

{

sleep(1);

system ("CLS");

cout<<"Calculating rent. Please wait......"<<endl;

sleep(2);

if(carmodel == "A"||carmodel=="a")

rentalfee=days\*56;

if(carmodel == "B" ||carmodel=="b")

rentalfee=days\*60;

if(carmodel == "C" ||carmodel=="c")

rentalfee=days\*75;

}

void showrent()

{

cout << "\n\t\t Car Rental - Customer Invoice "<<endl;

cout << "\t\t

///////////////////////////////////////////////////////////"<<endl;

cout << "\t\t| Invoice No. :"<<"------------------|"<<setw(10)<<"#Cnb81353"<<" |"<<endl;

cout << "\t\t| Customer Name:"<<"-----------------|"<<setw(10)<<customername<<" |"<<endl;

cout << "\t\t| Car Model :"<<"--------------------|"<<setw(10)<<carmodel<<" |"<<endl;

cout << "\t\t| Car No. :"<<"----------------------|"<<setw(10)<<carnumber<<" |"<<endl;

cout << "\t\t| Number of days :"<<"---------------|"<<setw(10)<<days<<" |"<<endl;

cout << "\t\t| Your Rental Amount is :"<<"--------|"<<setw(10)<<rentalfee<<" |"<<endl;

cout << "\t\t| Caution Money :"<<"----------------|"<<setw(10)<<"0"<<" |"<<endl;

cout << "\t\t| Advanced :"<<"---------------------|"<<setw(10)<<"0"<<" |"<<endl;

cout << "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout <<"\n";

cout << "\t\t| Total Rental Amount is :"<<"-------|"<<setw(10)<<rentalfee<<" |"<<endl;

cout << "\t\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout << "\t\t # This is a computer generated invoice and it does not"<<endl;

cout << "\t\t require an authorised signture #"<<endl;

cout <<" "<<endl;

cout << "\t\t///////////////////////////////////////////////////////////"<<endl;

cout << "\t\tYou are advised to pay up the amount before due date."<<endl;

cout << "\t\tOtherwise penelty fee will be applied"<<endl;

cout << "\t\t///////////////////////////////////////////////////////////"<<endl;

int f;

system("PAUSE");

system ("CLS");

ifstream inf("thanks.txt");

char str[300];

while(inf) {

inf.getline(str, 300);

if(inf) cout << str << endl;

}

inf.close();

}

};

class welcome //welcome class

{

public:

int welcum()

{

ifstream in("welcome.txt"); //displaying welcome ASCII image text on output screen fn1353

if(!in) {

cout << "Cannot open input file.\n";

}

char str[1000];

while(in) {

in.getline(str, 1000); // delim defaults to '\n' cp

if(in) cout << str << endl;

}

in.close();

sleep(1);

cout<<"\nStarting the program please wait....."<<endl;

sleep(1);

cout<<"\nloading up files....."<<endl;

sleep(1); //function which waits for (n) seconds

system ("CLS"); //cleares screen

}

};

int main()

{

welcome obj1; //object created for welcome class

obj1.welcum(); //welcum function is called

rent obj2;

//object created for rent class and further member functions are called

obj2.data();

obj2.calculate();

obj2.showrent();

return 0; //end of the program

}

int login(){

string pass ="";

char ch;

cout<<"\n\n\n\n\n\n\n\n\t\t\t\t\t CAR RENTAL SYSTEM \n\n";

cout<<"\t\t\t\t\t------------------------------";

cout<<"\n\t\t\t\t\t\t LOGIN \n";

cout<<"\t\t\t\t\t------------------------------\n\n";

cout << "\t\t\t\t\tEnter Password: ";

ch = \_getch();

while(ch != 13){//character 13 is enter

pass.push\_back(ch);

cout << '\*';

ch = \_getch();

}

if(pass == "pass"){

cout << "\n\n\n\t\t\t\t\t\tAccess Granted! \n";

system("PAUSE");

system ("CLS");

}else{

cout << "\n\n\t\t\t\t\t\t\tAccess Aborted...\n\t\t\t\t\t\t\tPlease Try Again\n\n";

system("PAUSE");

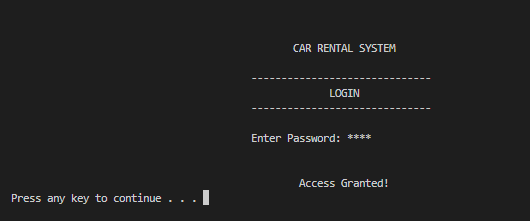
system("CLS");

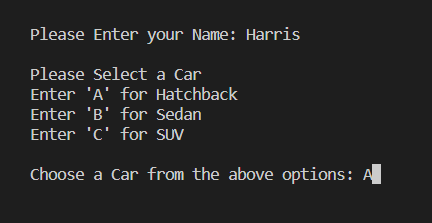
login();

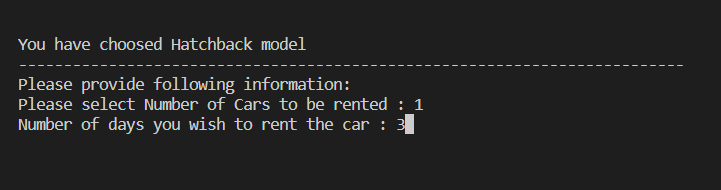
}

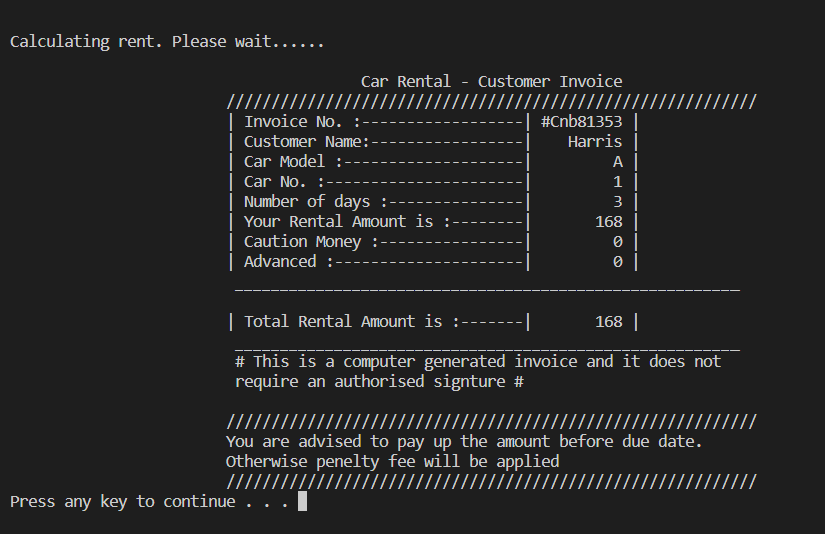
}

**Output: - Screenshots**









**5. Conclusion and Results**

The Car Rental Management System provides an efficient way for car rental companies to manage their fleet of vehicles, bookings, and customer data. By automating many of the manual processes involved in car rental operations, companies can improve their efficiency, responsiveness to customer inquiries, and overall customer experience. The system is easy to use and provides accurate rental fees based on the car model and rental days.

**6. References :-**

* C++ Programming Language
* GeeksforGeeks Tutorials
* StackOverflow Community