A black and white image of a building

Description automatically generated with medium confidence

Final Project

DOCUMENTATION

**Course:** Basics of Programming II

**Instructors:** Mr. Dunaev Dmitriy & Mr. Al-Magsoosi Husam

**Program:** Car Rental System

**Done By:** Maged Daoud

**Neptun Code:** I9XF28

**Contents**

[Introduction: 3](#_Toc135485467)

[Problem Statement: 3](#_Toc135485468)

[Class Diagram: 4](#_Toc135485468)

[Code Explanation: 5](#_Toc135485469)

[Class Car: 5](#_Toc135485470)

[Explanation: 5](#_Toc135485471)

[Class Rental: 6](#_Toc135485472)

[Explanation: 6](#_Toc135485473)

[Explanation: 7](#_Toc135485474)

[Class Car Rental (Main Class): 8](#_Toc135485475)

[Explanation: 8](#_Toc135485476)

[Function in Main Class: 8](#_Toc135485477)

[1. The loadCarData function loads car data from a text file. It reads the data line by line and creates Car objects using the read data. The cars are then stored in a vector. 8](#_Toc135485478)

[File Management: 9](#_Toc135485479)

[Cars Data Loading: 9](#_Toc135485480)

[Cars Data Saving: 10](#_Toc135485481)

[Cars Data Text File: 10](#_Toc135485482)

[Rental Data Text File: 11](#_Toc135485483)

[Future Enhancements: 12](#_Toc135485484)

[Conclusion: 12](#_Toc135485485)

[References: 13](#_Toc135485486)

**Introduction:**

This Car Rental Management System is a small software console-based application developed in C++ language using Object Oriented Programming (OOP) to manage the rental of cars.

This system allows users to perform various operations, such as adding new cars, renting cars, view rental cars history, and deleting a car.

The system is implemented using C++ programming language and provides a console based with command line interface for user doing operation.

# Problem Statement:

The car rental industry is growing rapidly, and there is a need for efficient management systems to streamline the rental process. Traditional manual methods for managing car rentals can be time-consuming and error-prone. Therefore, the aim of this project is to develop a Car Rental Management System that automates the process and provides a reliable and user-friendly solution.

The system consists of several key functionalities, including the ability to add new cars to the rental inventory, rent cars to customers, track car availability, and delete a car. The system stores car data, such as car ID, year, model, make, daily price, and availability, in a text file. It allows users to load car data from the file and update the file with any changes made to the car inventory.

To ensure data integrity, the system includes validation checks for user inputs. For example, when adding new cars or renting cars, the system verifies that the input values are of the correct data type and use exception handling (try catch while reading or load data from file).

In case of invalid input, the system prompts the user to enter valid data.

The Car Rental Management System aims to simplify the rental process for both the rental company and the customers. It provides an efficient and organized way to manage car inventory, track rentals, and generate reports for analysis and decision-making. By automating manual tasks and reducing the chances of errors, the system improves overall productivity and customer satisfaction.

**Class Diagram**:

I implemented the solution using object-oriented programming (OOP) that uses the concepts of classes and objects. I have created 3 classes **Car**, **Rental** and **Customer**. each class has private attributes that represent the features of each class and public methods that perform the operations.

You can see from the class diagram that **Rental** class has aggregation relationship with **Car** and **Customer** classes that means This means that an instance of the **Rental** class contains a pointer to an instance of the **Car** class.

A picture containing text, screenshot, diagram, document

Description automatically generated

# Code Explanation:

# Class Car: A screenshot of a computer code Description automatically generated with low confidence

## Explanation:

The class **"Car"** represents a car object. It has Private member variables such **as ID, Year, Model, Make, daily\_price, and isAvailable**.

The class has a constructor to initialize these variables, as well as getter and setter methods for accessing and modifying the data members.

The constructor takes in parameters value of all data member and assign values to the member variables. The getter and setter method in class also added to access or modify data members.

The method in class **printCarDetails()** that prints out the details of a car object, including the ID, Year, Model, Make, daily price, and availability status.

# Class Rental:

A screenshot of a computer program

Description automatically generated with medium confidence

## Explanation:

The class has private member variables: **start\_date and end\_date of type time\_t**, and pointers to objects of the "**Car**" and "**Customer**" classes.

The class has a constructor that takes parameters to initialize the **start\_date, end\_date, car**, and customer objects. The constructor **dynamically allocates memory** for the car and customer objects using the provided parameters.

The class also includes getter and setter methods for **start\_date, end\_date, car, and customer,** allowing access to and modification of these member variables.

The class has a **printRentalDetails()** method, which outputs the rental details, including the start and end dates, customer details, and car details.

**Class Customer:**

A screenshot of a computer program

Description automatically generated with low confidence

## Explanation:

The class has private member variables: name, Age, email, and phone\_number, all of which are of type string or int.

The class has a constructor that takes parameters to initialize the member variables **name, Age, email, and phone\_number**. The constructor assigns the parameter values to the corresponding member variables using this pointer.

The class also includes getter methods for accessing the private member variables: **get\_name(), get\_Age(), get\_phone\_number(), and get\_email().** These methods return the values of the respective member variables.

The class has a **printCustomerDetails()** method, which outputs the customer's details, including the **name, age, email, and phone number**.

# Class Car Rental (Main Class):

## A screenshot of a computer program Description automatically generated with medium confidence

## Explanation:

This is our main class that provide the user menu to perform operation on **Car Rental Management System**, such as adding a new car, renting a car, displaying car details, displaying rental details, removing a car, exit.

The program will not end until user want to exit the system.

## Function in Main Class:

## The loadCarData function loads car data from a text file. It reads the data line by line and creates Car objects using the read data. The cars are then stored in a vector.

1. The **getValidIntegerInput** function prompts the user with a message and accepts an integer input. It validates the input and only allows valid integer values. If the input is not a valid integer, it displays an error message and asks for input again.
2. The **convertStringToTimestamp** function takes a string date in the format "DD:MM:YYYY" and converts it to a timestamp in second.
3. The **loadRentalData** function loads rental data from a file. Similar to **loadCarData,** it reads the data line by line and creates Rental objects using the read data. The rentals are then stored in a vector.
4. The **saveCarData** function saves the car data to a file. It opens the file and writes the car details in the specified format.
5. The **saveRentalData** function saves the rental data to a file. It opens the file and writes the rental details in the specified format.

The main function is the entry point of the program. It initializes vectors for **cars** and **rentals**, and then it loads the existing data from text files using the **loadCarData** and **loadRentalData** functions.

# File Management:

## Cars Data Loading:

A picture containing text, screenshot, font, number

Description automatically generated

The function **loadCarData**, is responsible for loading car data from a text file. It takes a single parameter, filename, which is the name of the file to load the data from.

Inside the function, it creates an input file stream (ifstream) and opens the specified file. If the file is successfully opened, it enters a loop to read the data line by line. For each line, it extracts the ID, year, model, make, daily price, and availability values using the input operator (>>), and creates a Car object with these values. The newly created Car object is then added to the cars vector which store the pointers to these objects. After reading all the car data, the file is closed. If the file fails to open, it throws a runtime\_error with an appropriate error message.

At end of adding all cars it shows the message of how many cars object is added from file. (Actually, array size).

## Cars Data Saving:

A picture containing text, screenshot, font, line

Description automatically generated

The function **saveCarData**, is responsible for saving car data to a text file. It takes two parameters: filename, which is the name of the file to save the data to, and cars, which is a vector containing the car objects to be saved.

It creates an output file stream (ofstream) and opens the specified file. If the file is successfully opened, it iterates over each car object in the cars vector. For each car, it retrieves its ID, year, model, make, daily price, and availability using getter methods (car.get\_ID(), car.get\_year(), etc.), and writes these values to the file separated by spaces. Each car's data is written on a new line. After writing all the car data, the file is closed. If the file fails to open, it throws a runtime\_error with an appropriate error message.

At end of adding all rental data it shows the message of how many rental objects is added from file. (Actually, array size).

## Cars Data Text File:

The text file from where the cars data is read.

A screenshot of a computer

Description automatically generated with medium confidence

## 

## Rental Data Text File:

The text file from where the cars data is read.

A screenshot of a computer

Description automatically generated with medium confidence

**Exception Handling and Data Validation:**

The function **getValidIntegerInput** is used to obtain a valid integer input from the user. It takes a string parameter msg which is a message or prompt to be displayed to the user before input.

The user enters a value of it’s a valid integer the loop end and it returns the number and if not than it shows message that value is not valid and prompt user to enter again until get the valid number.

A picture containing text, screenshot, font

Description automatically generated

We use **try catch** to catch exception every where we are loading or saving data into the files.

A picture containing text, font, screenshot, line

Description automatically generated

# Future Enhancements:

1. Adding GUI (Graphical User Interface) to the Application:
2. Adding an actual database like (SQL) to manage and store data instead of text files.
3. Adding more features like Customer Verification or Car inspection.

# Conclusion:

The Car Rental Management System developed in this project provides an effective solution for managing car rentals. By automating key processes such as adding cars, renting cars, and generating reports, the system streamlines the rental process and improves efficiency. The system incorporates validation checks to ensure data integrity and provides a user-friendly interface for easy interaction.

In conclusion, the Car Rental Management System presented in this project offers a demo solution for managing car rentals efficiently. It simplifies the rental process, improves data accuracy, and enhances overall productivity. With further development and enhancements, the system can be customized and expanded to meet the specific needs of car rental businesses

# References:

<https://www.geeksforgeeks.org/file-handling-c-classes/>

<https://www.w3schools.com/cpp/cpp_exceptions.asp>

<https://www.tutorialspoint.com/cplusplus/cpp_exceptions_handling.htm>

<https://www.softwaretestinghelp.com/date-and-time-in-cpp/>

<https://www.tutorialspoint.com/cplusplus/cpp_date_time.htm>

<https://www.javatpoint.com/university-management-system-in-cpp>