

# **Car Rental Management System**

**Diploma in Information Technology**

**ESOFT Metro Campus – Gampaha**

**195 B Group 5**

## Project Details

**Title of Project : Car Rental Management System**

**Batch : DiTEC 195 B**

**Details of the Students :**

No.	Student ID	Registration Number	Student Name
01.	E166775	00132066	Muditha Lakmal (Leader)
02.	E167555	00138069	Yasith Nimsara
03.	E164367	00134699	Yenuka Ranathunga
04.	E165750	00135844	Chanupa Athsara
05.	E161988	00131489	Induja Thathsarani
06.	E162717	00132303	Umesha Nethmi
07.	E168674	00139545	Vishwa Tharanga
08.	E167613	00138136	Tharindu Kavinda
09.	E167122	00137498	Dinuka Dhanujaya
10.	E166844	00137172	Sasmitha Kodikara
11.	E163430	00135540	Amaya Pasinduni

## Abstract

Our Aim is to design and create a data management System for a car rental company. This enables front officer can rent a vehicle that can be used by a customer. This system increases customer retention and simplify vehicle and staff Management in an efficient way.

This software car Rental System has a very user-friendly interface. Thus, the users will feel very easy to work on it. By using this system front officer can manage customer registration, customer information edits and delete. Front officer can also manage booking and return cars. The car information can be added to the system or existed car information can be edited or deleted too by Admin. The actions of the front officer can also be done by the admin.

The customers cannot use the system to get car rent. It should be done through the front officer. This system will helpful to the front officer as well as to the customer also. There is no delay in the availability of any car information, whenever needed, car information can be captured very quickly and easily.

We have learnt a lot during this project and liked the improvement in our testing skills and deep concept related to these kinds of projects. This is basically a tool that makes it easy for the management of the concerned organization to get a detailed report about their company/shop in various sectors without using the internet.

## Acknowledgement

It's a great pleasure to present this documentation on the project named "Easy Car", Car Rental Management System undertaken by us as part of our DiTEC final project. We would like to thank Esoft Metro Campus and its DiTEC program for providing us with such a wonderful challenging opportunity and our sincere thanks to all the coordinators of Esoft Metro Campus Gampaha Branch for providing all possible help and support and their constant encouragement.

First, we would like to express our special thanks to our training guide and coordinator Mr. Gayan Kavinda and his constant interest. He helped us complete this project. Without the honest and sincere guidance of our esteemed project guide we would not have been able to reach the present stage.

# Table of Contents

Project Details	1
Title of Project	1
Batch	1
Details of the Students	1
Abstract	2
Acknowledgement	3
List of Acronyms	7
<b>Chapter 01 – INTRODUCTION</b>	8
<b>Introduction of Car Rental Management System</b>	8
<b>Aim and Objectives</b>	8
Aim	8
Objectives	9
<b>Scope of the Project</b>	9
<b>Resource Requirements</b>	10
System Requirements	10
User Requirements	10
Functional Requirements	11
Non-Functional Requirements	11
<b>Chapter 02 - ANALYSIS</b>	12
<b>SDLC (Software Development Life Cycle)</b>	12
What Is SDLC?	12
Stages Of SDLC	12
<b>Waterfall Model</b>	13
Analysis	14
Design	14
Implementation	15
Testing	15
Maintenance	15

<b>Analysis Stage</b>	15
Review Similar Projects	16
Wire Frames	17
<b>Chapter 03 – DESIGN</b>	18
<b>UML Diagrams</b>	18
Class Diagrams	18
Use Case Diagrams	18
ER Diagrams	18
Class Diagram	19
Use Case Diagram	20
ER Diagram	21
<b>Chapter 04 – DEVELOPMENT</b>	26
<b>Visual Studio</b>	26
<b>SQL Server Management Studio</b>	26
<b>C#</b>	27
<b>Codes of Our Project</b>	27
Login Form	27
Welcome Form	29
Customer Form	32
Car Form	35
Booking Form	37
Return Form	39
<b>Chapter 05 – EVALUATION</b>	42
<b>Types of Testing</b>	42
Black Box Testing	42
White Box Testing	42
Unit Testing	42
System Testing	42
User Acceptance Testing	43
<b>Test Code</b>	43
<b>Chapter 06 – CONCLUSION</b>	44
Gantt Chart	47

## Table of Figures

Figure 1 : SDLC	12
Figure 2 : Waterfall Model	14
Figure 3 : Compare Table	17
Figure 4 : Class Diagram	19
Figure 5 : Use Case Diagram	20
Figure 6 : ER Diagram	21
Figure 7 : Login - Wire Frame	22
Figure 8 : Welcome - Wire Frame	23
Figure 9 : Manage Car - Wire Frame	23
Figure 10 : Manage Customer - Wire Frame	24
Figure 11 : Manage Booking - Wire Frame	24
Figure 12 : Manage Return - Wire Frame	25
Figure 13 : Login Form	27
Figure 14 : Login Form Design	28
Figure 15 : Login Button Coding	28
Figure 16 : Exit Button Coding	29
Figure 17 : Welcome Form	29
Figure 18 : Welcome Form Design	30
Figure 19 : Customer Button Cording	30
Figure 20 : Car Button Coding	30
Figure 21 : Booking Button Coding	31
Figure 22 : Return Button Coding	31
Figure 23 : Logout Button Coding	31
Figure 24 : Customer Form	32
Figure 25 : Customer Form Design	33
Figure 26 : Add Button Coding	33

Figure 27 : Delete Button Coding	34
Figure 28 : Update Button	34
Figure 29 : Next Button Coding	34
Figure 30 : Back Button coding	34
Figure 31 : View All Button Coding	35
Figure 32 : Car Form	35
Figure 33 : Car Form Design	36
Figure 34 : Add Button Coding	36
Figure 35 : Update Button Coding	37
Figure 36 : Booking Form	37
Figure 37 : Booking Form Design	38
Figure 38 : Add Button Coding	38
Figure 39 : Update Button Coding	39
Figure 40 : Return Form	39
Figure 41 : Return Form Design	40
Figure 42 : Add Button Coding	40
Figure 43 : Update Button Coding	41
Figure 44 : Gantt Chart	45

## List of Acronyms

- SQL – Structured Query Language
- RAM – random Access Memory
- UML - Unified Modeling Language
- SDLC – Software Development Life Cycle
- ER Diagram – Entity Relationship Diagram
- SSMS - SQL Server Management Studio



# Chapter 01 – INTRODUCTION

## Introduction of Car Rental Management System

To give a better experience for our customers and to manage the internal staff with a high productivity, we looked forward to creating an application software for “Easy Car” company. Hence, we create a simple application with some effective features to manage staff details, item details etc.

With the previous traditional management system, we must face many problems when performing our ordinary activities. They are;

- We store details on the papers, therefore we lost some data sometimes due to this reason.
- It's very bulky when we use paper documents.
- It took long time to search details of the items when we want them.
- Higher risk of data loss: A manual system can be more susceptible to data loss due to physical damage, theft, or other issues. This can result in lost rental data, customer information, and other critical data that can be difficult or impossible to recover.
- Limited scalability: A manual system may have limited scalability, which can make it difficult to expand operations or integrate with other systems. This can limit a car rental company's ability to grow and compete with other players in the market.
- Inefficient processes: A manual system can be more time-consuming and prone to errors because it relies on manual processes. This can lead to lower productivity levels and decreased customer satisfaction.

## Aim and Objectives

### Aim

Our aim is reduced manual works and decrease employees and increase work efficiency and speed and to make the process easier for customer.

## OBJECTIVES

For solutions for above problems, we hope to propose a car rental system that is secured and enabled users to reserve the vehicle they wanted. The proposed car rental system has been replaced by the traditional way of renting and hiring vehicles. General functions such as adding, editing, and removing information will be added to the software. Other features such as login, check vehicle availability, check vehicle reservation, and so on will be added to the software.

To overcome above challenges, research has focused on developing solutions that can improve the efficiency and performance of car rental management systems. These solutions may include:

- Developing more advanced rental software and hardware systems that can process rental data more quickly and accurately.
- Implementing better data security measures to protect against data breaches and loss.
- Developing more efficient and cost-effective processes for managing rental operations.
- Integrating car rental management systems with online or cloud-based systems to provide real-time access to rental data and improve scalability.

## Scope of the Project

The scope of a car rental management system software project involves identifying the key features and functionalities of the system and determining the requirements needed to meet the needs of the car rental company. The goal of the project is to design and develop a comprehensive software system that can help streamline the rental process, enhance customer experience, and improve operational efficiency.

- The car rental system to keep detail records of both the cars and the customers, the duration they rent car as well as the type of car they rent.
- The system will be mainly design for small a company that renders it car rental services to customers.
- The system will have three levels of access
  - ✓ The Admin
  - ✓ Front officer

## Resource Requirements

Project resources simply mean resources that are required for successful development and completion of project. These resources can be capital, people, material, tool, or supplies that are helpful to carry out certain tasks in project. Without these resources, it is impossible to complete project. In project planning phase, identification of resources that are required for completion of project and how they will be allocated is key element and very important task to do. In project management, some resources that are required are assigned to each task of project to get job done.

---

## System Requirements

### Hardware Requirements

(Microsoft SQL Server, Visual Studio)

- Processor of Intel(R) Core (TM) i3-8130U CPU @ 2.20GHz 2.21GHz
- 12.0 GB RAM
- 64-bit operating system
- 1TB Hard Disk

### Software Requirements

- Microsoft Visual Studio 2022
- C# .net frame work
- MS SQL server 19
- Microsoft Windows 11 OS
- MS Word
- MS PowerPoint
- Google Browser
- 

---

## User Requirements

- Basically, knowledge to handle Microsoft Windows 10
- MS Office
- Knowledge to
  - Use visual studio
  - Use SQL server management studio

---

## Functional Requirements

Functional requirements can be most simply defined as something the system must do. If the system does not meet a functional requirement it will fail. This is because it will not be able to achieve something it must do to operate properly. The functional requirement concept can also be understood through reviewing the system in terms of inputs and outputs. Functional requirements specify what the system must do in response to different inputs and what it must output.

- The system should be able to manage information about the rental cars, including their availability, and maintenance records.
- The system should be able to manage customer information, including their contact details, rental history, and payment status.
- The system should provide tools for managing
- The system should display the number of bookings, number of customers and number of cars in over shop.

---

## Non-Functional Requirements

Non-functional requirements in software engineering can be explained as requirements that describe how the system works. Non-functional requirements are focused on how the system goes about delivering a specific function. At first glance they might be less important than functional requirements, but both have a part to play in a good system. Non-functional requirements do not have an impact on the functionality of the system, but they do impact on how it will perform. In short, non-functional requirements are all about system usability.

Next Chapter will describe how we build our system, using first stage of Waterfall Methodology.

## Chapter 02 - ANALYSIS

Defining requirements is considered part of planning to determine what the application is supposed to do and its requirements. The critical phase of the project is the analyzing part. Because it will design the output of the project. Our ideation will execute in this phase. For this, we can use other developed application as a model, and we can get ideas with that projects.

The software development process model is also called SDLC – Software Development Life Cycle, as the process is repeated when the software is needed to be changed while the software is in use.

### SDLC (Software Development Life Cycle)

#### What Is SDLC?

SDLC stands for Software Development Life Cycle. SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use.

#### Stages Of SDLC

There are seven primary stages of the modern system development life cycle. Here's a brief breakdown:

- Planning Stage
- Feasibility or Requirements of Analysis Stage
- Design and Prototyping Stage
- Software Development Stage
- Software Testing Stage
- Implementation and Integration
- Operations and Maintenance Stage

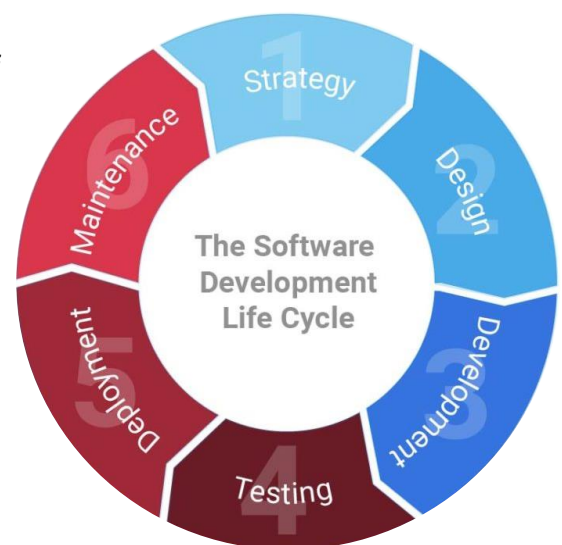


Figure 1 : SDLC

There are different software process models used in the industry and the choice of the model should be based on the product and project needs. Popular SDLC models include the waterfall model, spiral model, and Agile model. The reasons for using Waterfall model as a software process model, in our software are given below.

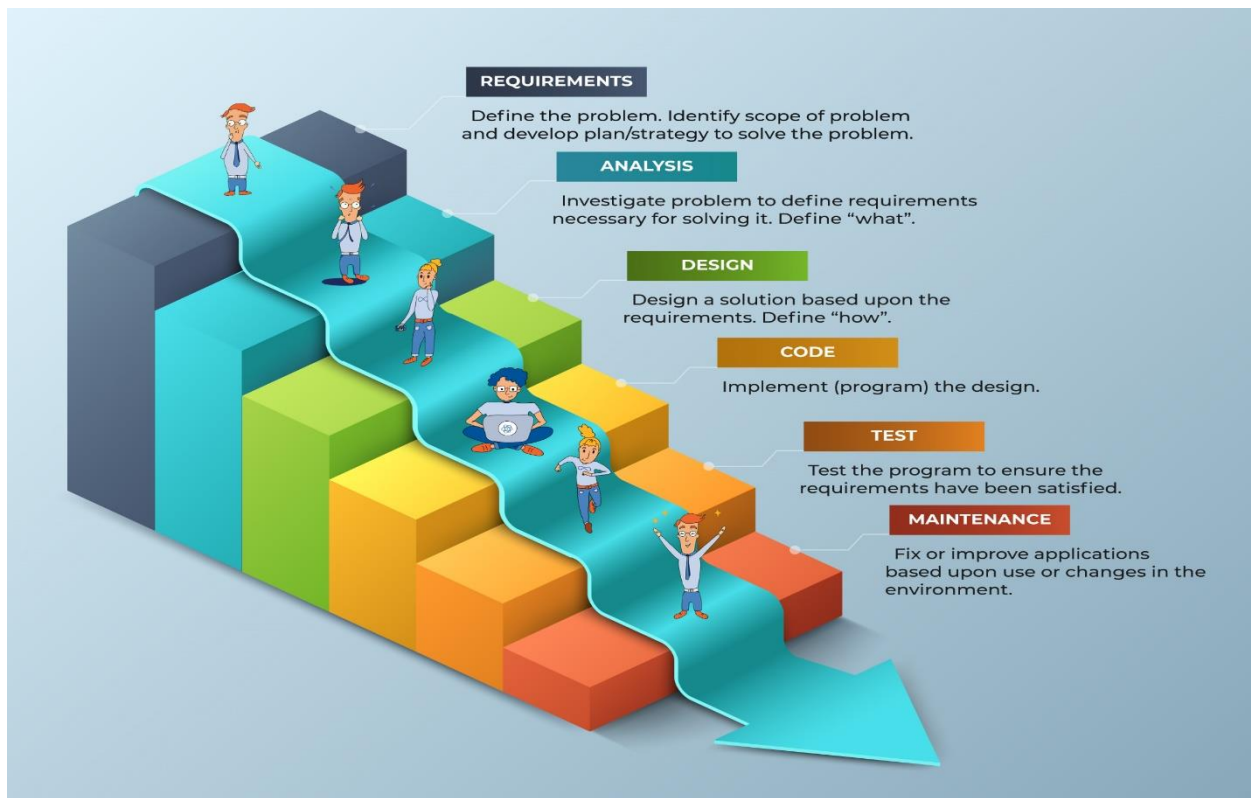
- The waterfall methodology works well when the project requirements are stable and well-understood upfront. It requires a comprehensive and detailed specification document at the beginning of the project, which serves as a blueprint throughout the development process.
- The waterfall model provides a predictable and structured approach to software development. Since each phase is completed before moving on to the next, it allows for better estimation of time, resources, and costs. This makes it easier to plan and manage the project.
- The waterfall methodology is suitable for projects that have a strict order of tasks and dependencies. It allows for a step-by-step progression from requirements gathering, design, development, testing, and deployment. This sequential flow makes it easier to track progress and identify potential issues at each stage.
- The waterfall model emphasizes documentation throughout the development lifecycle. This documentation serves as a reference for project stakeholders, including developers, testers, and end-users. It ensures better knowledge transfer and reduces the risk of miscommunication.

## Waterfall Model

### INTRODUCTION

Waterfall model is a simple model that has a set of stages, those are to be followed one after the other (move from one phase to other). Using this model progress monitoring is easier because the development process is more visible than other models, and also each stage has an output (a product to deliver), Stages of Waterfall model are,

1. Requirements Analysis & Definition
2. System & Software Design
3. Implementation & Unit testing
4. Integration & System testing
5. Operations & Maintenance



**Figure 2 : Waterfall Model**

## Analysis

The linear nature of the Waterfall methodology gives additional significance to this first stage in its life cycle. All requirements of the final software product's utility and features must be gathered here.

Once the requirements analysis stage is completed the software development team should have all the information needed to complete the project without any, or minimal, further involvement from whoever has contracted or otherwise. initiated the project.

## Design

The design phase is often divided into two subphases – logical (or preliminary) design and physical (or detailed) design. The logical or preliminary design phase involves putting all possible solutions on the table and analyzing their strengths and weaknesses within the context.

Once theoretical ideas have been assessed and decisions taken on which to go with, the physical or detailed design phase is when they are documented and detailed as concrete specifications.

---

## Implementation

This fourth chapter will cover all the issues in implementation. Here the important codes are listed and explained. Implementation environment such as hardware and software and other related issues are summarized, the methods and strategies used.

---

## Testing

When the implantation phase has been fully completed, manual software testers (who might be supported by automated testing tools in contemporary software development projects) have to make sure every component of the software system works as intended, both autonomously and across any dependencies.

Testers will use documentation created at the design phase, user personas and user journey scenarios to run as many test cases as possible in the attempt to uncover any bugs that need to be fixed before deployment.

---

## Maintenance

In the maintenance phase, the software is in use and the primary job is now to keep it available and running smoothly as well as fixing any bugs reported by users that may have been missed during the testing phase.

# IN OUR PROJECT,

## Analysis Stage

In the requirement analysis stage of the waterfall methodology for developing a car rental system, we need to perform several tasks to gather and define the project requirements. Here are the key activities we should undertake,

- Identify all the individuals or groups who have an interest or will be affected by the car rental system



- Engage with stakeholders to collect their needs and expectations from the car rental system. Conduct interviews, surveys, and workshops to gather comprehensive information about the desired features, functionalities, and constraints.
- Analyze the collected requirements and prioritize them based on their importance and impact on the system. Categorize them as "must-have," "should-have," or "nice-to-have" to ensure focus on essential features.
  - **Must Have**
    - ✓ Customer Details
    - ✓ Car Details
    - ✓ Booking Details
    - ✓ Return Details
  - **Should Have**
    - ✓ Car rental prices
    - ✓ Time durations
    - ✓ Driver Details
    - ✓ Car Location

---

## Review Similar Projects

By reviewing with similar projects, we can correct our mistakes and get some new things from those projects. We can go with its main functions and can make changes to them with our favor.

The review typically involves examining the features, strengths, and weaknesses of other systems in the market, including their architecture, user interface, and functionality. It also involves analyzing the system's impact on user experience and efficiency.

Here we are going to compare our application with “EqliPS Car”

	Easy Car	EqliPS Car
Model	Waterfall Model	Waterfall Model
Platform	Visual Studio & SQL	Standalone solution
Login page	✓	✓
Welcome Page	✓	✓
Customer Page	✓	✓
Car Page	✓	✓
Booking Page	✓	✓
Return Page	✓	✓

Figure 3 : Compare Table

Next Chapter will describe how we build our system, using first second stage of Waterfall Methodology.

## Chapter 03 – DESIGN

The system design process partitions the requirements to either hardware or software systems. It establishes overall system architecture. Software design involves representing the software system function in a form that may be transformed into one or more executable programs.

### UML Diagrams

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

---

#### Class Diagrams

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class.

In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.

---

#### Use Case Diagrams

As the most known diagram type of the behavioral UML types, Use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact. It's a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system.

---

#### ER Diagrams

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. They use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict

the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

## IN OUR PROJECT,

Class Diagram, Use Case Diagram and ER Diagram are commonly used in software development to represent various aspects of a system. In the case of an offline car rental system, these diagrams can be used for the following purposes:

### Class Diagram

Class diagrams are used to model the classes, attributes, and methods in the system. In the case of a car rental system, the classes could be Customer, Car, Booking and Rental, and the attributes and methods of these classes could be things like the name and contact information of a customer, the make and model of a car, the start and end date of a rental, etc.

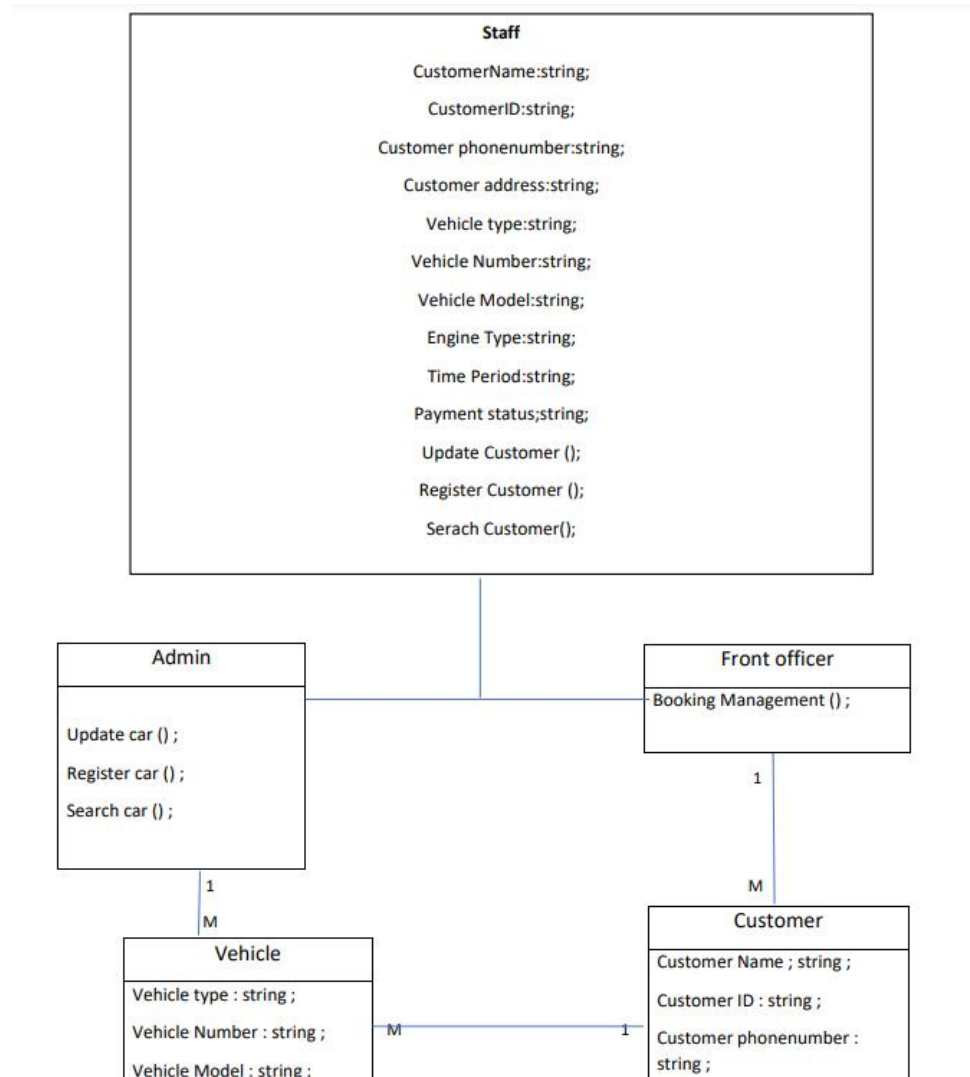


Figure 4 : Class Diagram

## Use Case Diagram

Use case diagrams are used to model the different use cases or scenarios in the system. In the case of a car rental system, the use cases could be things like making a reservation, picking up a booking, returning a booking, etc.

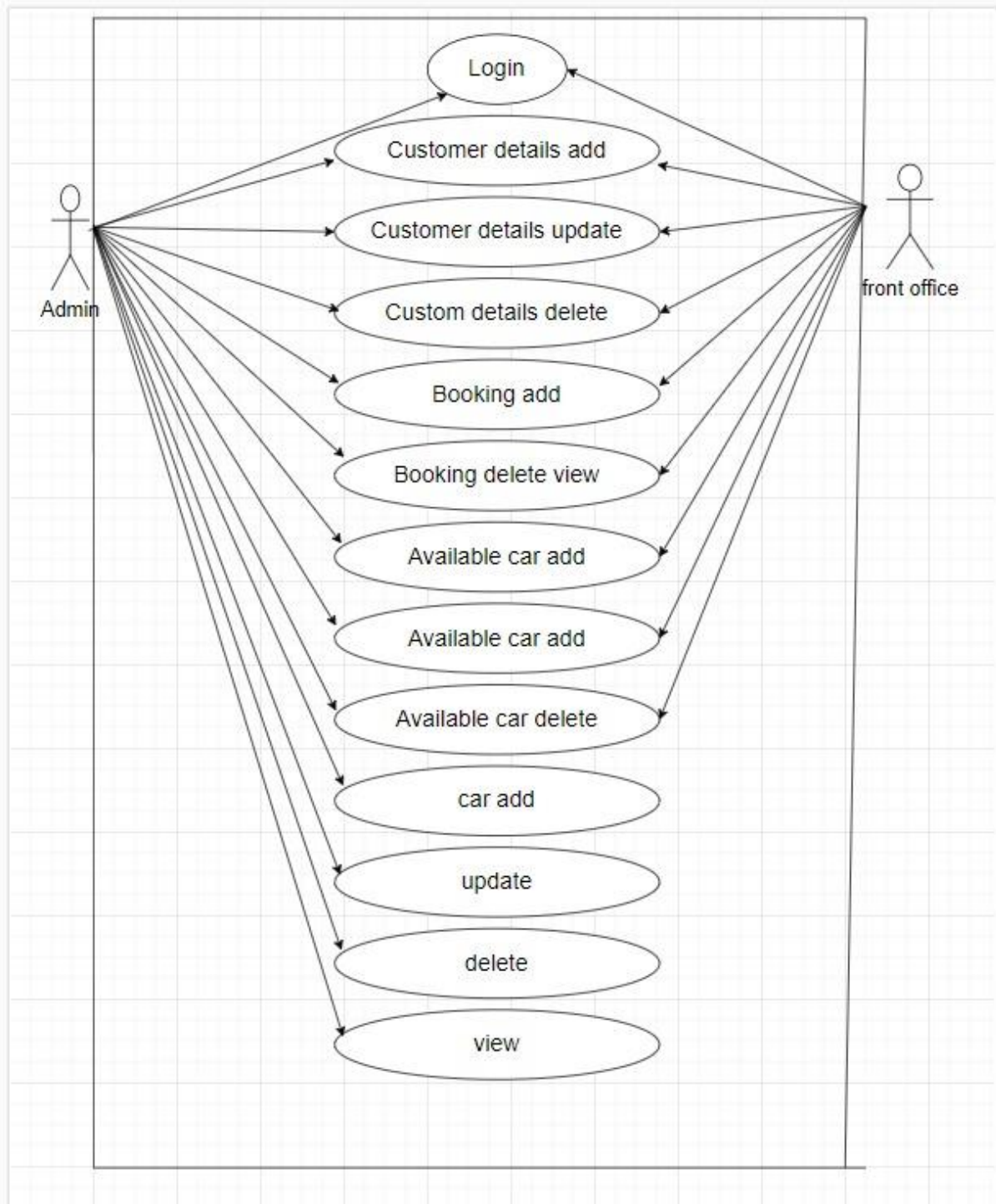


Figure 5 : Use Case Diagram

## ER Diagram

ER diagrams are used to model the data entities and the relationships between them in the system. In the case of a car rental system, the entities could be customers, cars, rentals, reservations, etc., and the relationships between them could be the rental history of a customer, the availability of a car, the duration of a rental, etc.

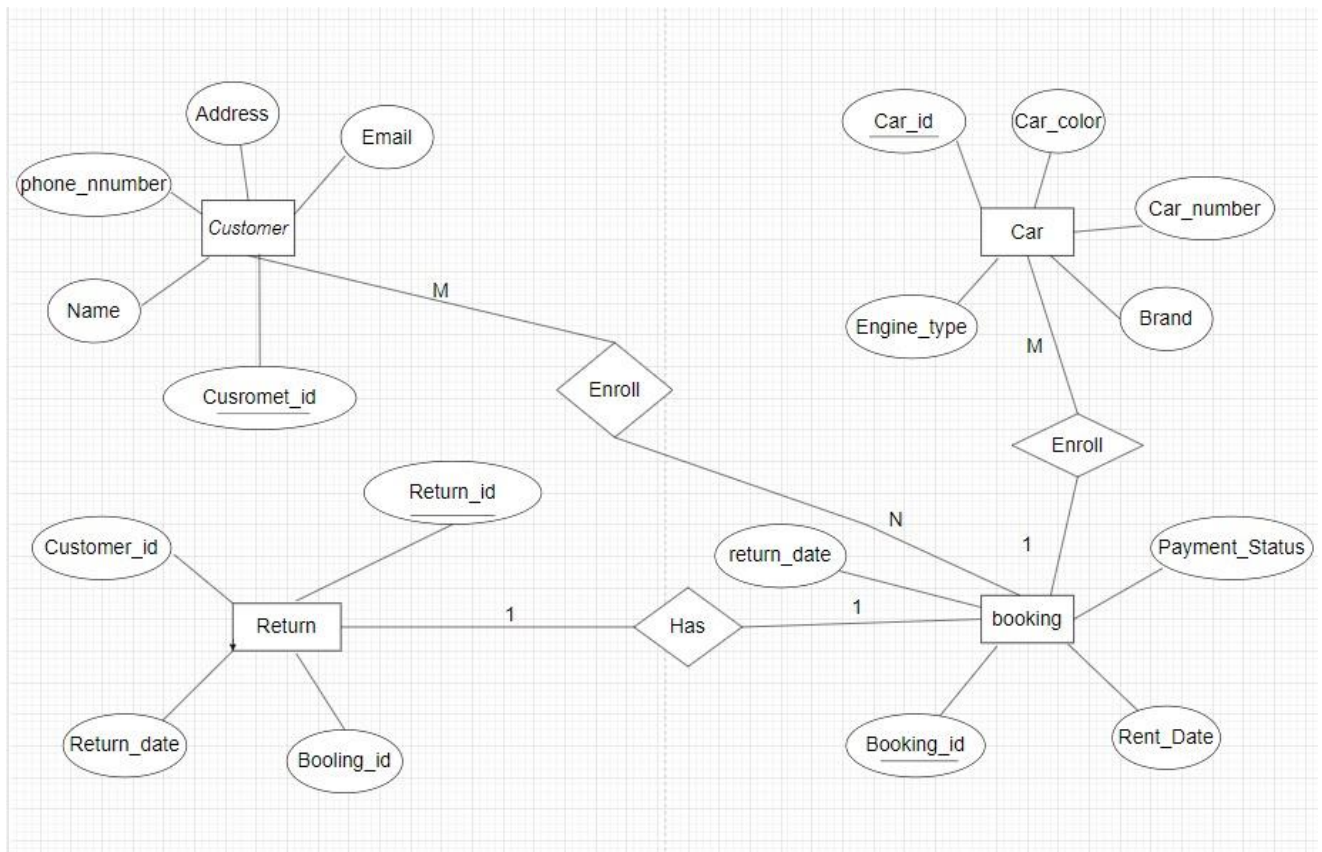


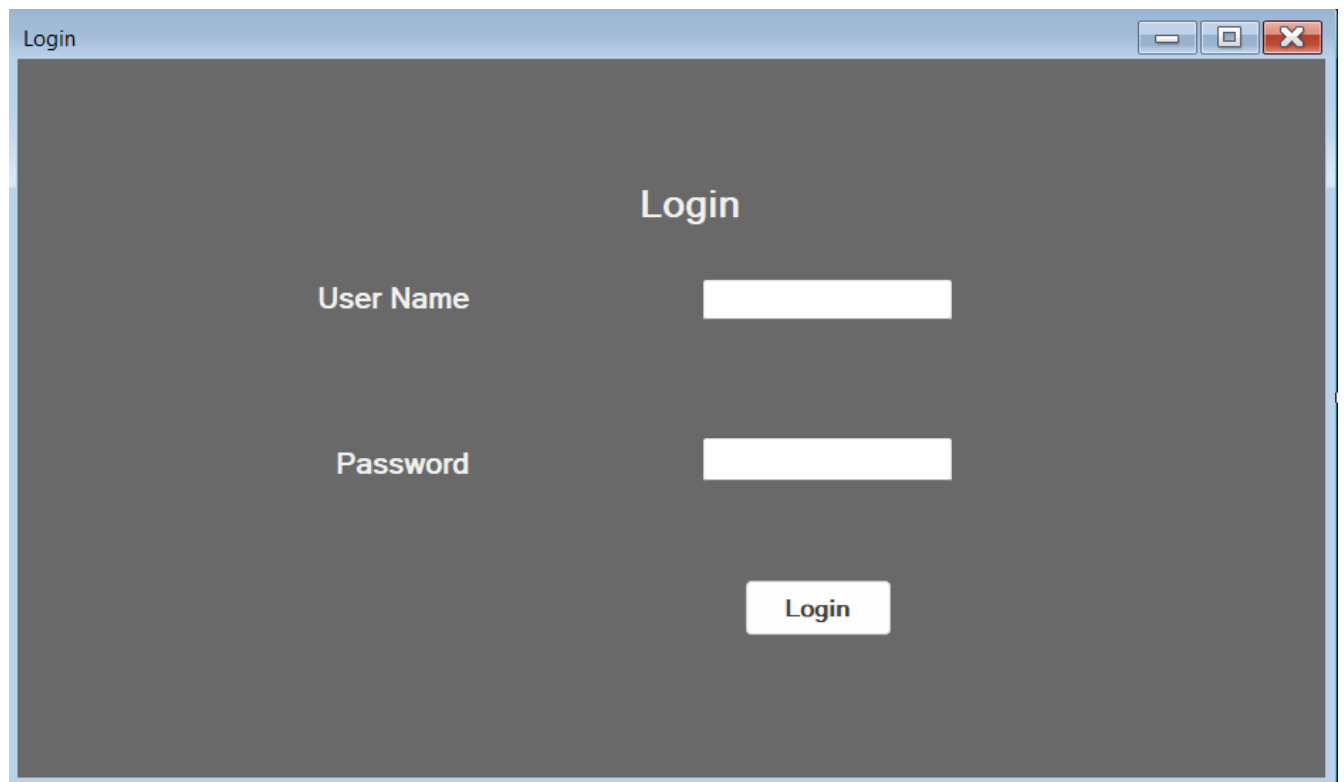
Figure 6 : ER Diagram

## Wire Frames

Wireframes are simple, low-fidelity graphical representations of a software application's user interface. They are used in software development to help stakeholders visualize the layout, content, and functionality of a system before actual design and development work begins. In the case of an offline car rental system, wireframes can be used to:

- Visualize the layout and flow of the user interface.
- Clarify requirements and design decisions
- Save time and money

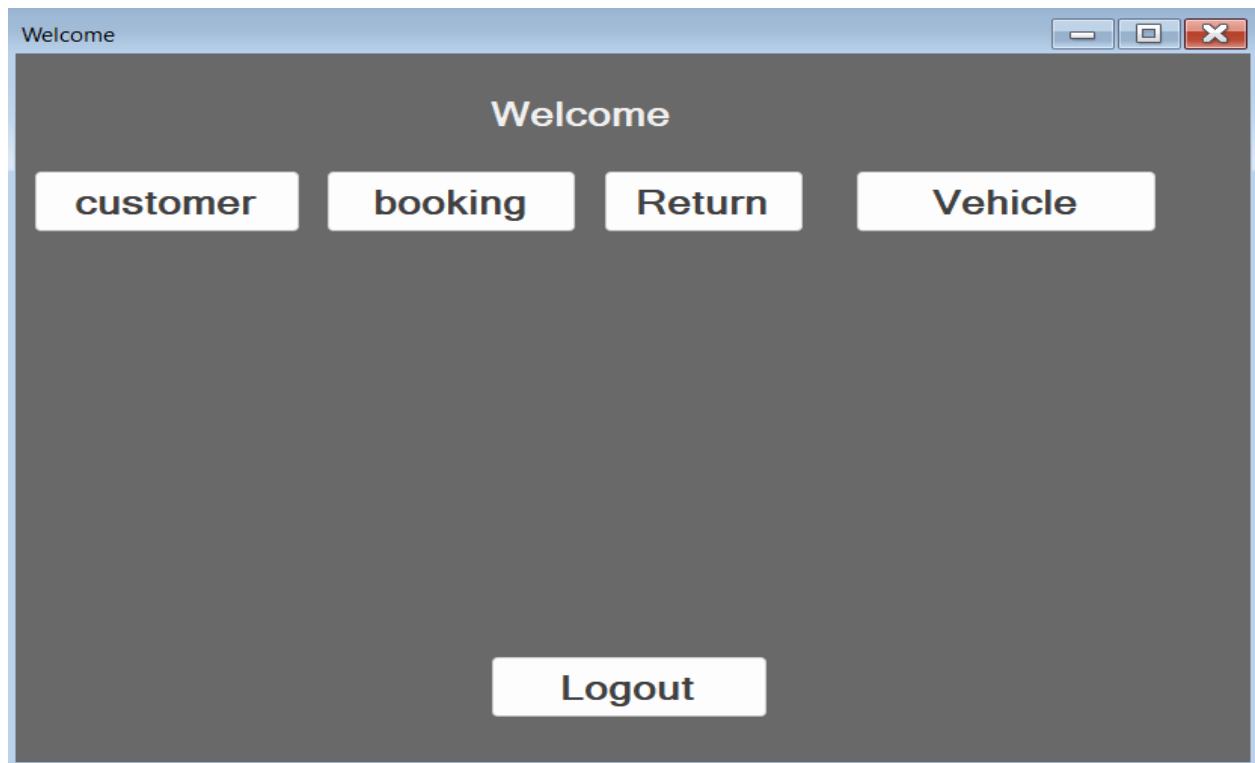
### Login Form



The image shows a wireframe of a login form. It is presented as a window with a title bar that says "Login" and standard minimize, maximize, and close buttons. The main area of the window has a dark gray background. Centered at the top is the word "Login" in a light gray font. Below this, there are two input fields. The first is labeled "User Name" and the second is labeled "Password", both in a light gray font. At the bottom center, there is a light gray button with the word "Login" in a dark gray font.

Figure 7 : Login - Wire Frame

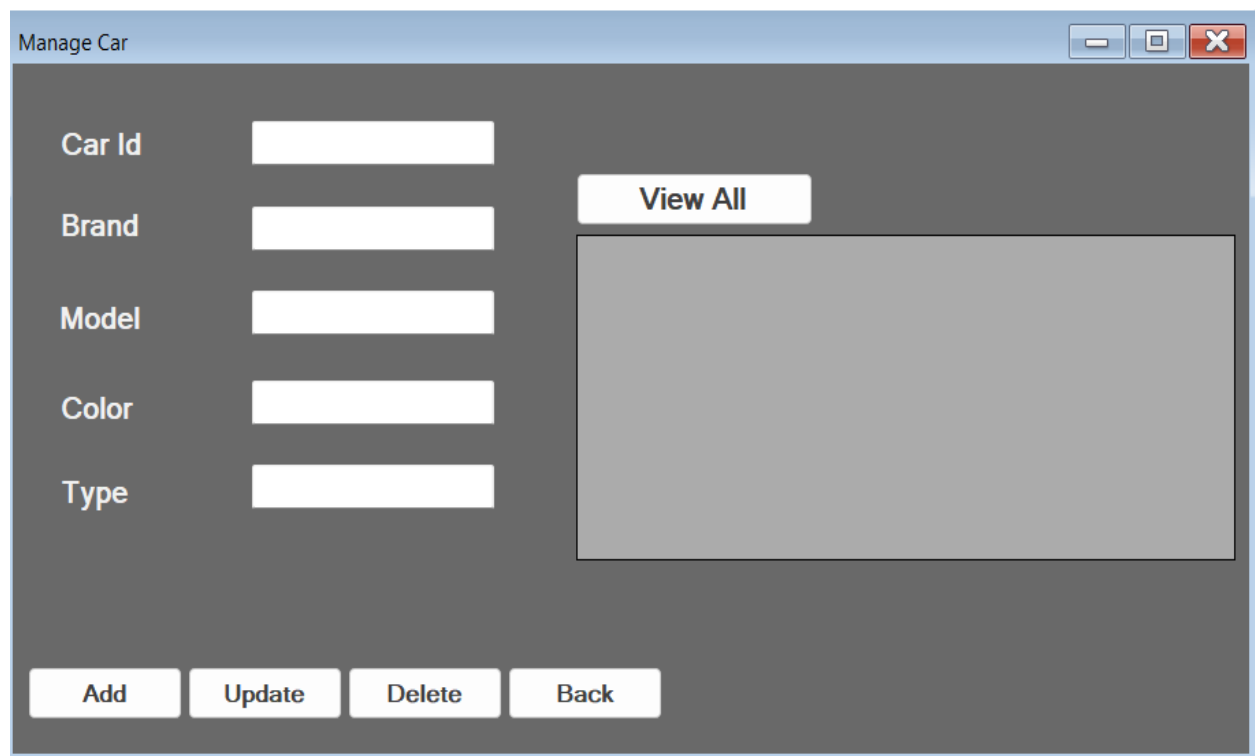
## Welcome Form



The wireframe shows a window titled "Welcome" with standard minimize, maximize, and close buttons. The main content area has a dark gray background. At the top center, the word "Welcome" is displayed in white. Below it, there are four white buttons with black text: "customer", "booking", "Return", and "Vehicle", arranged horizontally. At the bottom center, there is a single white button with black text labeled "Logout".

Figure 8 : Welcome - Wire Frame

## Manage Car Form

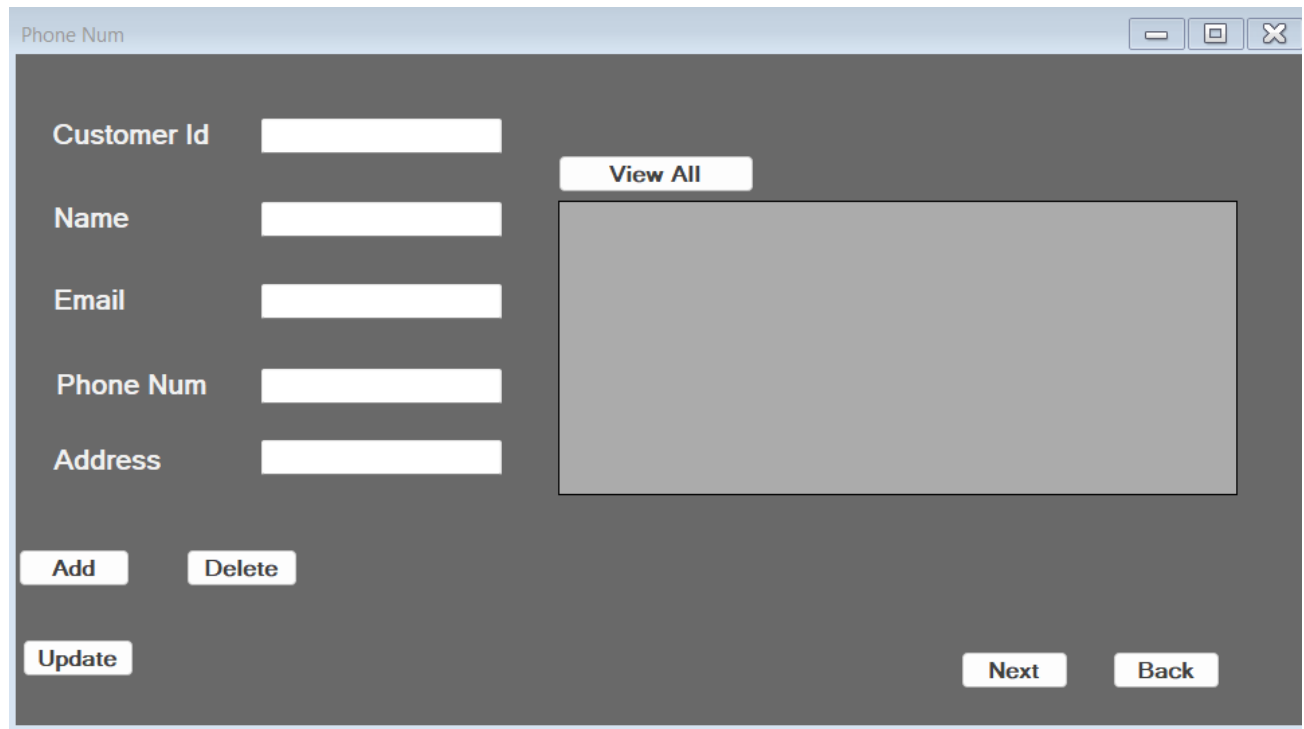


The wireframe shows a window titled "Manage Car" with standard minimize, maximize, and close buttons. The main content area has a dark gray background. On the left side, there are five labels: "Car Id", "Brand", "Model", "Color", and "Type", each followed by a white text input field. To the right of these fields is a white button with black text labeled "View All". Below the input fields and the "View All" button is a large, empty gray rectangular area, likely a placeholder for a table or list of cars. At the bottom of the window, there are four white buttons with black text: "Add", "Update", "Delete", and "Back", arranged horizontally.

Figure 9 : Manage Car - Wire Frame



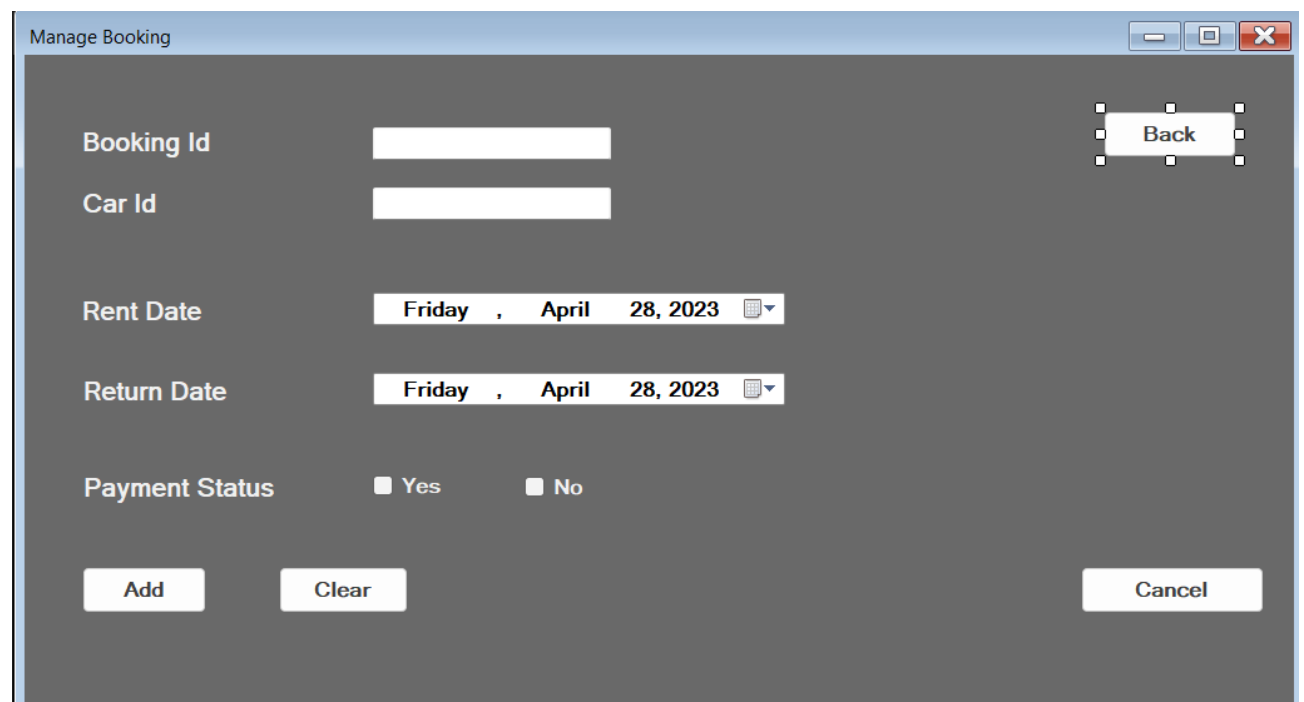
## Manage Customer Form



The wireframe shows a window titled "Phone Num" with standard minimize, maximize, and close buttons. On the left, there are five input fields labeled "Customer Id", "Name", "Email", "Phone Num", and "Address". To the right of these fields is a "View All" button. Below the input fields are three buttons: "Add", "Delete", and "Update". At the bottom right of the window are "Next" and "Back" buttons. A large gray rectangular area is positioned to the right of the "View All" button, likely representing a data table or list.

Figure 10 : Manage Customer - Wire Frame

## Manage Booking Form



The wireframe shows a window titled "Manage Booking" with standard minimize, maximize, and close buttons. It contains several input fields: "Booking Id", "Car Id", "Rent Date", and "Return Date". The date fields are pre-filled with "Friday , April 28, 2023" and include a calendar icon. Below these is a "Payment Status" section with "Yes" and "No" radio buttons. At the bottom, there are three buttons: "Add", "Clear", and "Cancel". A "Back" button is also present in the top right corner of the form area.

Figure 11 : Manage Booking - Wire Frame

Manage Return Form

Return Page

Back

Booking id

Return

Return Date

Friday . April ▾

Figure 12 : Manage Return - Wire Frame

## Chapter 04 – DEVELOPMENT

This is the actual writing of the program. A small project can be written by a single developer, while a large project can be broken up and worked on by several teams. Use an access control or source code management application at this stage. These systems help developers track changes in code. They also help ensure compatibility between different team projects and ensure achievement of target goals.

The coding process involves many other tasks. Most developers need skills or working in a team. Finding and correcting bugs and errors is very important. Tasks such as waiting for test results or compiling code so that an application can run often hold up the development process. SDLC can anticipate these delays and assign other duties to developers.

In our project we used Visual Studio 2022, SQL Server Management Studio 2019 and C# programming language on development stage.

### Visual Studio

Visual Studio is a powerful developer tool that we can use to complete the entire development cycle in one place. It is a comprehensive integrated development environment (IDE) that we can use to write, edit, debug, and build code, and then publish our app. Beyond code editing and debugging, Visual Studio includes compilers, code completion tools, source control, extensions, and many more features to enhance every stage of the software development process.

### SQL Server Management Studio

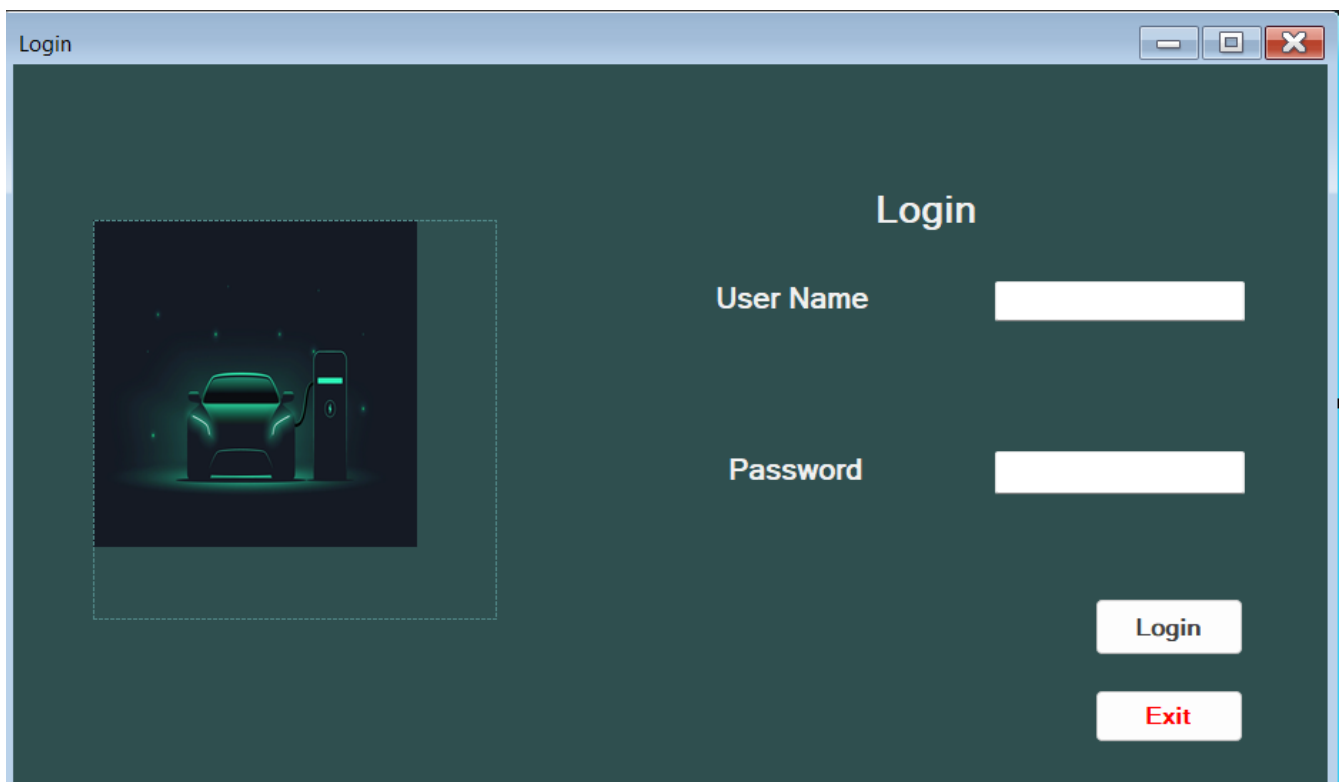
SQL Server Management Studio (SSMS) is an integrated environment for managing any SQL infrastructure. Use SSMS to access, configure, manage, administer, and develop all components of SQL Server, Azure SQL Database, Azure SQL Managed Instance. SSMS provides a single comprehensive utility that combines a broad group of graphical tools with many rich script editors to provide access to SQL Server for developers and database administrators of all skill levels.

## C#

C# (pronounced "See Sharp") is a modern, object-oriented, and type-safe programming language. C# enables developers to build many types of secure and robust applications that run in .NET. C# has its roots in the C family of languages and will be immediately familiar to C, C++, Java, and JavaScript programmers. This tour provides an overview of the major components of the language in C# 11 and earlier. If you want to explore the language through interactive examples, try the introduction to C# tutorials.

## Codes of Our Project

### Login Form



The image shows a screenshot of a Windows-style application window titled "Login". The window has a dark blue background. On the left side, there is a square image of a futuristic car with glowing green lights, parked next to a charging station. On the right side, the word "Login" is displayed in white. Below it, there are two input fields: "User Name" and "Password". At the bottom right, there are two buttons: "Login" and "Exit". The "Exit" button has red text.

Figure 13 : Login Form

```

1 namespace Vehicel_rent_system
2 {
3     partial class flogin
4     {
5         /// <summary>
6         /// Required designer variable.
7         /// </summary>
8         private System.ComponentModel.IContainer components = null;
9
10        /// <summary>
11        /// Clean up any resources being used.
12        /// </summary>
13        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
14        protected override void Dispose(bool disposing)
15        {
16            if (disposing && (components != null))
17            {
18                components.Dispose();
19            }
20            base.Dispose(disposing);
21        }
22
23        Windows Form Designer generated code
24
25        private System.Windows.Forms.Label label1;
26        private System.Windows.Forms.Label label2;
27        private System.Windows.Forms.TextBox txtuname;
28        private System.Windows.Forms.Button btnlogin;
29        private System.Windows.Forms.TextBox txtpwd;
30        private System.Windows.Forms.Label label3;
31        private System.Windows.Forms.PictureBox pictureBox1;
32        private System.Windows.Forms.Button btnExit;
33    }
34 }

```

Figure 14 : Login Form Design

## LOGIN BUTTON

```

1 reference
private void button1_Click(object sender, EventArgs e)
{
    if (txtuname.Text == "A" && txtpwd.Text == "Ad")
    {
        welcome welcome = new welcome();
        welcome.Show();

        this.Hide();
        MessageBox.Show("Succesfully Login", "New", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else if (txtuname.Text == "Fofficer" && txtpwd.Text == "F123")
    {
        welcome welcome = new welcome();
        welcome.Show();

        this.Hide();
        MessageBox.Show("Succesfully Login", "New", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    else
    {
        MessageBox.Show("Incorrect username or password", "Alert", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

Figure 15 : Login Button Coding

## EXIST BUTTON

```
1 reference  
private void btnExit_Click(object sender, EventArgs e)  
{  
    Application.Exit();  
}
```

Figure 16 : Exit Button Coding

---

## Welcome Form

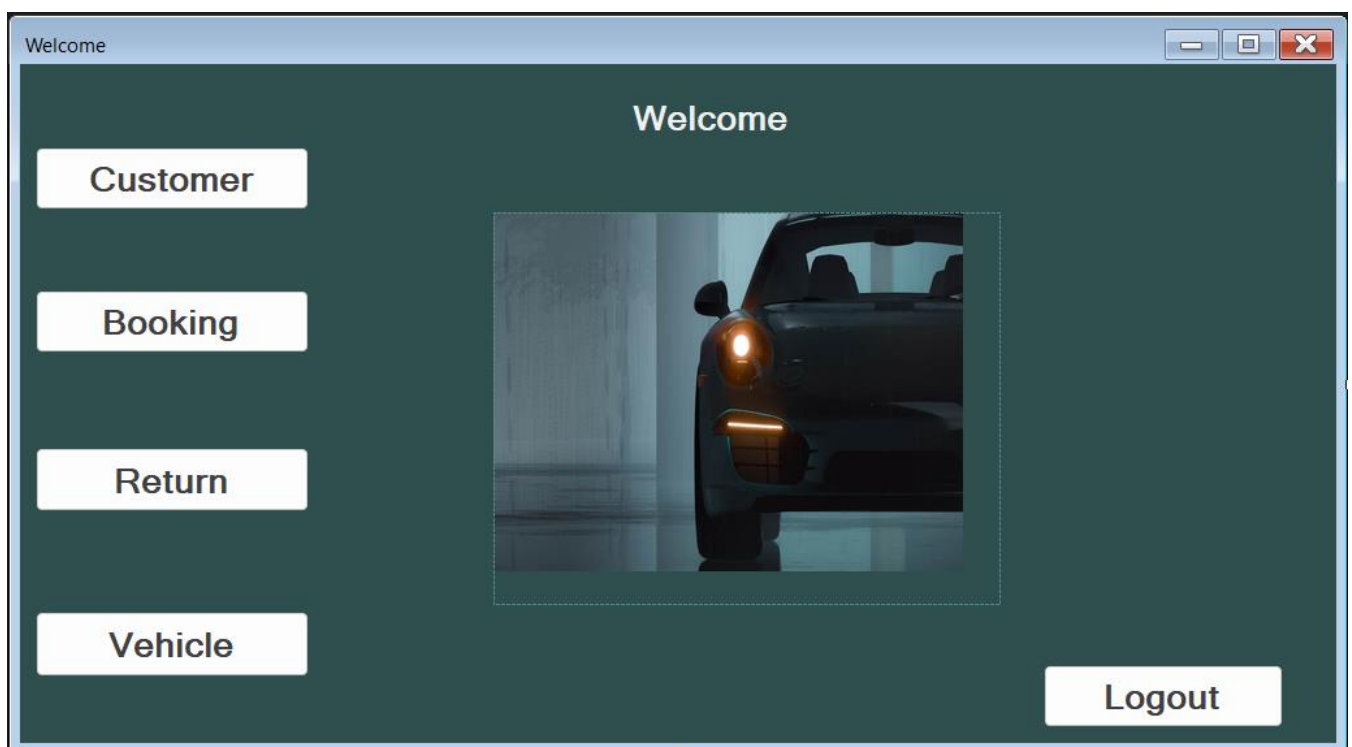


Figure 17 : Welcome Form

```

1 namespace venice_rent_system
2 {
3     17 references
4     partial class welcome
5     {
6         /// <summary>
7         /// Required designer variable.
8         /// </summary>
9         private System.ComponentModel.IContainer components = null;
10
11         /// <summary>
12         /// Clean up any resources being used.
13         /// </summary>
14         /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
15         0 references
16         protected override void Dispose(bool disposing)
17         {
18             if (disposing && (components != null))
19             {
20                 components.Dispose();
21             }
22             base.Dispose(disposing);
23         }
24
25         Windows Form Designer generated code
26
27         private System.Windows.Forms.Button btncustomer;
28         private System.Windows.Forms.Label label1;
29         private System.Windows.Forms.PictureBox pictureBox1;
30         private System.Windows.Forms.Button btnbooking;
31         private System.Windows.Forms.Button btnteturn;
32         private System.Windows.Forms.Button btnvehicle;
33         private System.Windows.Forms.Button btnlogout;
34         private System.Windows.Forms.Button button1;
35         private System.Windows.Forms.Label label2;
36         private System.Windows.Forms.Button button2;
37         private System.Windows.Forms.Button button3;
38         private System.Windows.Forms.Button button4;
39         private System.Windows.Forms.Button button5;
40         private System.Windows.Forms.PictureBox pictureBox2;
41     }
42 }

```

Figure 18 : Welcome Form Design

## CUTOMER BUTTON

```

1 reference
private void button1_Click_1(object sender, EventArgs e)
{
    customerpg customerpg = new customerpg();
    customerpg.Show();
    this.Hide();
}

```

Figure 19 : Customer Button Cording

## CAR BUTTON

```

1 reference
private void button4_Click(object sender, EventArgs e)
{
    managecar managecar = new managecar();
    managecar.Show();
    this.Hide();
}

```

Figure 20 : Car Button Coding

## BOOKING BUTTON

```
1 reference
private void button3_Click_1(object sender, EventArgs e)
{
    bookingPg bookingPg = new bookingPg();
    bookingPg.Show();
    this.Hide();
}
```

Figure 21 : Booking Button Coding

## RETURN BUTTON

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    returns returns = new returns();
    returns.Show();
    this.Hide();
}
```

Figure 22 : Return Button Coding

## LOGOUT BUTTON

```
1 reference
private void button5_Click(object sender, EventArgs e)
{
    MessageBox.Show("Are you sure !", "Alert", MessageBoxButtons.OK, MessageBoxIcon.Warning);

    flogin flogin = new flogin();
    flogin.Show();
    this.Hide();

    this.Hide();
}
```

Figure 23 : Logout Button Coding



# Customer Form

Customer Details

Customer Id

Name

Email

Phone Num

Address

	CustomerId	C_Name	Email	PhoneNum
*				

AddDeleteBackView All

UpdateNext

Figure 24 : Customer Form

```

1 namespace Vehicel_rent_system
2 {
3     4 references
4     partial class customerpg
5     {
6         /// <summary>
7         /// Required designer variable.
8         /// </summary>
9         private System.ComponentModel.IContainer components = null;
10
11         /// <summary>
12         /// Clean up any resources being used.
13         /// </summary>
14         /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
15         0 references
16         protected override void Dispose(bool disposing)
17         {
18             if (disposing && (components != null))
19             {
20                 components.Dispose();
21             }
22             base.Dispose(disposing);
23         }
24
25         Windows Form Designer generated code
26
27         private System.Windows.Forms.Button btnAdd1;
28         private System.Windows.Forms.TextBox txtCid;
29         private System.Windows.Forms.Label label1;
30         private System.Windows.Forms.Label label2;
31         private System.Windows.Forms.Label label3;
32         private System.Windows.Forms.Label label4;
33         private System.Windows.Forms.Label label5;
34         private System.Windows.Forms.TextBox txtAdr;
35         private System.Windows.Forms.TextBox txtPN;
36         private System.Windows.Forms.TextBox txtEmail;
37         private System.Windows.Forms.TextBox txtName;
38         private System.Windows.Forms.Button btnBack;
39         private System.Windows.Forms.Button btnUpdate1;
40         private System.Windows.Forms.Button btnNext;
41         private System.Windows.Forms.Button btnDelete1;
42         private System.Windows.Forms.BindingSource customerDetailsBindingSource;
43         private Vehicle_SystemDataSet1 vehicle_SystemDataSet1;
44         private Vehicle_SystemDataSet3 vehicle_SystemDataSet3;
45         private System.Windows.Forms.BindingSource customerDetailsBindingSource1;
46         private Vehicle_SystemDataSet3TableAdapters.Customer_DetailsTableAdapter customer_DetailsTableAdapter;
47         private V_ManageDataSet2 v_ManageDataSet2;
48         private System.Windows.Forms.BindingSource cDetailsBindingSource;
49         private V_ManageDataSet2TableAdapters.C_DetailsTableAdapter c_DetailsTableAdapter;
50         private V_ManageDataSet3 v_ManageDataSet3;
51         private System.Windows.Forms.BindingSource cusDetailsBindingSource;
52         private V_ManageDataSet3TableAdapters.Cus_DetailsTableAdapter cus_DetailsTableAdapter;
53         private System.Windows.Forms.DataGridview dataGridView1;

```

Figure 25 : Customer Form Design

## ADD BUTTON

```

1 reference
private void btnAdd1_Click(object sender, EventArgs e)
{
    try
    {
        con.Open();
        SqlCommand cmd = new SqlCommand("INSERT INTO Cus_Details(CustomerId,C_Name,Email,PhoneNum,Address) VALUES(" + txtCid.Text + "','" + txtName.Text + "','" +
            txtEmail.Text + "','" + txtPN.Text + "','" + txtAdr.Text + "')", con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Done");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    con.Close();
}

```

Figure 26 : Add Button Coding

## DELETE BUTTON

```
1 reference
private void btnDelete1_Click(object sender, EventArgs e)
{
    try
    {
        con.Open();
        SqlCommand cmd = new SqlCommand("DELETE FROM Cus_Details WHERE CustomerId=" + dataGridView1.SelectedRows[0].Cells[0].Value, con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Deleted");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        con.Close();
    }
}
```

Figure 27 : Delete Button Coding

## UPDATE BUTTON

```
1 reference
private void btnUpdate1_Click(object sender, EventArgs e)
{
    try
    {
        con.Open();
        SqlCommand cmd = new SqlCommand("UPDATE Cus_Details SET C_Name='" + txtName.Text + "', Email='" + txtEmail.Text + "', PhoneNum='" + txtPN.Text + "', Adress='" + txtAdr.Text + "' WHERE CustomerId=" + t
        cmd.ExecuteNonQuery();
        MessageBox.Show("Updated");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    finally
    {
        con.Close();
    }
}
```

Figure 28 : Update Button

## NEXT BUTTON

```
1 reference
private void btnNext_Click(object sender, EventArgs e)
{
    bookingPg bookingPg = new bookingPg();
    bookingPg.Show();
    this.Hide();
}
```

Figure 29 : Next Button Coding

## BACK BUTTON

```
1 reference
private void btnBack_Click(object sender, EventArgs e)
{
    welcome welcome = new welcome();
    welcome.Show();
    this.Hide();
}
```

Figure 30 : Back Button coding

## VEIW ALL BUTTON

```

1 reference
private void btnView_Click(object sender, EventArgs e)
{
    string connectionString = ("Data Source=DESKTOP-EU1QSR0\\SQLEXPRESS;Initial Catalog=V_Manage;Integrated Security=True");
    string ViewCD = ("SELECT * FROM Cus_Details ");
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        try
        {
            SqlCommand cmd = new SqlCommand(ViewCD, con);
            SqlDataAdapter adapter = new SqlDataAdapter(cmd);
            DataTable dataTable = new DataTable();
            adapter.Fill(dataTable);

            dataGridView1.DataSource = dataTable;
        }
        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message);
        }
    }
}

```

Figure 31 : View All Button Coding

## Car Form

Manage Car

Car Id

Car Number

Brand

Model

Color

Type

View All

Add Delete

Update Back

VehicleId	VehicleNum	Brand	Model
*			

Figure 32 : Car Form

```

1 namespace Vehicel_rent_system
2 {
3     4 references
4     partial class managecar
5     {
6         /// <summary>
7         /// Required designer variable.
8         /// </summary>
9         private System.ComponentModel.IContainer components = null;
10
11         /// <summary>
12         /// Clean up any resources being used.
13         /// </summary>
14         /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
15         0 references
16         protected override void Dispose(bool disposing)
17         {
18             if (disposing && (components != null))
19             {
20                 components.Dispose();
21             }
22             base.Dispose(disposing);
23         }
24
25         Windows Form Designer generated code
26
27         403
28         private System.Windows.Forms.Button btnUpdate;
29         private System.Windows.Forms.Label label1;
30         private System.Windows.Forms.TextBox txtCarId;
31         private System.Windows.Forms.Label label2;
32         private System.Windows.Forms.Label label3;
33         private System.Windows.Forms.Label label4;
34         private System.Windows.Forms.Label label5;
35         private System.Windows.Forms.TextBox txtType;
36         private System.Windows.Forms.TextBox txtColor;
37         private System.Windows.Forms.TextBox txtModel;
38         private System.Windows.Forms.TextBox txtbrand;
39         private System.Windows.Forms.Button btnBack;
40         private System.Windows.Forms.Button btnDelete;
41         private System.Windows.Forms.Button btnAddCar;
42         private System.Windows.Forms.Button btnView;
43         private System.Windows.Forms.Label label6;
44         private System.Windows.Forms.TextBox txtCarNum;
45         private Vehicle_SystemDataSet vehicle_SystemDataSet;
46         private System.Windows.Forms.BindingSource managecarBindingSource;
47         private Vehicle_SystemDataSetTableAdapters.Manage_carTableAdapter manage_carTableAdapter;
48         private Vehicle_SystemDataSet1TableAdapters.Customer_DetailsTableAdapter customer_DetailsTableAdapter;
49         private Vehicle_SystemDataSet2 vehicle_SystemDataSet2;
50         private System.Windows.Forms.BindingSource managecarBindingSource1;
51         private Vehicle_SystemDataSet2TableAdapters.Manage_carTableAdapter manage_carTableAdapter1;
52         private System.Windows.Forms.DataGridview1 dataGridview1;
53         private V_ManageDataSet v_ManageDataSet;
54         private System.Windows.Forms.BindingSource vDetailsBindingSource;
55     }

```

Figure 33 : Car Form Design

## ADD BUTTON

```

1 reference
private void btnAddCar_Click(object sender, EventArgs e)
{
    try
    {
        con.Open();
        SqlCommand cmd = new SqlCommand("INSERT INTO V_Details(VehicleId,VehicleNum,Brand,Model,color,Type) VALUES(" + txtCarId.Text + "," + txtCarNum.Text + "," + txtbrand.Text + "," +
        "" + txtModel.Text + "," + txtColor.Text + "," + txtType.Text + ")", con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Done");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    con.Close();
}

```

Figure 34 : Add Button Coding

## UPDATE BUTTON

```

1 reference
private void btnUpdate_Click(object sender, EventArgs e)
{
    {
        try
        {
            con.Open();
            SqlCommand cmd = new SqlCommand("UPDATE V_Details SET VehicleNum='" + txtCarNum.Text + "', Brand='" + txtBrand.Text + "', Model='" + txtModel.Text + "', Color='" + txtColor.Text + "'" +
            ", Type='" + txtType.Text + "' WHERE VehicleId=" + txtCarId.Text, con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Updated");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            con.Close();
        }
    }
}

```

Figure 35 : Update Button Coding

## Booking Form

Manage Booking

Booking Id

Car Id

Rent Date

Return Date

Payment Status

**Available Vehicles**

	VehicleId	VehicleNum	Brand	Model
*				

**Booked Vehicles**

	BookinId	CarId	RentDate	ReturnDate
*				

Add Update Back View All Delete

Figure 36 : Booking Form

```

1 namespace Veniter_rent_System
2 {
3     6 references
4     partial class bookingPg
5     {
6         /// <summary>
7         /// Required designer variable.
8         /// </summary>
9         private System.ComponentModel.IContainer components = null;
10
11         /// <summary>
12         /// Clean up any resources being used.
13         /// </summary>
14         /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
15         0 references
16         protected override void Dispose(bool disposing)
17         {
18             if (disposing && (components != null))
19             {
20                 components.Dispose();
21             }
22             base.Dispose(disposing);
23         }
24
25         Windows Form Designer generated code
26
27         private System.Windows.Forms.Button button1;
28         private System.Windows.Forms.TextBox txtBooking;
29         private System.Windows.Forms.Label label1;
30         private System.Windows.Forms.DateTimePicker dateP1;
31         private System.Windows.Forms.Label label2;
32         private System.Windows.Forms.Label label3;
33         private System.Windows.Forms.Label label4;
34         private System.Windows.Forms.Label label5;
35         private System.Windows.Forms.Button button2;
36         private System.Windows.Forms.Button button3;
37         private System.Windows.Forms.DateTimePicker dateP2;
38         private System.Windows.Forms.TextBox txtcid;
39         private System.Windows.Forms.DataGridview dataGridview1;
40         private System.Windows.Forms.DataGridview dataGridview2;
41         private System.Windows.Forms.Label label6;
42         private System.Windows.Forms.Label label7;
43         private V_ManageDataSet1 v_ManageDataSet1;
44         private System.Windows.Forms.BindingSource bDetailsBindingSource;
45         private V_ManageDataSet1TableAdapters.B_DetailsTableAdapter b_DetailsTableAdapter;
46         private System.Windows.Forms.DataGridviewTextBoxColumn bookinIdDataGridviewTextBoxColumn;
47         private System.Windows.Forms.DataGridviewTextBoxColumn carIdDataGridviewTextBoxColumn;
48         private System.Windows.Forms.DataGridviewTextBoxColumn rentDateDataGridviewTextBoxColumn;
49         private System.Windows.Forms.DataGridviewTextBoxColumn returnDateDataGridviewTextBoxColumn;
50         private System.Windows.Forms.DataGridviewTextBoxColumn paymentsDataGridviewTextBoxColumn;
51         private V_ManageDataSet5 v_ManageDataSet5;
52         private System.Windows.Forms.BindingSource vDetailsBindingSource;
53         private V_ManageDataSet5TableAdapters.V_DetailsTableAdapter v_DetailsTableAdapter;

```

Figure 37 : Booking Form Design

## ADD BUTTON

```

1 reference
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        con.Open();
        SqlCommand cmd = new SqlCommand("INSERT INTO B_Details(BookinId,CarId,RentDate,ReturnDate,Payment_S) VALUES(" + txtBooking.Text + "," + txtcid.Text + "," +
            "" + dateP1.Text + "," + dateP2.Text + "," + comb81.Text + ")", con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Done !", "-:-");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    con.Close();
}

```

Figure 38 : Add Button Coding

## UPDATE BUTTON

```

1 reference
private void btnUpdate_Click(object sender, EventArgs e)
{
    {
        try
        {
            con.Open();
            SqlCommand cmd = new SqlCommand("UPDATE B_Details SET CarId='" + txtcid.Text + "', RentDate='" + dateP1.Text + "', ReturnDate='" + dateP2.Text + "', Payment_S='" + combS1.Text +
            "' WHERE BookinId='" + txtBooking.Text, con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Updated");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            con.Close();
        }
    }
}

```

Figure 39 : Update Button Coding

## Return Form

Return

Return Data

Return Id

Bookin Id

Customer Id

Return Date 4/29/2023

	ReturnId	BookinId	CustomerId	R_Date
*				

Add Update Back Delete View All

Figure 40 : Return Form



```

1 namespace Vehicel_rent_system
2 {
3     4 references
4     partial class returns
5     {
6         /// <summary>
7         /// Required designer variable.
8         /// </summary>
9         private System.ComponentModel.IContainer components = null;
10
11         /// <summary>
12         /// Clean up any resources being used.
13         /// </summary>
14         /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
15         0 references
16         protected override void Dispose(bool disposing)
17         {
18             if (disposing && (components != null))
19             {
20                 components.Dispose();
21             }
22             base.Dispose(disposing);
23         }
24
25         Windows Form Designer generated code
26 private System.Windows.Forms.Button btnAdd;
27 private System.Windows.Forms.Label label1;
28 private System.Windows.Forms.Label label2;
29 private System.Windows.Forms.Label label3;
30 private System.Windows.Forms.TextBox txtBookingId;
31 private System.Windows.Forms.TextBox txtCustomerId;
32 private System.Windows.Forms.DataGridView dataGridView1;
33 private System.Windows.Forms.Label label4;
34 private System.Windows.Forms.Label label5;
35 private System.Windows.Forms.TextBox txtReturn;
36 private System.Windows.Forms.DateTimePicker dateP1;
37 private V_ManageDataSet6 v_ManageDataSet6;
38 private System.Windows.Forms.BindingSource rDetailsBindingSource;
39 private V_ManageDataSet6TableAdapters.R_DetailsTableAdapter r_DetailsTableAdapter;
40 private System.Windows.Forms.DataGridViewTextBoxColumn returnIdDataGridViewTextBoxColumn;
41 private System.Windows.Forms.DataGridViewTextBoxColumn bookinIdDataGridViewTextBoxColumn;
42 private System.Windows.Forms.DataGridViewTextBoxColumn customerIdDataGridViewTextBoxColumn;
43 private System.Windows.Forms.DataGridViewTextBoxColumn rDateDataGridViewTextBoxColumn;
44 private System.Windows.Forms.Button btnView;
45 private System.Windows.Forms.Button btnBack;
46 private System.Windows.Forms.Button btnDelete;
47 private System.Windows.Forms.Button btnUpdate;
48
49 }
50
51 }

```

Figure 41 : Return Form Design

## ADD BUTTON

```

1 reference
private void btnUpdate_Click(object sender, EventArgs e)
{
    try
    {
        con.Open();
        SqlCommand cmd = new SqlCommand("INSERT INTO R_Details(ReturnId,BookinId,CustomerId,R_Date) VALUES(" + txtReturn.Text + "," + txtBookingId.Text + "," + txtCustomerId.Text + "," +
            "" + dateP1.Text + "" )", con);
        cmd.ExecuteNonQuery();
        MessageBox.Show("Done");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    con.Close();
}

```

Figure 42 : Add Button Coding

## UPDATE BUTTON

```
1 reference
private void btnUpdate_Click_1(object sender, EventArgs e)
{
    {
        try
        {
            con.Open();
            SqlCommand cmd = new SqlCommand("UPDATE R_Details SET BookinId='" + txtBookingId.Text + "', CustomerId='" + txtCustomerId.Text + "', R_Date='" + dateP1.Text +
            "' WHERE ReturnId=" + txtrReturn.Text, con);
            cmd.ExecuteNonQuery();
            MessageBox.Show("Updated");
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
        finally
        {
            con.Close();
        }
    }
}
```

Figure 43 : Update Button Coding

## Chapter 05 – EVALUATION

### Types of Testing

Every software needs a testing part after they developed. It will help to confirm its functionalities with the users' reactions. There are some testing types;

---

#### Black Box Testing

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test.

---

#### White Box Testing

White box testing is a form of application testing that provides the tester with sophisticated knowledge of the application being tested, including access to source code and design documents. This deep visibility enables white-box testing to address identity issues invisible to gray- and black-box testing.

---

#### Unit Testing

Unit testing is a way of testing a unit -the smallest piece of code that can be logically isolated in a system.in most programming languages, that is a function, a subroutine a method or property. The isolated part of the definition is important.

---

#### System Testing

System testing also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application.

## User Acceptance Testing

User acceptance testing which is performed on most UIT projects, sometimes called beta testing or end user testing is a phase of software development in which the software is tested in the “real world “by the intended audience or business represented

Testers will use documentation created at the design phase, user personas and user journey scenarios to run as many test cases as possible in the attempt to uncover any bugs that need to be fixed before deployment.

## Test Code

### Login Form

no	Test case	Expected result	Test result
1	Enter a valid username and a valid password	Show a message box "login successful" and press ok Software should display work area interface	successful
2	Enter a invalid username and a valid password	Show a message box "username or password is incorrect" and press ok Software should not display work area interface	successful
3	Enter a valid username and a invalid password	Show a message box " username or password is incorrect " and press ok Software should not display work area interface	successful
4	Enter a invalid username and a invalid password	Show a message box "username or password is incorrect" and press ok Software should not display work area interface	successful

**Figure 44 : Login Form Testing**

## Chapter 06 – CONCLUSION

Finally, in our project, we introduce a low-cost system that is suitable and more useful for the problems that we have analyzed. The Customer form is connected to the Car form through the booking process. When a customer books a car, the front officer selects a specific car from the available inventory. The booking form would include information about the car, such as the car ID, model, and engine type.

The Car form is connected to the Booking form through the inventory management process. When a customer books a car, the car is marked as reserved in the inventory management system. The Booking form would include information about the car, such as the rental period and the rental rate. The Booking form is connected to the Return form through the rental process. When a customer returns a car, the rental period is recorded in the Booking form, along with any additional charges or fees. But that additional charges or fees are not including the system.

The Return form would include information about the condition of the car when it was returned, any damage or issues that need to be addressed, and the final charges for the rental. The Customer form is connected to both the Booking and Return forms through the rental process. When a customer makes a booking, their personal information is recorded in the Customer form. When they return the car, their personal information may be required again in the Return form to complete the rental transaction.

In our project, the Gantt Chart is as follow.

# Gantt Chart

## Gantt Chart - Vehicle Rental System

Select a period to highlight at right. A legend describing the charting follows.

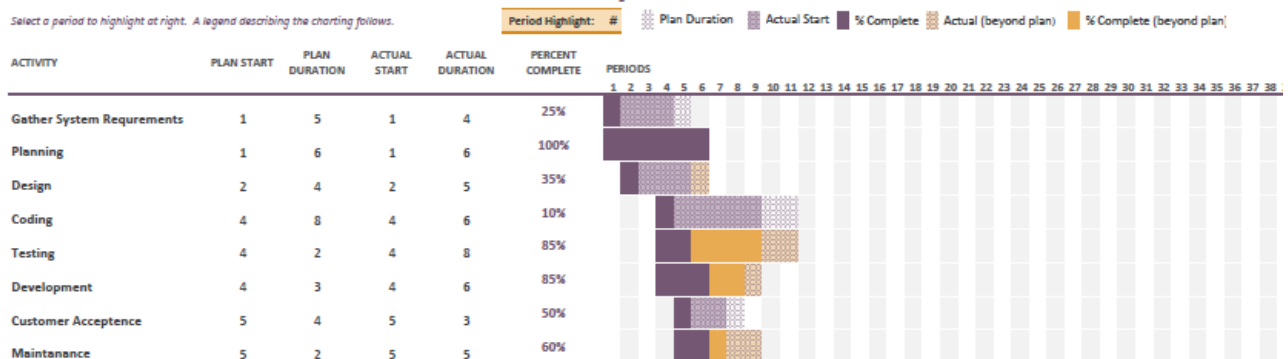


Figure 45 : Gantt Chart

We could not get our final system as we planned the system in the design stage using wire frame. Although it was possible to design the interface better than the wire frame, the functionality of the system could not be obtained in the same way. There were various reasons for that. But finally, as a group, we think we have found the best solution to the problem.

## REFERENCES

W3Schools is a popular online platform that offers tutorials, references, and examples on web development technologies such as HTML, CSS, JavaScript, and more. The platform is constantly updated to reflect the latest industry standards and best practices. Also, we used to google platform. Google is a multinational technology company that provides a wide range of services and products, including search engines, advertising, cloud computing, software, and hardware.

We referenced You Tube for our coding and making our system interface. YouTube is the largest video-sharing platform globally, with over two billion monthly active users. YouTube offers various educational channels and content creators that provide tutorials and references on a wide range of topics, including programming, web development, and software engineering.

These resources are available for free, and users can interact with content creators through comments and social media platforms.