A

Report

On

SOFTWARE ENGINEERING LABORATORY

(CS-713)

CAR SHOWROOM MANAGEMENT SYSTEM

Submitted by

Bikram Tripathi, Roll No.

21RISTCSE004 for

The Partial Fulfillment of 7th Semester Examination

(Session: August-December, 2024)

Department of Computer Science and Engineering
Regional Institute of Science and Technology
9th Mile, Meghalaya

Affiliated to
North-Eastern Hill University
Shillong, Meghalaya

Regional Institute of Science and Technology

Department of Computer Science and Engineering

Certificate of Approval

This is to certify that Bikram Tripathi, Roll No. 21RISTCSE004 has successfully completed the lab report for Software Engineering Laboratory prescribed by NEHU for the partial fulfillment of B.Tech in CSE for the 7th semester for the academic session August-December, 2024.

Signature
HOD
(Department of
Computer Science and Engineering)

Signature
Faculty In-Charge
(Department of
Computer Science and Engineering)

TABLE OF CONTENTS

CHAPTER 1

A. Introduction

- **Overview of the Car Showroom Management System**: Begin by explaining what a car showroom management system entails. Highlight its purpose in helping showrooms manage daily operations, such as customer registration, inventory management, and sales tracking.
- **Technology and Tools Used**: Mention that you are using Visual Basic for the programming component and MySQL as the database management system. Explain why these technologies were chosen, perhaps for their compatibility, ease of use, or robustness.
- **Importance of Automation in Showrooms**: Discuss how digital solutions can reduce errors, save time, and enhance customer satisfaction. Emphasize that the system can automate routine tasks, ensure data consistency, and improve the decision-making process by providing real-time data access.

B. Problem Statement

- Managing the operations of a car showroom manually is complex and time-consuming.
- Key aspects such as car inventory maintenance, customer order processing, and employee record management are prone to inefficiencies.
- Generating sales reports and other critical information becomes cumbersome and error-prone.
- Manual data handling increases the chances of errors and delays in processing.
- Inefficient resource management negatively impacts customer satisfaction and showroom productivity.
- Communication with manufacturers is not streamlined, leading to potential delays in car procurement.
- Tracking and managing employee performance is challenging without a centralized system.
- The absence of an integrated solution hinders the ability to make informed decisions and respond quickly to operational demands.

C. Objectives

A. **Automate Inventory Management**
   Develop a system to add, update, and track the availability of cars in the inventory.
B. **Streamline Customer Order Processing**
   Simplify customer registration, order placement, and billing processes to minimize errors and improve efficiency.
C. **Centralize Employee Management**
   Provide a platform for maintaining employee details, monitoring performance, and managing their roles and activities effectively.
D. **Enhance Reporting Capabilities**
   Enable the generation of detailed sales, inventory, and employee activity reports for better decision-making.
E. **Improve Manufacturer Collaboration**
   Create a module to manage manufacturer information and facilitate seamless communication for car procurement.
F. **Boost Customer Satisfaction**
   Reduce delays in service delivery and ensure a smooth customer experience by automating critical processes.
G. **Provide Secure Access**
   Implement a secure login system for employees and managers to ensure data confidentiality and restricted access.
H. **Ensure Scalability**
   Design the system to accommodate future enhancements, such as online car bookings and advanced analytics.

CHAPTER 2
A. **SDLC Model**

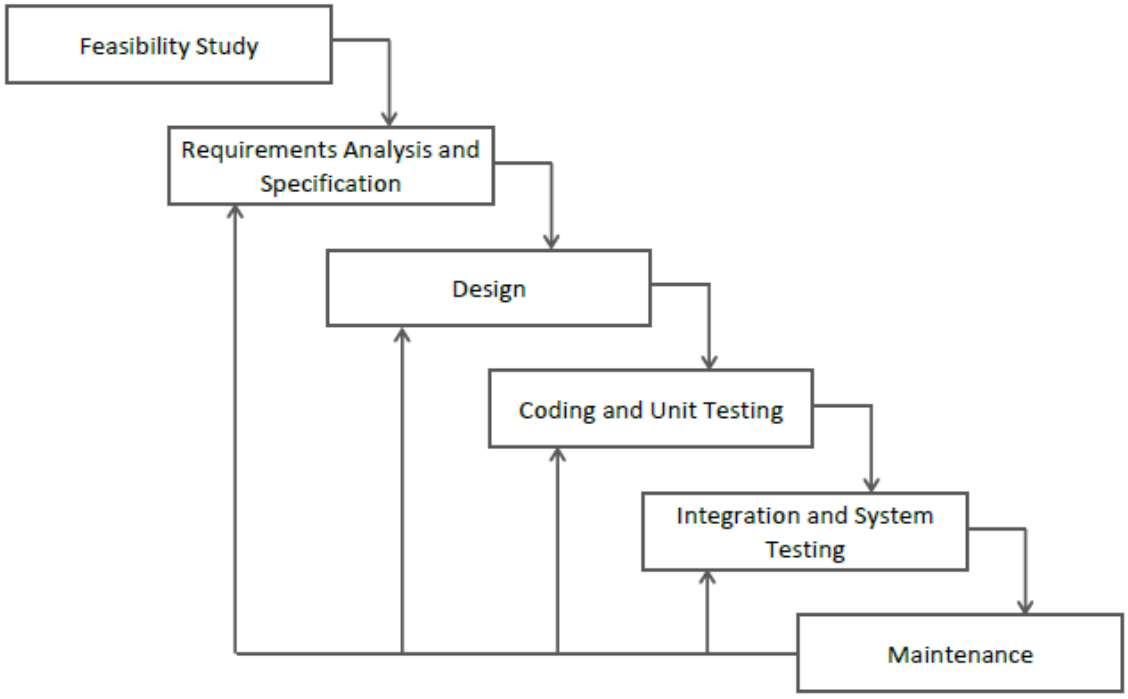


Figure 1: Iterative Waterfall Model

The **Car Showroom Management System** is a comprehensive application designed to streamline the operations of a car showroom. It enables efficient management of car inventory, customer orders, employee information, and billing. The system aims to automate and simplify complex processes like adding new cars, tracking inventory status, processing sales, and generating reports. This ensures improved productivity and customer satisfaction. The project follows the **Iterative Waterfall Model**, a software development methodology that progresses systematically through predefined phases while allowing feedback and iteration between steps. This approach ensures that each phase is carefully executed and refined before moving to the next.

Steps in the Iterative Waterfall Model for the Project

1. Feasibility Study

- **Objective**: Determine the project's practicality and assess its technical, economic, and operational feasibility.
- **Activities**:
  - Conducted an analysis of the existing manual processes in the showroom.
  - Assessed the available resources, including hardware, software, and workforce.
  - Estimated the costs of development and maintenance against potential benefits like reduced errors and improved efficiency.
- **Output**: A feasibility report confirming that the project is viable and beneficial to the showroom.

2. Requirement Analysis and Specification

- **Objective**: Gather and document the functional and non-functional requirements of the system.
- **Activities**:
  - Engaged with stakeholders (e.g., showroom managers, employees) to understand their needs.
  - Identified key requirements like managing car inventory, processing customer orders, and generating invoices.
  - Documented requirements in a clear and detailed Software Requirements Specification (SRS).
- **Output**: The SRS document outlining all system requirements.

3. System Design

- **Objective**: Create a blueprint for the system that fulfills the specified requirements.
- **Activities**:
  - Designed the database schema, including tables for cars, customers, orders, employees, and manufacturers.
  - Developed diagrams such as Data Flow Diagrams (DFDs) and Entity-Relationship Diagrams (ERDs) to represent system processes and data relationships.
  - Defined the user interface design for functionalities like adding cars, processing orders, and billing.
- **Output**: Design documents, including database structure, process workflows, and UI mockups.

4. Coding and Unit Testing

- **Objective**: Develop the system's modules and ensure their correctness individually.
- **Activities**:
  - Wrote code for core functionalities using Visual Basic for the frontend and MySQL for the backend.
  - Developed individual modules like "Add Car," "Process Customer Order," and "Generate Bill."
  - Conducted unit testing to verify the correctness of each module.
- **Output**: Functional and tested software modules.

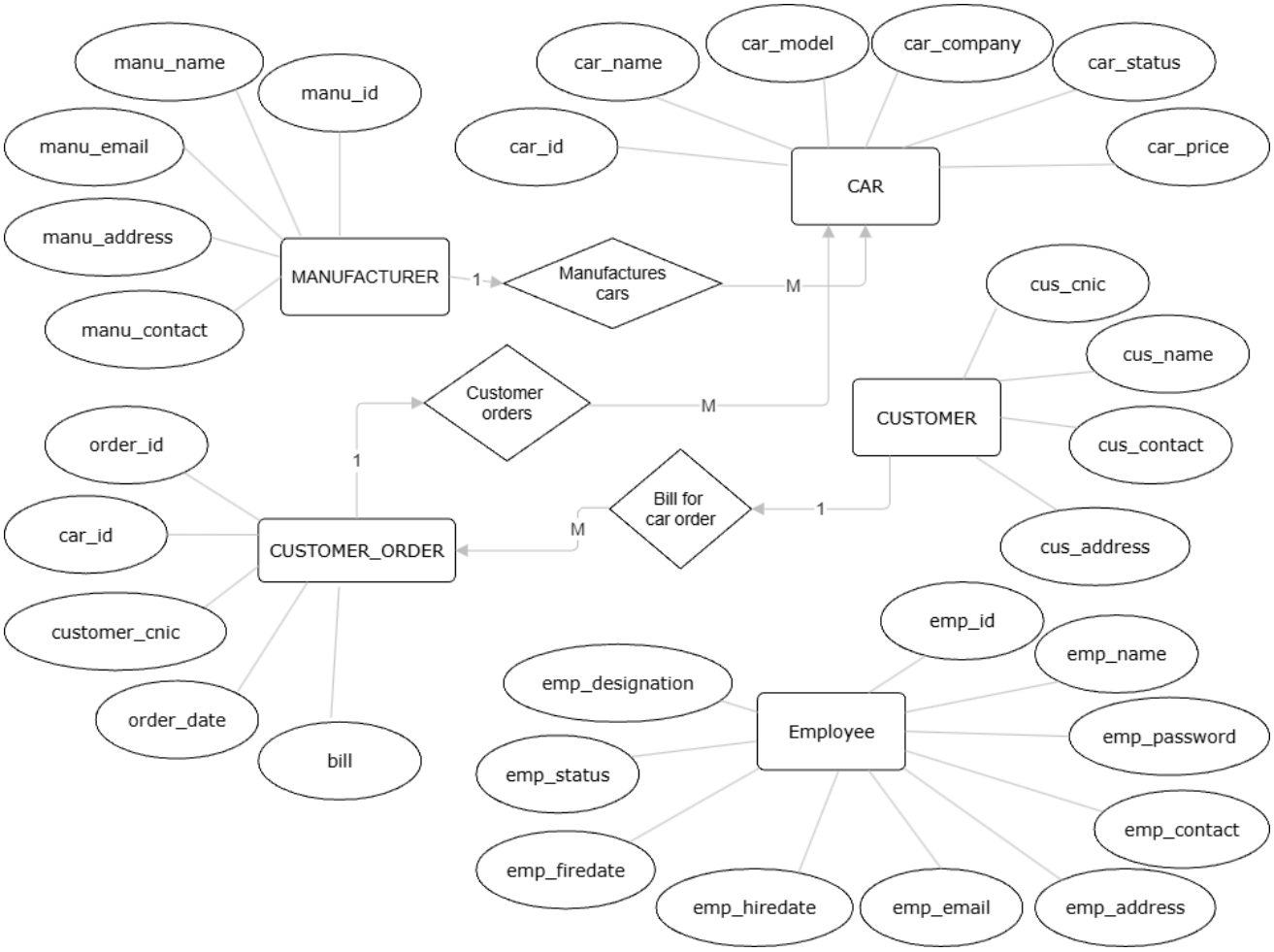
5. Integration and System Testing

- **Objective**: Combine all modules and test the system as a whole.
- **Activities**:
  - Integrated modules like inventory management, customer order processing, and billing into a single system.
  - Performed system testing to identify bugs and ensure the system worked as intended.
  - Validated the system against the SRS to confirm it met all specified requirements.
- **Output**: A fully integrated and tested Car Showroom Management System.

6. Maintenance

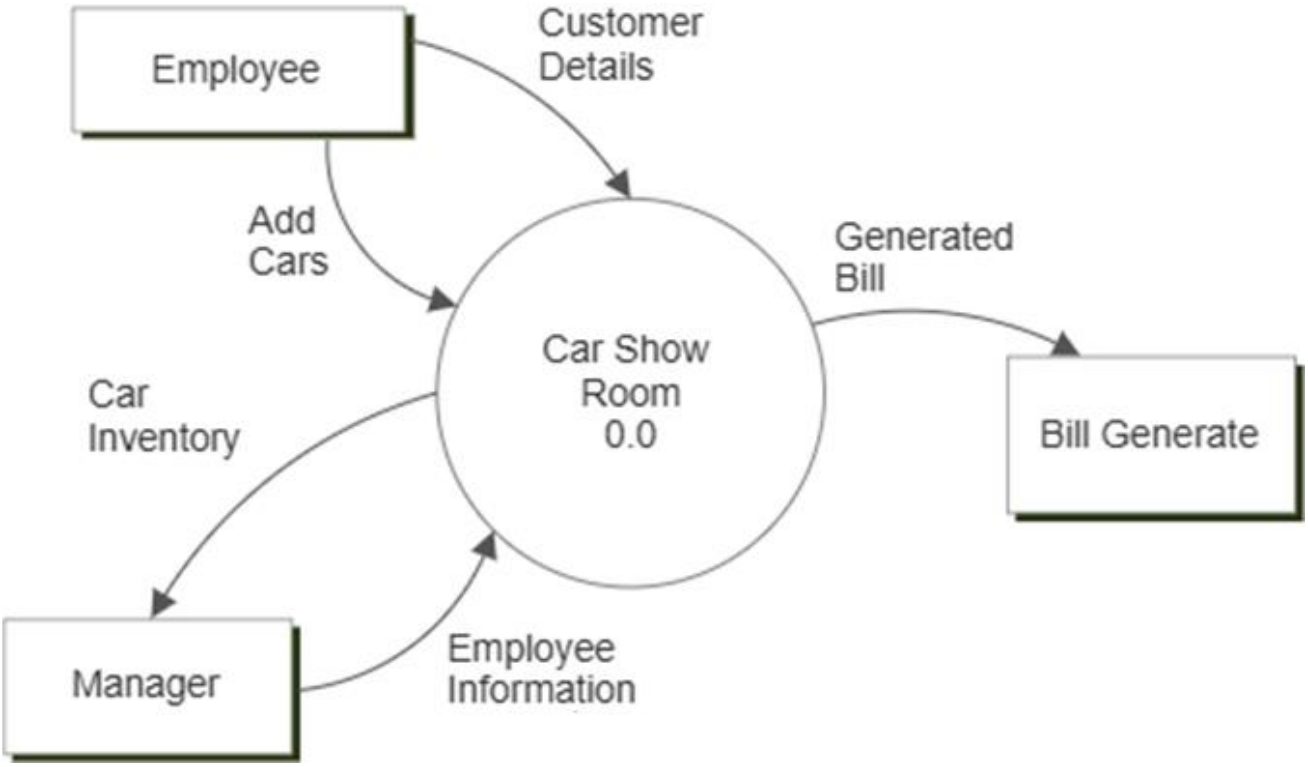
- **Objective**: Ensure the system remains operational and evolves to meet new requirements.
- **Activities**:
  - Addressed bugs and issues reported by end-users.
  - Added new features, such as advanced sales analytics and enhanced inventory reporting.
  - Upgraded the system for compatibility with newer technologies when needed.
- **Output**: An updated and robust system that adapts to the changing needs of the showroom.

B. **E-R Diagram**

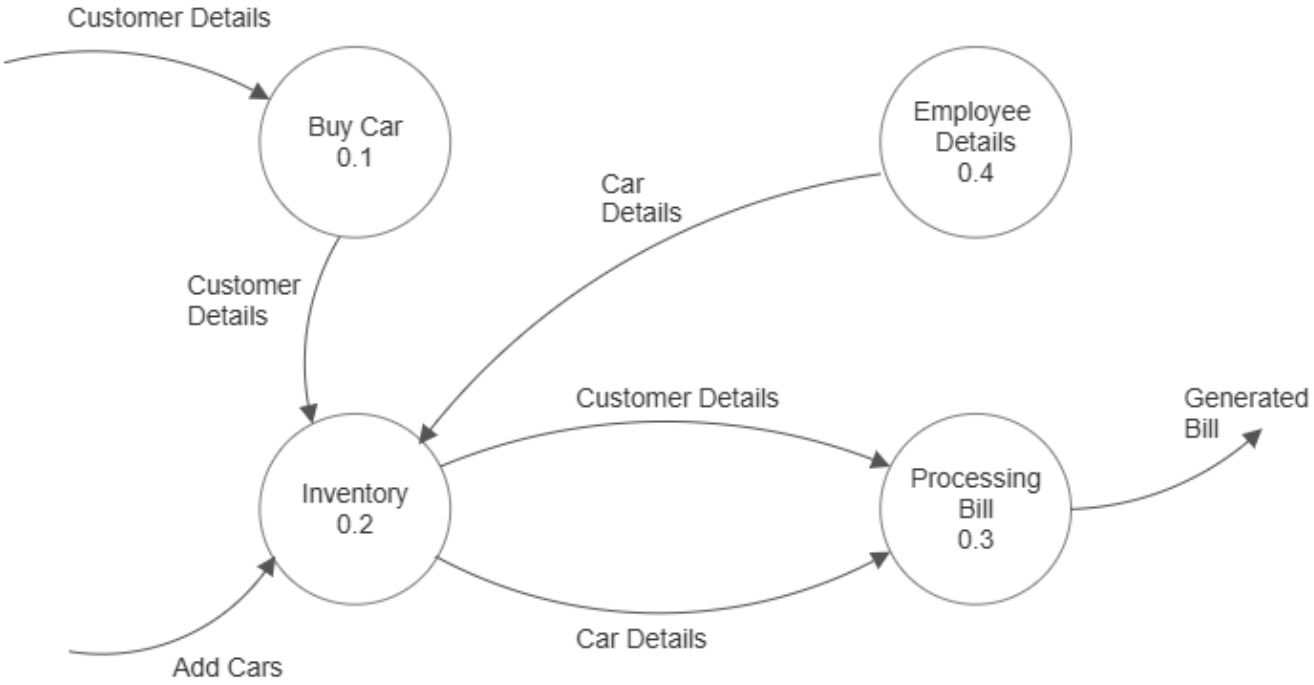

manu_name
manu_id
manu_email
manu_address
manu_contact
MANUFACTURER
1
Manufactures cars
M
car_name
car_model
car_company
car_status
car_id
car_price
CAR
cus_cnic
cus_name
CUSTOMER
cus_contact
cus_address
Customer orders
M
1
order_id
car_id
CUSTOMER_ORDER
customer_cnic
order_date
bill
M
Bill for car order
1
emp_id
emp_name
emp_designation
Employee
emp_password
emp_status
emp_contact
emp_firedate
emp_hiredate
emp_email
emp_address

C. **Data Flow Diagram Model**

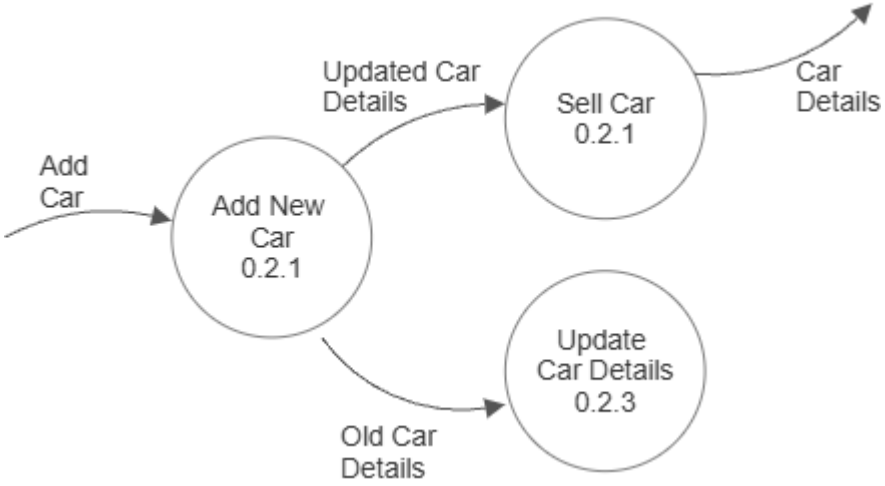
C.1 DFD Level 0



C.2 DFD Level 1



C.3 DFD Level 2



D. **Data Dictionary**

- **Customer Details**

  customer-id: integer
  customer-name: string
  customer-contact: string
  customer-address: string
  customer-data: {customer-id, customer-name, customer-contact, customer-address}
  input: Customer provides registration details.
  output: Stored customer information in the system.

- **Add Cars**

  car-id: integer
  car-name: string
  car-model: string
  car-price: float
  car-status: string
  add-car-data: {car-id, car-name, car-model, car-price, car-status}
  input: Car details to be added to the inventory.
  output: A new car record added to the system.

Car Inventory

  car-id: integer
  car-name: string
  car-model: string
  car-price: float
  car-status: string
  car-inventory-data: {car-id, car-name, car-model, car-price, car-status}
  input: Car records from the "Add Cars" process or updates from the "Update Car Details" process.
  output: A list of cars with their respective details available in the inventory.

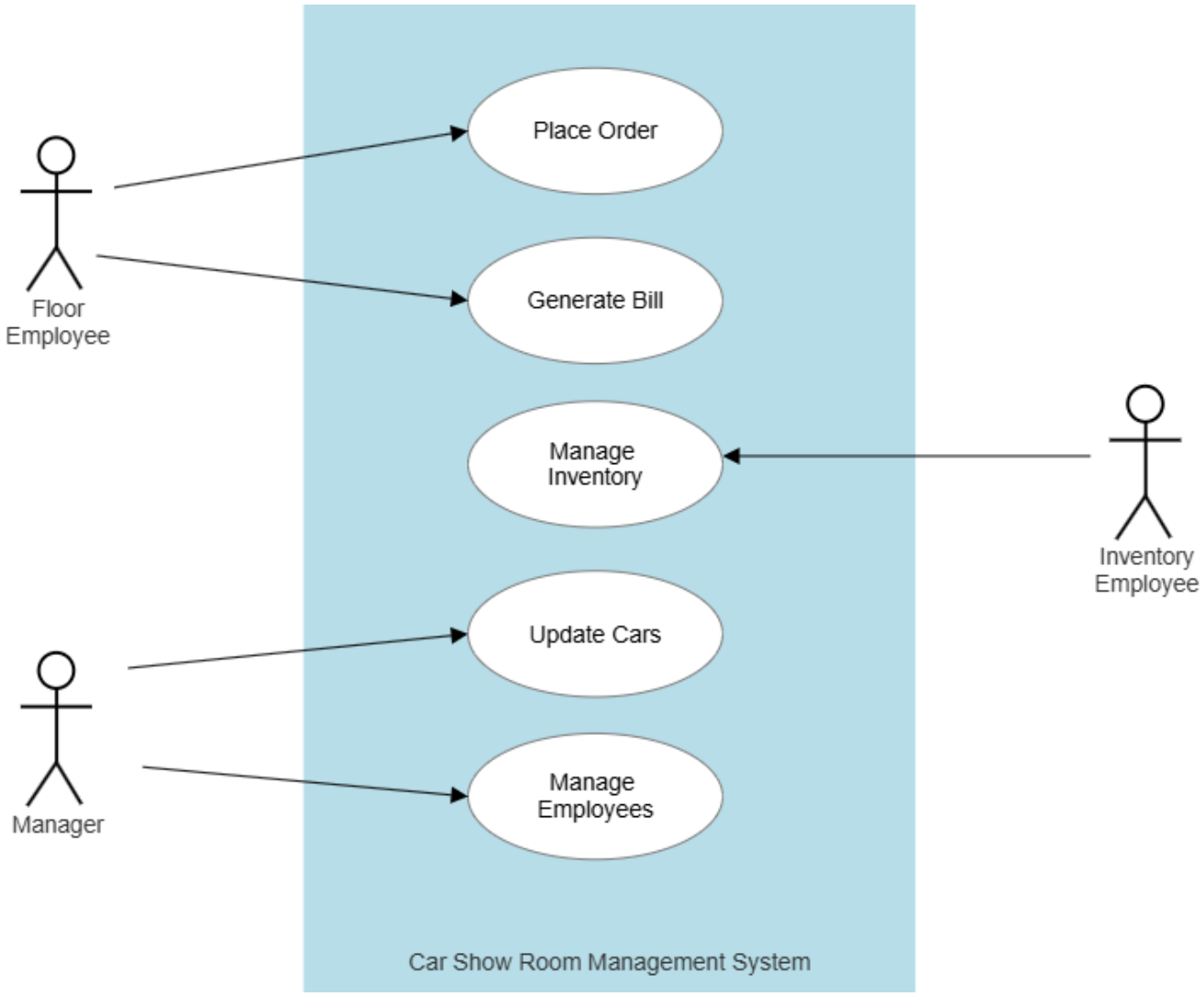
- **Employee Information**

  employee-id: integer
  employee-name: string
  employee-contact: string
  employee-role: string
  employee-status: string
  employee-info: {employee-id, employee-name, employee-contact, employee-role, employee-status}
  input: Employee registration details provided by HR or management.
  output: Employee records are stored in the system.

- **Generated Bill**

  bill-id: integer
  order-id: integer
  car-id: integer
  customer-id: integer
  bill-amount: float
  bill-date: date
  bill-details: {bill-id, order-id, car-id, customer-id, bill-amount, bill-date}
  input: Order details and car details.
  output: A generated bill for the customer for their car purchase.

E. **Use Case Diagram**



F. **Use Case Description**

1. Place Order

- **Actor**: Customer
- **System**: Car Show Room
- **Description**: Customer places an order by selecting a car and providing necessary details.
- **Preconditions**: Customer is registered in the system.
- **Basic Flow**:
    1. Customer logs in to the system.
    2. Customer selects a car and provides order details.
    3. System confirms order details and processes them.
- **Postconditions**: Order is recorded, and an invoice is generated.

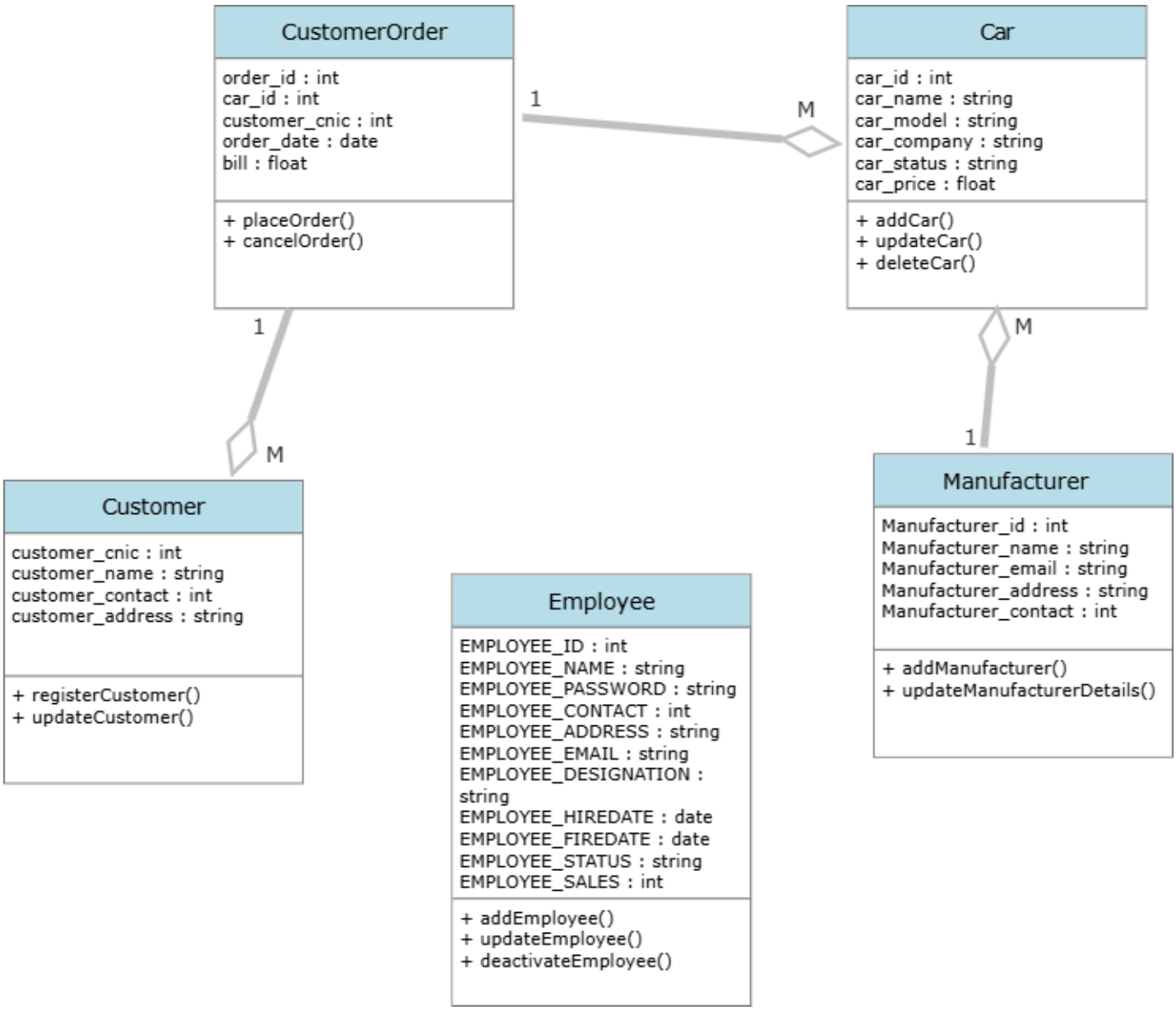
2. Manage Inventory

- **Actor**: Inventory Employee
- **System**: Car Show Room
- **Description**: The Inventory Employee manages the inventory by adding new cars, selling cars, and updating car details.
- **Preconditions**: Employee has inventory access permissions.
- **Basic Flow**:
    1. Inventory Employee accesses the inventory module.
    2. Employee performs actions (add, update, or sell cars).
    3. System updates the inventory database.
- **Postconditions**: Inventory reflects the latest updates.

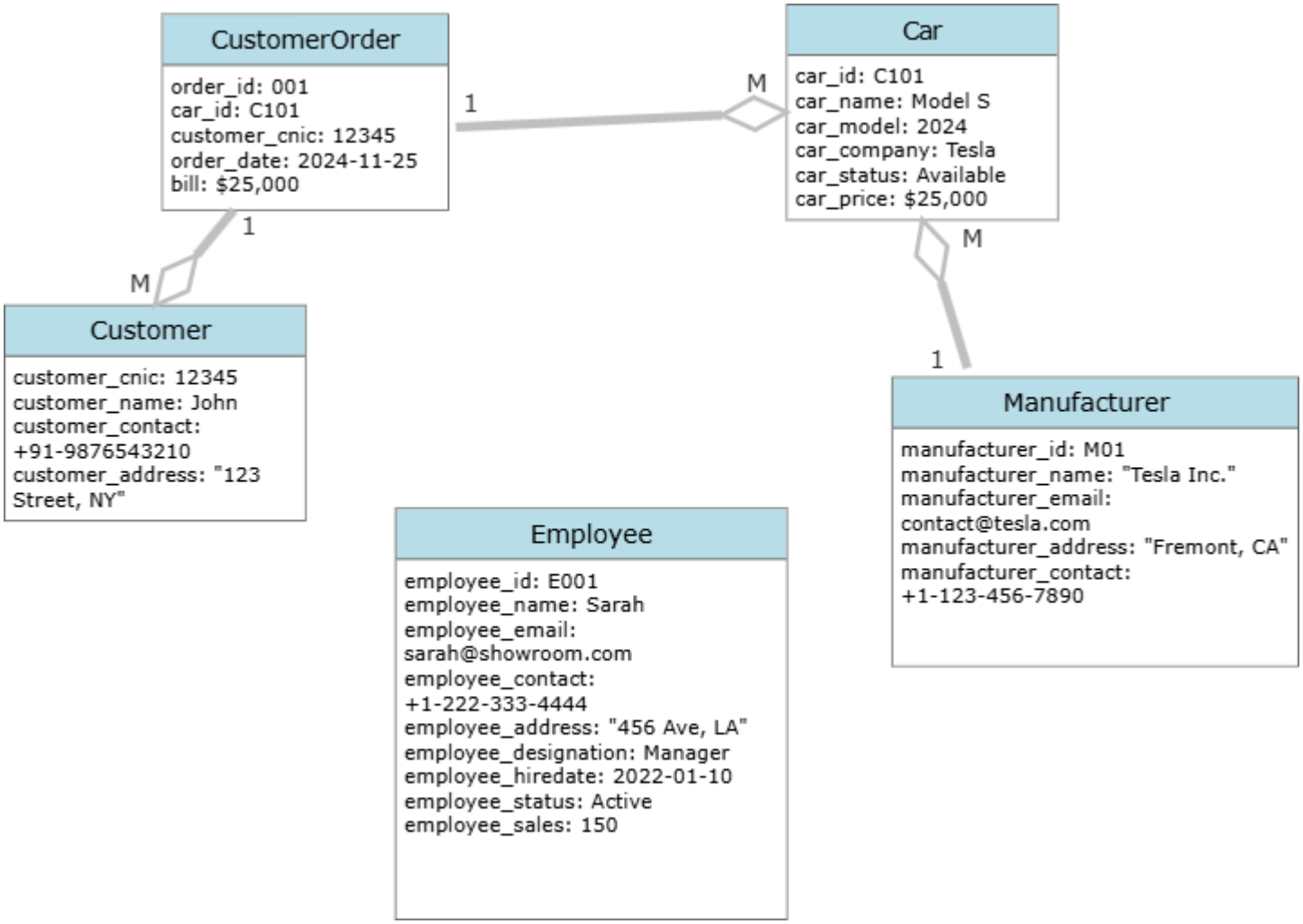
3. Manage Employees

- **Actor**: Manager
- **System**: Car Show Room
- **Description**: The Manager handles employee records, including adding, updating, and deactivating employees.
- **Preconditions**: Manager has administrative access.
- **Basic Flow**:
    1. Manager selects the "Employee Management" module.
    2. Manager performs actions (add, update, deactivate employees).
    3. System saves changes in the employee database.
- **Postconditions**: Employee records are updated.

G. **Class Diagram**

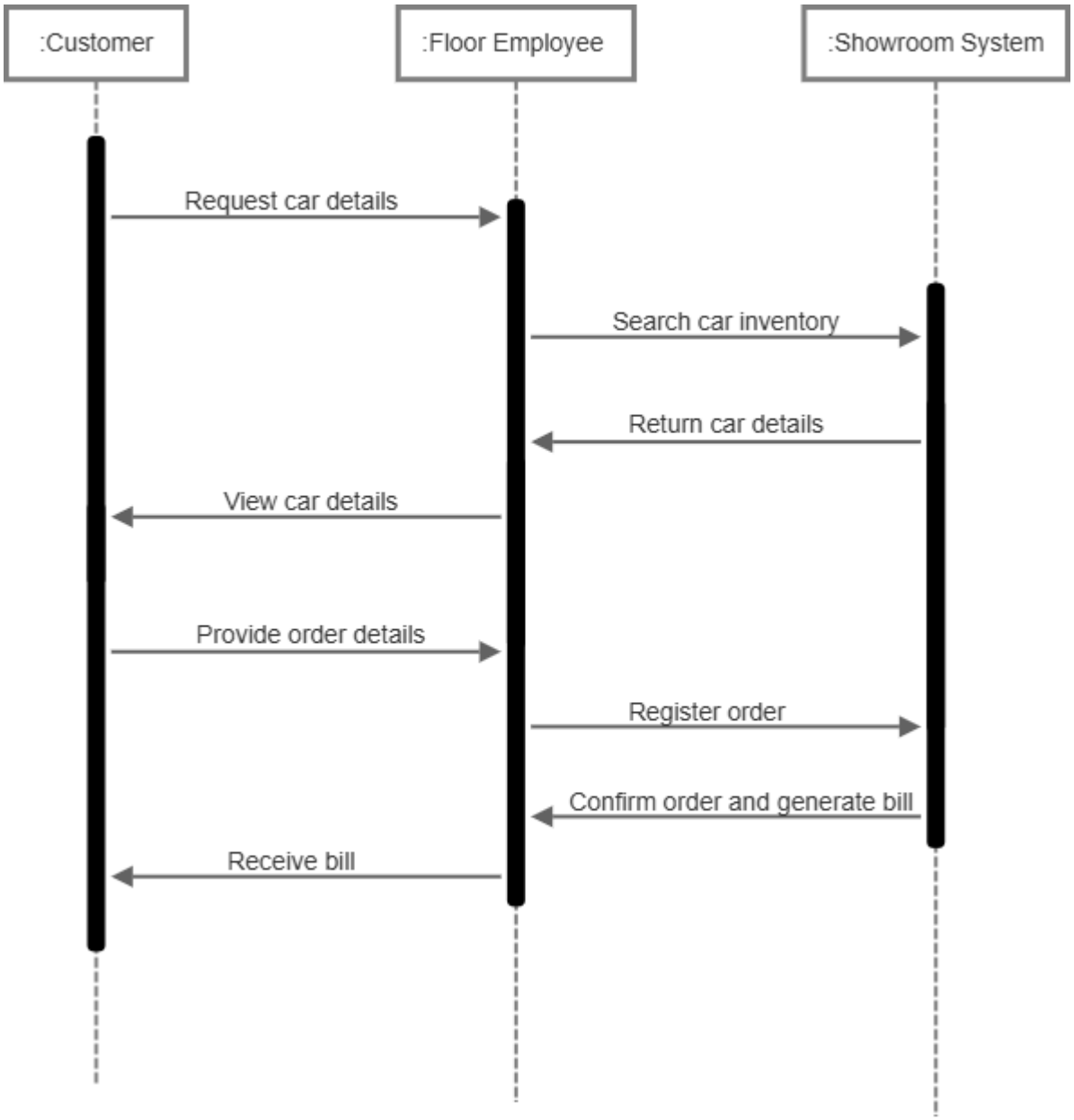


H. **Object Diagram**



I. **Sequence Diagram**

Sequence Diagram for “Buy Car” Event



CHAPTER 3

A. Implementation

Form1

```
Imports MySql.Data.MySqlClient

Public Class Form1
' Connection string for your MySQL database
Dim connectionString As String =
"server=localhost;userid=root;password=";database=CSM"
Dim connection As New MySqlConnection(connectionString)

' Event handler for login button click
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
' Get the username (email) and password entered by the user
Dim username As String = TextBox1.Text
Dim password As String = TextBox2.Text

' Check if either field is empty
If String.IsNullOrWhiteSpace(username) OrElse String.IsNullOrWhiteSpace(password) Then
MessageBox.Show("Please enter both username and password.", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
Return
End If

' SQL query to check if the user exists in the EMPLOYEE table with the entered username
and password
Dim query As String = "SELECT employee_id, employee_designation FROM EMPLOYEE
WHERE EMPLOYEE_EMAIL = @username AND EMPLOYEE_PASSWORD =
@password"
Dim command As New MySqlCommand(query, connection)
command.Parameters.AddWithValue("@username", username)
command.Parameters.AddWithValue("@password", password)

Try
connection.Open()
Dim reader As MySqlDataReader = command.ExecuteReader()

If reader.Read() Then
' Get the employee ID and designation
```

```
Dim employeeID As String = reader("employee_id").ToString()
Dim userDesignation As String = reader("employee_designation").ToString()

If userDesignation = "Employee" Then
' Redirect to Form4 (passing employeeID)
Dim form4 As New Form4(employeeID)
Form2.Show()
Me.Hide() ' Hide current form
ElseIf userDesignation = "Manager" Then
' Redirect to Form3 (manager view)
Dim form3 As New Form3()
form3.Show()
Me.Hide() ' Hide current form
Else
' Invalid role
MessageBox.Show("Incorrect username or password.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
End If
Else
' User not found, incorrect username/password
MessageBox.Show("Incorrect username or password.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
End If

Catch ex As Exception
MessageBox.Show("Error: " & ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
Finally
connection.Close()
End Try
End Sub

Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
End Sub

Private Sub PictureBox1_Click(sender As Object, e As EventArgs) Handles
PictureBox1.Click
TextBox1.Clear()
```

```
End Sub

Private Sub PictureBox4_Click(sender As Object, e As EventArgs) Handles
PictureBox4.Click
TextBox2.Clear()

End Sub

Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
Me.Close()
End Sub
End Class
```

Form2

```
Imports MySql.Data.MySqlClient

Public Class Form2
' Connection string for the MySQL database
Dim connectionString As String =
"server=localhost;userid=root;password=";database=CSM"
Dim connection As New MySqlConnection(connectionString)

' Event handler for Panel2 (Sell Car) click
Private Sub Panel2_Click(sender As Object, e As EventArgs) Handles Panel2.Click
Dim form4 As New Form4()
form4.Show()
Me.Hide()     ' Hide current form
End Sub

' Event handler for Panel3 (Buy Car) click


' Event handler for Panel4 (Sold Cars) click
Private Sub Panel4_Click(sender As Object, e As EventArgs) Handles Panel4.Click
' Query to retrieve sold car records
Dim query As String = "
SELECT
CUSTOMER_ORDER.ORDER_ID AS 'Order ID',
```

```
CAR.CAR_NAME AS 'Car Name',
CAR.CAR_MODEL AS 'Model',
CUSTOMER.CUSTOMER_NAME AS 'Customer Name',
CUSTOMER_ORDER.BILL AS 'Sold Price',
CUSTOMER_ORDER.ORDER_DATE AS 'Order Date'
FROM CUSTOMER_ORDER
INNER JOIN CAR ON CUSTOMER_ORDER.CAR_ID = CAR.CAR_ID
INNER JOIN CUSTOMER ON CUSTOMER_ORDER.CUSTOMER_CNIC =
CUSTOMER.CUSTOMER_CNIC"
DisplayRecords(query) ' Display the records in DataGridView1
End Sub

' Event handler for Panel5 (Available Cars) click
Private Sub Panel5_Click(sender As Object, e As EventArgs) Handles Panel5.Click
' Query to retrieve available car records
Dim query As String = "
SELECT
CAR_ID AS 'Car ID',
CAR_NAME AS 'Car Name',
CAR_MODEL AS 'Model',
CAR_COMPANY AS 'Company',
CAR_PRICE AS 'Price',
CAR_STATUS AS 'Status'
FROM CAR
WHERE CAR_STATUS = 'Available'"
DisplayRecords(query) ' Display the records in DataGridView1
End Sub

' Function to display records in DataGridView1
Private Sub DisplayRecords(query As String)
Try
Dim adapter As New MySqlDataAdapter(query, connection)
Dim dataTable As New DataTable()
adapter.Fill(dataTable)  ' Fill the DataTable with query results
DataGridView1.DataSource = dataTable  ' Bind the DataTable to DataGridView1
Catch ex As Exception
MessageBox.Show("Error: " & ex.Message, "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
```

```
End Try
End Sub

' Optional: Form closing event to close the database connection
Private Sub Form2_FormClosing(sender As Object, e As FormClosingEventArgs) Handles Me.FormClosing
connection.Close()
End Sub

Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
Form1.Show()
Me.Close()
End Sub

Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load

End Sub

Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
Form1.Close()
End Sub

Private Sub DataGridView1_CellContentClick(sender As Object, e As DataGridViewCellEventArgs) Handles DataGridView1.CellContentClick

End Sub

Private Sub Panel2_Paint(sender As Object, e As PaintEventArgs) Handles Panel2.Paint

End Sub

Private Sub Panel3_Click(sender As Object, e As EventArgs) Handles Panel3.Click
Form6.Show()
Me.Close()

End Sub
End Class
```

Form9

```
Imports System.Drawing.Printing
Imports MySql.Data.MySqlClient

Public Class Form9
' Variables to hold passed data
Private carId As String
Private customerName As String
Private customerCNIC As String
Private customerContact As String
Private customerAddress As String
Private employeeId As String
Private employeeName As String
Private employeeContact As String
Private employeeDesignation As String
Private orderId As Integer
Private carName As String
Private carModel As String
Private carCompany As String
Private carPrice As Decimal
Private taxAmount As Decimal = 1500 ' Example tax amount
Private billAmount As Decimal
Private orderDate As DateTime

' Custom Constructor
Public Sub New(carId As String, customerName As String, customerCNIC As String,
customerContact As String, customerAddress As String, employeeId As String,
employeeName As String, employeeContact As String, employeeDesignation As String,
orderId As Integer, carName As String, carModel As String, carCompany As String, carPrice
As Decimal, orderDate As Date)
' Initialize the form
InitializeComponent()

' Assign data to local variables
Me.carId = carId
Me.customerName = customerName
Me.customerCNIC = customerCNIC
Me.customerContact = customerContact
Me.customerAddress = customerAddress
```

```
Me.employeeId = employeeId
Me.employeeName = employeeName
Me.employeeContact = employeeContact
Me.employeeDesignation = employeeDesignation
Me.orderId = orderId
Me.carName = carName
Me.carModel = carModel
Me.carCompany = carCompany
Me.carPrice = carPrice
Me.orderDate = orderDate

' Calculate total bill
billAmount = carPrice + taxAmount
End Sub

' Fetch Employee Details from the Database
Private Sub FetchEmployeeDetails(empId As String)
Try
' Connection string to your MySQL database
Dim connectionString As String =
"server=localhost;userid=root;password=yourpassword;database=yourdatabase"
Using conn As New MySqlConnection(connectionString)
conn.Open()

' Query to get employee details
Dim query As String = "SELECT Employee_name, Employee_contact,
Employee_designation FROM Employee WHERE Employee_ID = @empId"
Using cmd As New MySqlCommand(query, conn)
cmd.Parameters.AddWithValue("@empId", empId)

' Execute the query and fetch the data
Using reader As MySqlDataReader = cmd.ExecuteReader()
If reader.Read() Then
employeeName = reader("Employee_name").ToString()
employeeContact = reader("Employee_contact").ToString()
employeeDesignation = reader("Employee_designation").ToString()
Else
MessageBox.Show("Employee not found!", "Error", MessageBoxButtons.OK,
```

```
MessageBoxIcon.Error)
End If
End Using
End Using
End Using
Catch ex As Exception
MessageBox.Show("An error occurred while fetching employee details: " & ex.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
End Try
End Sub


' Create PrintDocument object
Private WithEvents PrintDoc As New PrintDocument

' Create PrintPreviewDialog object
Private WithEvents PrintPreviewDialog As New PrintPreviewDialog

' Button to open Print Preview
Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
' Set the PrintDocument for the PrintPreviewDialog
PrintPreviewDialog.Document = PrintDoc

' Display the PrintPreviewDialog
PrintPreviewDialog.ShowDialog()
End Sub

' PrintPage event for PrintDocument
Private Sub PrintDoc_PrintPage(sender As Object, e As PrintPageEventArgs) Handles
PrintDoc.PrintPage
' Set fonts and layout
Dim titleFont As New Font("Arial", 16, FontStyle.Bold)
Dim headerFont As New Font("Arial", 12, FontStyle.Bold)
Dim regularFont As New Font("Arial", 10)
Dim margin As Integer = 50
Dim currentY As Integer = margin
Dim lineSpacing As Integer = 25
```

```
' Draw a centered header
Dim centerX As Integer = e.PageBounds.Width / 2
Dim headerText As String = "CAR SHOWROOM BILL"
Dim headerSize As SizeF = e.Graphics.MeasureString(headerText, titleFont)
e.Graphics.DrawString(headerText, titleFont, Brushes.Black, centerX - headerSize.Width / 2, currentY)
currentY += lineSpacing + 10

' Draw a horizontal line
e.Graphics.DrawLine(Pens.Black, margin, currentY, e.PageBounds.Width - margin, currentY)
currentY += lineSpacing

' Section: Customer Details
e.Graphics.DrawString("CUSTOMER DETAILS", headerFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("Name: " & customerName, regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("CNIC: " & customerCNIC, regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("Contact: " & customerContact, regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("Address: " & customerAddress, regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing + 10

' Draw a horizontal line
e.Graphics.DrawLine(Pens.Black, margin, currentY, e.PageBounds.Width - margin, currentY)
currentY += lineSpacing

' Section: Car Details
e.Graphics.DrawString("CAR DETAILS", headerFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("Car Name: " & carName, regularFont, Brushes.Black, margin,
```

```
currentY)
currentY += lineSpacing
e.Graphics.DrawString("Model: " & carModel, regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("Company: " & carCompany, regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("Price: " & carPrice.ToString("C2"), regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing + 10

' Draw a horizontal line
e.Graphics.DrawLine(Pens.Black, margin, currentY, e.PageBounds.Width - margin, currentY)
currentY += lineSpacing

' Section: Billing Details
e.Graphics.DrawString("BILLING DETAILS", headerFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("Tax Amount: " & taxAmount.ToString("C2"), regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("Total Bill: " & billAmount.ToString("C2"), headerFont, Brushes.Black, margin, currentY)
currentY += lineSpacing + 10

' Draw a horizontal line
e.Graphics.DrawLine(Pens.Black, margin, currentY, e.PageBounds.Width - margin, currentY)
currentY += lineSpacing

' Section: Order Details
e.Graphics.DrawString("ORDER DETAILS", headerFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
e.Graphics.DrawString("Order ID: " & orderId.ToString(), regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
```

```
e.Graphics.DrawString("Order Date: " & orderDate.ToString("yyyy-MM-dd"), regularFont, Brushes.Black, margin, currentY)
currentY += lineSpacing
' Footer
currentY += 30
e.Graphics.DrawString("Thank you for choosing our showroom!", headerFont, Brushes.Black, margin, currentY)
End Sub
' Method to update label values
Private Sub UpdateLabels()
' Car Section
Label4.Text = carName
'Label5.Text = carId
Label7.Text = carModel
Label9.Text = carCompany

' Seller/Customer Section
Label35.Text = customerName
Label33.Text = customerCNIC
Label31.Text = customerContact
Label29.Text = customerAddress

' Order Section
Label26.Text = orderId.ToString()
Label24.Text = billAmount.ToString("C2") ' Format as currency
Label22.Text = orderDate.ToString("yyyy-MM-dd"
End Class
```

CHAPTER 4

A. Testing

1. Functional Testing

Verifies that system components, such as customer order processing, inventory management, and billing, function as expected.

2. Integration Testing

Checks the seamless interaction between modules, such as updating inventory after a car is sold or generating invoices for processed customer orders.

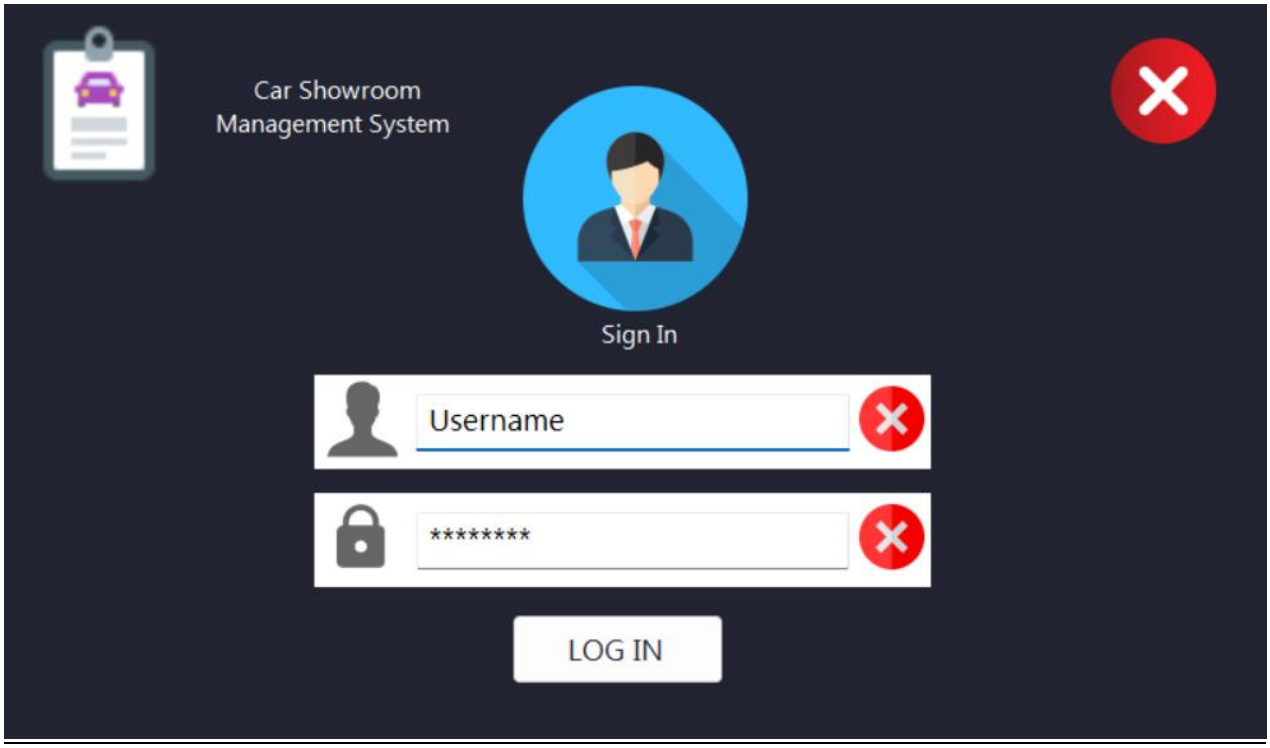
3. Black Box Testing

Black box testing focuses on evaluating the software's functionality without knowledge of the internal code or logic. Testers assess the system based on inputs and expected outputs.

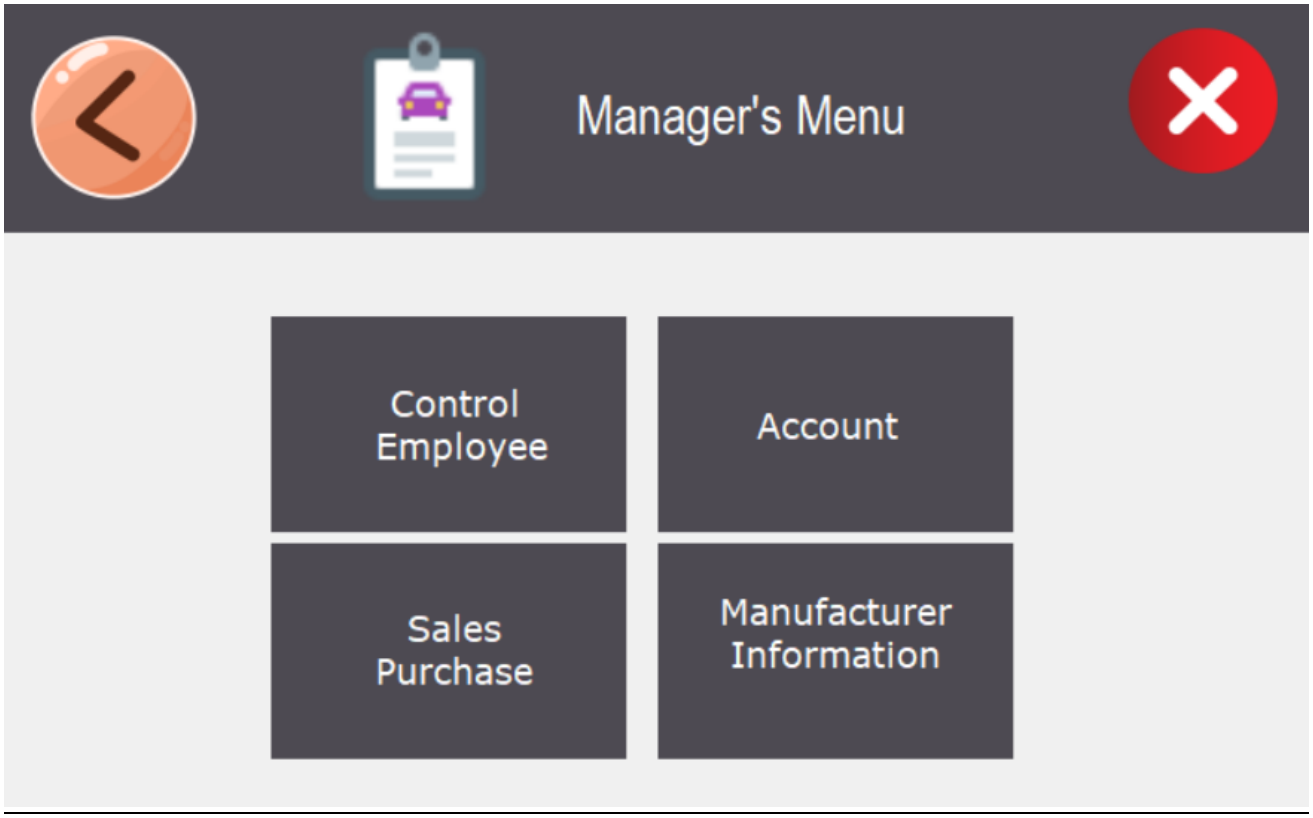
TestCaseID	Test Description	Input	Expected Output	Result
1	Employee login with valid inputs	Username: emp001, Password: pass123	Access to employee dashboard	Pass
2	Employee login with invalid inputs	Username: emp001, Password: wrongpass	"Invalid credentials" error message	Pass
3	Add a new car to inventory	Car Name: Model X, Car ID: C102	Car added and visible in inventory list	Pass
4	Add a new car to inventory	Car ID: C101, Customer ID: 12345	Order processed, invoice generated, and inventory updated	Pass
5	Update car details	Car ID: C101, Status: Sold	Car status updated in inventory	Pass
6	Generate sales report	Date Range: 01-01-2024 to 01-31-2024	Sales report generated with accurate details	Pass

B. Snapshots of Interfaces

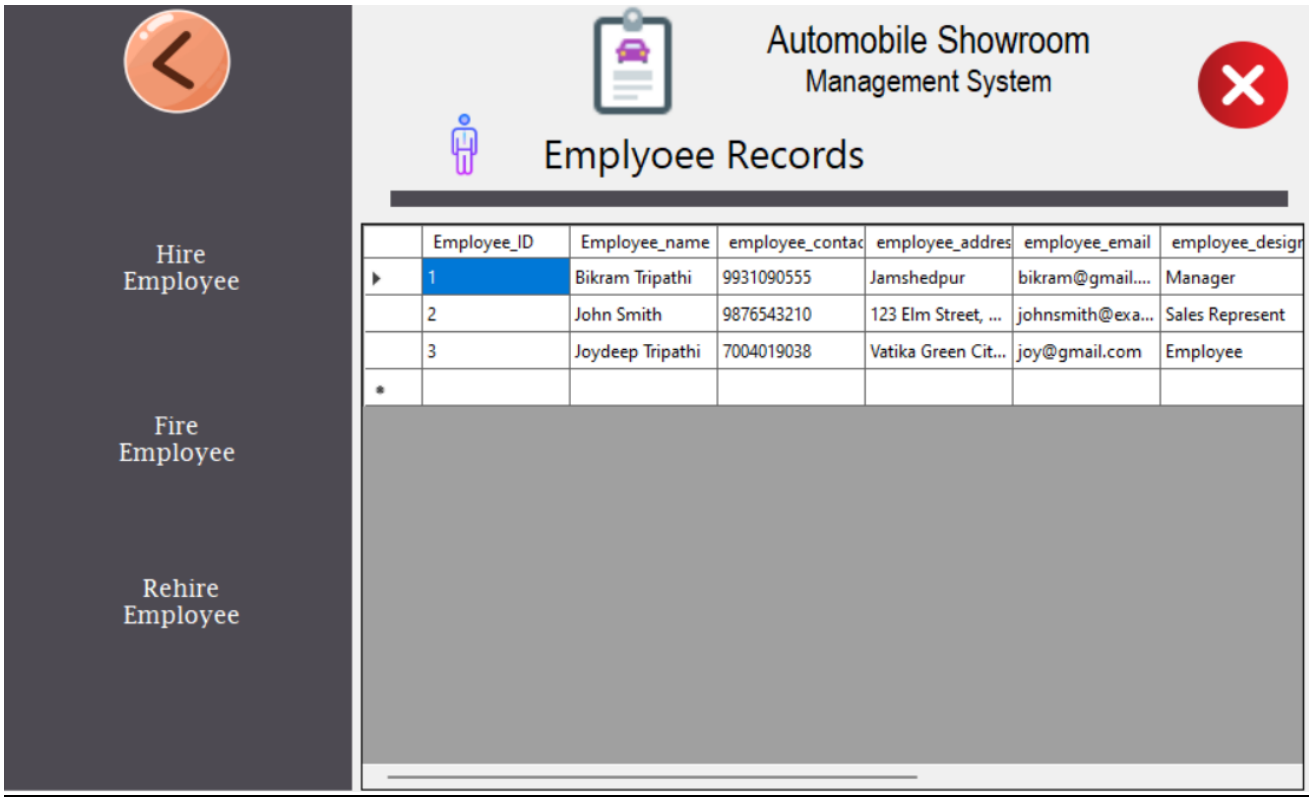
Login Form



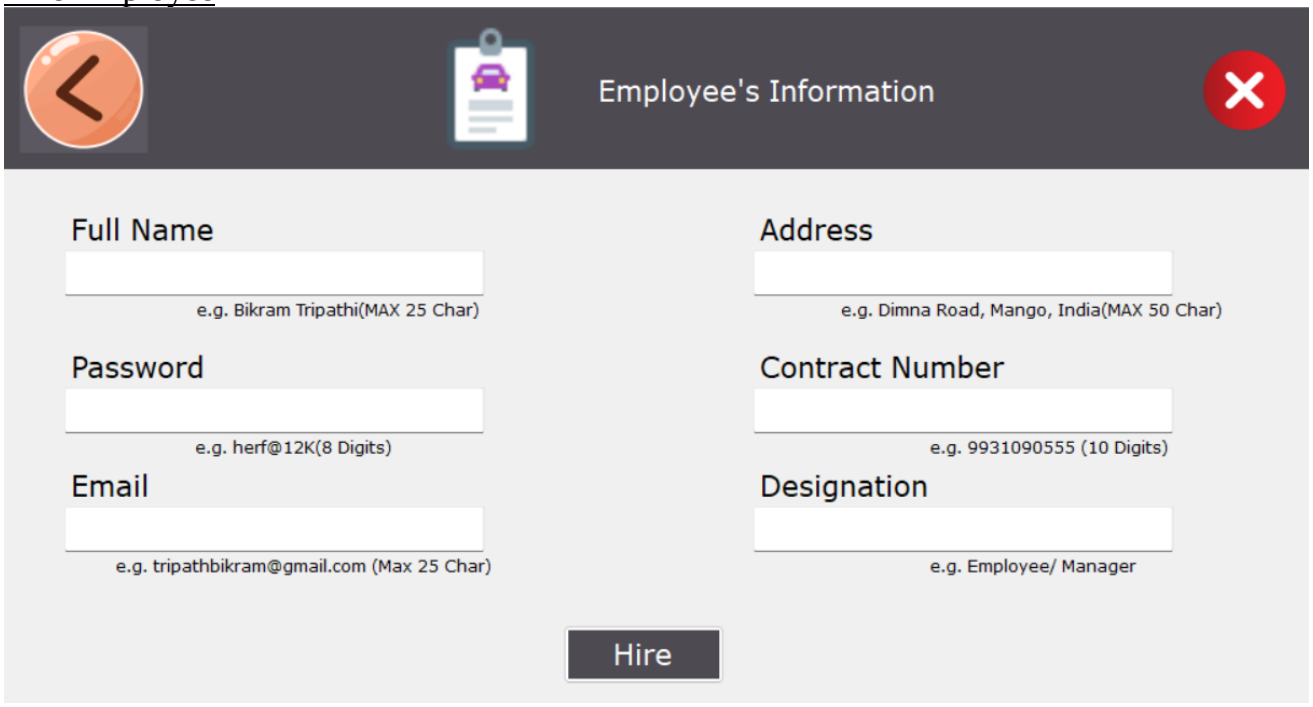
Manager Dashboard



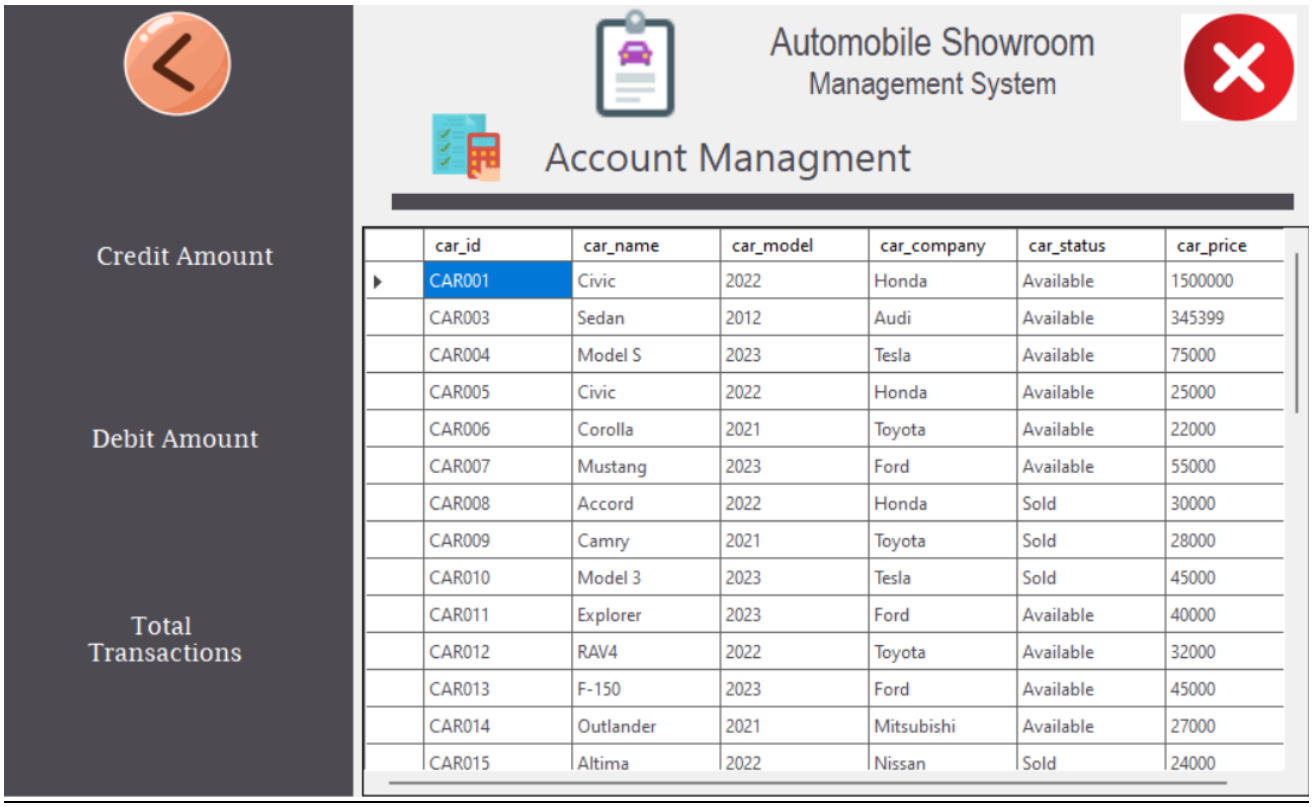
Control Employee



Hire Employee

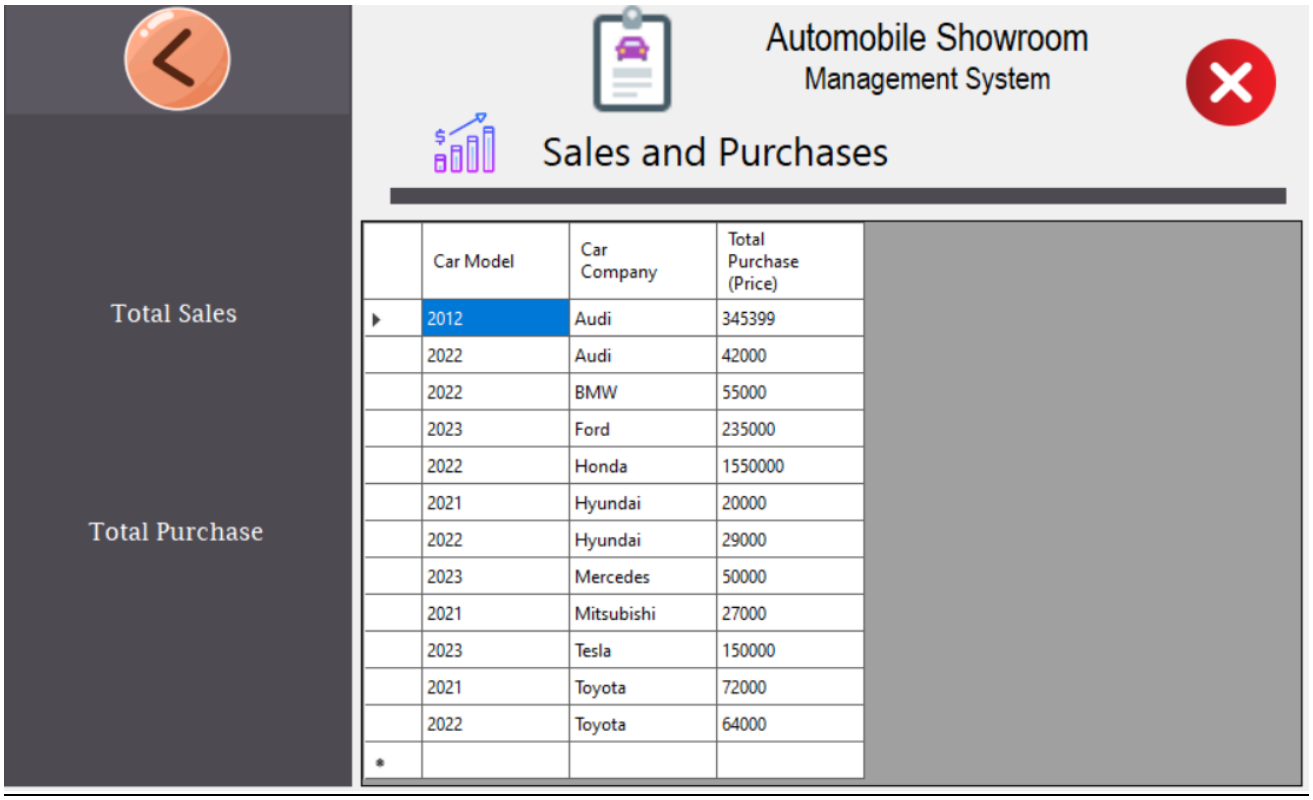


Account



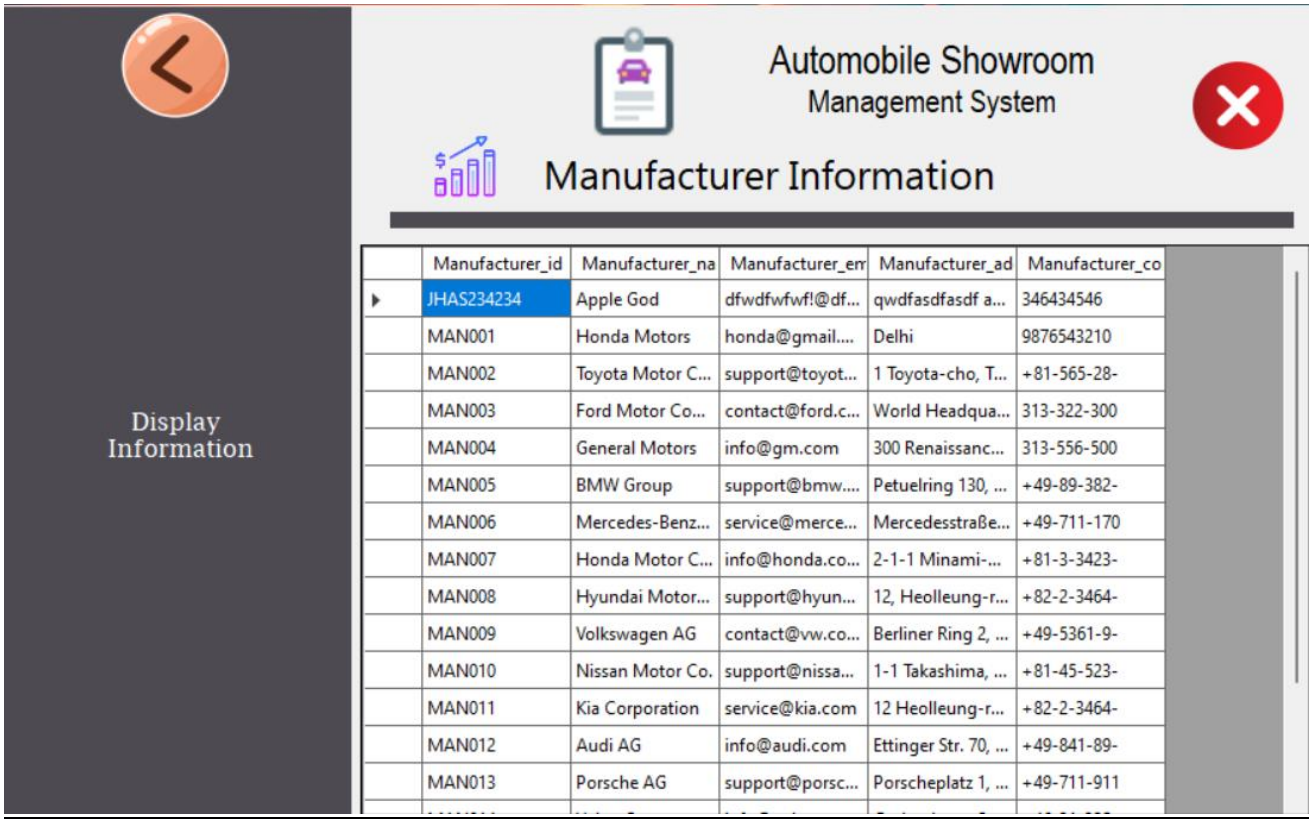
car_id	car_name	car_model	car_company	car_status	car_price
CAR001	Civic	2022	Honda	Available	1500000
CAR003	Sedan	2012	Audi	Available	345399
CAR004	Model S	2023	Tesla	Available	75000
CAR005	Civic	2022	Honda	Available	25000
CAR006	Corolla	2021	Toyota	Available	22000
CAR007	Mustang	2023	Ford	Available	55000
CAR008	Accord	2022	Honda	Sold	30000
CAR009	Camry	2021	Toyota	Sold	28000
CAR010	Model 3	2023	Tesla	Sold	45000
CAR011	Explorer	2023	Ford	Available	40000
CAR012	RAV4	2022	Toyota	Available	32000
CAR013	F-150	2023	Ford	Available	45000
CAR014	Outlander	2021	Mitsubishi	Available	27000
CAR015	Altima	2022	Nissan	Sold	24000

Sales Purchase



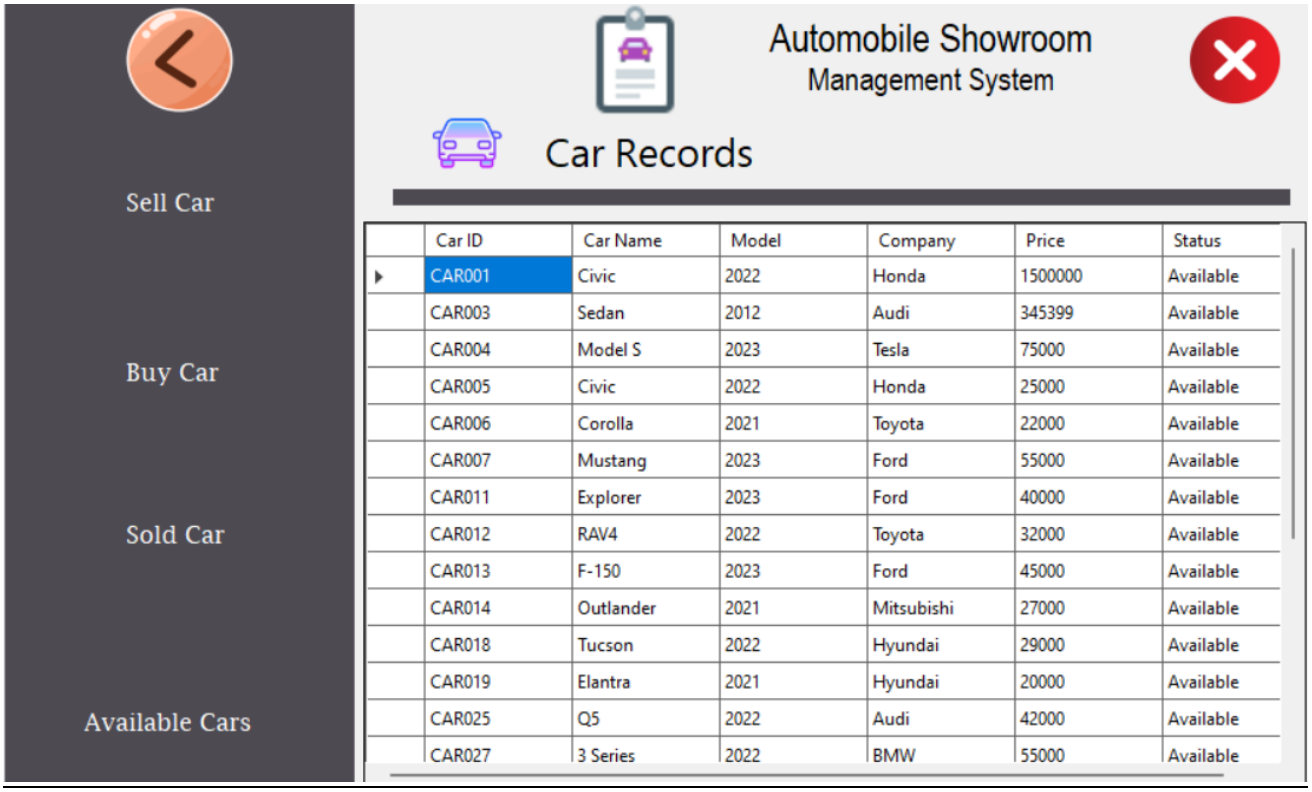
Car Model	Car Company	Total Purchase (Price)
2012	Audi	345399
2022	Audi	42000
2022	BMW	55000
2023	Ford	235000
2022	Honda	1550000
2021	Hyundai	20000
2022	Hyundai	29000
2023	Mercedes	50000
2021	Mitsubishi	27000
2023	Tesla	150000
2021	Toyota	72000
2022	Toyota	64000

Manufacturer Information



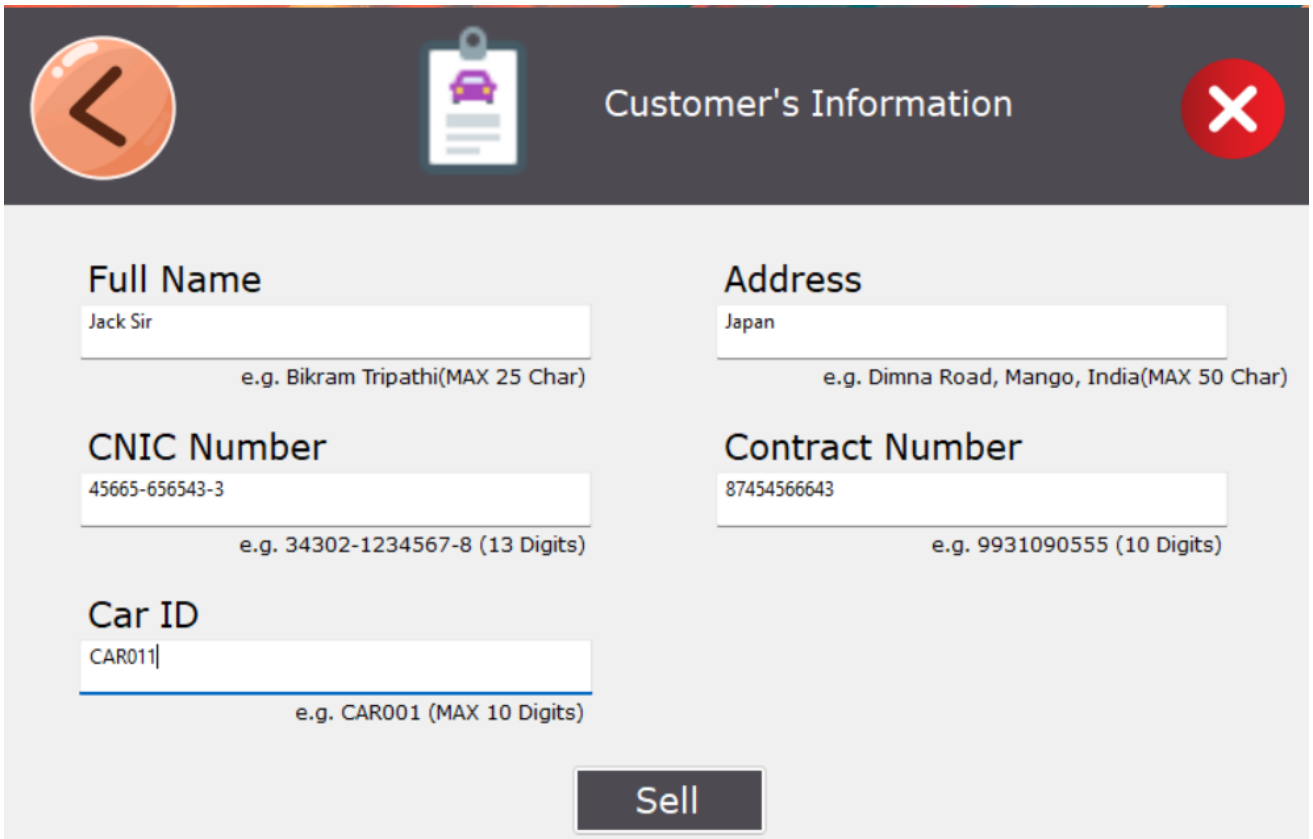
Manufacturer_id	Manufacturer_na	Manufacturer_en	Manufacturer_ad	Manufacturer_co
JHAS234234	Apple God	dfwdfwfwf!@df...	qwdfasdfasdf a...	346434546
MAN001	Honda Motors	honda@gmail....	Delhi	9876543210
MAN002	Toyota Motor C...	support@toyot...	1 Toyota-cho, T...	+81-565-28-
MAN003	Ford Motor Co...	contact@ford.c...	World Headqua...	313-322-300
MAN004	General Motors	info@gm.com	300 Renaissanc...	313-556-500
MAN005	BMW Group	support@bmw....	Petuelring 130, ...	+49-89-382-
MAN006	Mercedes-Benz...	service@merce...	Mercedesstraße...	+49-711-170
MAN007	Honda Motor C...	info@honda.co...	2-1-1 Minami-...	+81-3-3423-
MAN008	Hyundai Motor...	support@hyun...	12, Heolleung-r...	+82-2-3464-
MAN009	Volkswagen AG	contact@vw.co...	Berliner Ring 2, ...	+49-5361-9-
MAN010	Nissan Motor Co.	support@nissa...	1-1 Takashima, ...	+81-45-523-
MAN011	Kia Corporation	service@kia.com	12 Heolleung-r...	+82-2-3464-
MAN012	Audi AG	info@audi.com	Ettinger Str. 70, ...	+49-841-89-
MAN013	Porsche AG	support@porsc...	Porscheplatz 1, ...	+49-711-911

Employee Dashboard

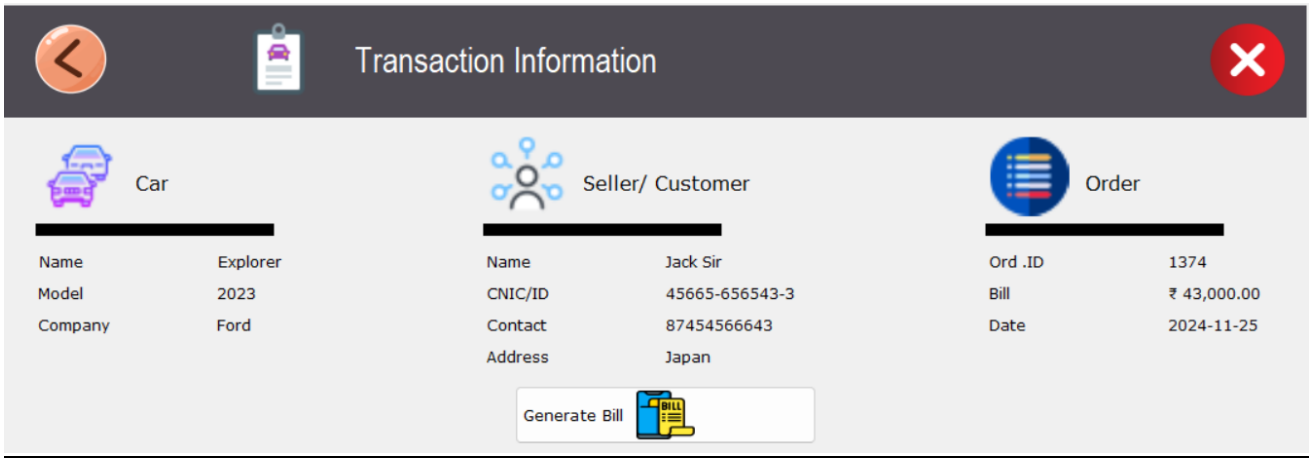


Car ID	Car Name	Model	Company	Price	Status
CAR001	Civic	2022	Honda	1500000	Available
CAR003	Sedan	2012	Audi	345399	Available
CAR004	Model S	2023	Tesla	75000	Available
CAR005	Civic	2022	Honda	25000	Available
CAR006	Corolla	2021	Toyota	22000	Available
CAR007	Mustang	2023	Ford	55000	Available
CAR011	Explorer	2023	Ford	40000	Available
CAR012	RAV4	2022	Toyota	32000	Available
CAR013	F-150	2023	Ford	45000	Available
CAR014	Outlander	2021	Mitsubishi	27000	Available
CAR018	Tucson	2022	Hyundai	29000	Available
CAR019	Elantra	2021	Hyundai	20000	Available
CAR025	Q5	2022	Audi	42000	Available
CAR027	3 Series	2022	BMW	55000	Available

Sell Car



Transaction Information



Generated Bill

CAR SHOWROOM BILL

CUSTOMER DETAILS

Name: Jack Sir

CNIC: 45665-656543-3

Contact: 87454566643

Address: Japan

CAR DETAILS

Car Name: Explorer

Model: 2023

Company: Ford

Price: ₹ 41,500.00

BILLING DETAILS

Tax Amount: ₹ 1,500.00

Total Bill: ₹ 43,000.00

ORDER DETAILS

Order ID: 1374

Order Date: 2024-11-25

Thank you for choosing our showroom!

Conclusion

The **Car Showroom Management System** was developed to streamline and enhance the operations of a car dealership, providing robust solutions for inventory management, customer orders, employee handling, and report generation. By employing the iterative waterfall model, each phase of development was meticulously planned, executed, and tested to ensure the system meets user requirements and business objectives.

The system successfully addresses challenges such as efficient car inventory tracking, automated billing, and role-based access for employees. Comprehensive testing verified its functionality, reliability, and seamless integration of modules. The project demonstrates the practical application of software engineering principles, delivering a solution that is scalable and adaptable to the evolving needs of the organization.

References

1. Sommerville, I. (2011). *Software Engineering (9th Edition)*. Addison-Wesley.
2. Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.
3. IEEE Standard for Software Testing Documentation (IEEE 829).
4. Visual Basic for Applications Guide. Available at: https://learn.microsoft.com/
5. Godfrey, M. (2008). *Practical Database Design*. Wiley.
6. Agile Alliance. *Iterative Waterfall vs. Agile Development*. Available at: https://www.agilealliance.org