
Experiment 1: UML Diagram for Hotel Reservation System

Aim:

To design a UML diagram for a hotel reservation system that allows customers to book hotels online.

Apparatus Required:

- Computer with CASE Tool
- UML Diagramming Software (e.g., Lucidchart, Microsoft Visio, StarUML)

Procedure:

1. Identify actors: Customer, Hotel Management System, and Admin.
2. Create use case diagrams to represent the interaction between actors and system functions.
3. Develop class diagrams including classes like Customer, Room, Reservation, Payment, etc.
4. Draw a sequence diagram for booking confirmation.
5. Verify the diagram for correctness.

Result:

Successfully designed a UML diagram representing the hotel reservation system.

Experiment 2: UML Diagram for Coffee Shop Ordering System

Aim:

To draw a UML diagram for a coffee shop ordering system using use case diagrams.

Apparatus Required:

- CASE Tool
- UML Diagramming Software

Procedure:

1. Identify actors: Customer, Service Assistant, Coffee Machine.
2. Identify use cases: Place Order, Dispense Coffee, Add Recipe, Edit/Delete Recipe.
3. Draw use case diagrams to represent system functionality.
4. Develop class and sequence diagrams for detailed operations.

Result:

A complete UML diagram illustrating the coffee shop ordering system is created.

Experiment 3: Class Diagram for Online Airline Reservation System

Aim:

To design a class diagram for an online airline reservation system.

Apparatus Required:

- CASE Tool
- UML Diagramming Software

Procedure:

1. Identify entities like User, Admin, Flight, Booking, Payment.
2. Define relationships among the classes.
3. Use association, aggregation, and inheritance to link classes.
4. Implement constraints using class attributes and methods.

Result:

A class diagram representing the airline reservation system is created.

Experiment 4: UML Diagram for ATM System

Aim:

To create a UML diagram for an ATM system.

Apparatus Required:

- CASE Tool
- UML Diagramming Software

Procedure:

1. Identify actors: Customer, ATM, Bank Database, Technician.
2. Create use case diagrams: Withdraw Cash, Check Balance, Deposit Money.
3. Develop sequence diagrams to visualize interactions.
4. Verify logical correctness.

Result:

A functional UML diagram of an ATM system is created.

Experiment 5: UML Diagram for Food Ordering System

Aim:

To draw a UML diagram for a food ordering system.

Apparatus Required:

- CASE Tool
- UML Diagramming Software

Procedure:

1. Identify actors: Customer, Chef, Delivery Person, Admin.
2. Draw use case diagrams for actions like Place Order, Prepare Food, Deliver Food.
3. Create sequence diagrams for detailed interactions.

Result:

A clear UML representation of the food ordering system is created.

Experiment 6: Use Case Diagram for Quiz System

Aim:

To create a use case diagram for a quiz system.

Apparatus Required:

- CASE Tool
- UML Diagramming Software

Procedure:

1. Identify actors: User, Helper, Admin.
2. Draw use case diagrams for actions like Start Quiz, Submit Answer, Provide Hint.
3. Develop class and sequence diagrams for detailed views.

Result:

A comprehensive use case diagram for the quiz system is created.

Experiment 7: UML Diagram for Online Purchasing System

Aim:

To design a UML diagram for an online purchasing system.

Apparatus Required:

- CASE Tool
- UML Diagramming Software

Procedure:

1. Identify actors: Web Customer, Payment Gateway, Admin.
2. Create use case diagrams for actions like View Items, Make Purchase, Register.

3. Draw sequence diagrams to show interaction flow.

Result:

A UML diagram representing the online purchasing system is completed.

Experiment 8: Hospital Reception Management System

Aim:

To describe major services provided by a hospital's reception using a system model.

Apparatus Required:

- CASE Tool
- UML Diagramming Software

Procedure:

1. Identify entities like Receptionist, Patient, Doctor, Payment System.
2. Draw use case diagrams for actions like Schedule Appointment, Admit Patient.
3. Visualize interactions using sequence diagrams.

Result:

A detailed hospital reception management system model is created.

Experiment 9: Class Diagram for Security Management System

Aim:

To develop a class diagram for a security management system.

Apparatus Required:

- CASE Tool
- UML Diagramming Software

Procedure:

1. Identify classes like Security Guard, Schedule, Manager, Payment.
2. Define class relationships and methods.
3. Develop a class diagram using aggregation and inheritance.

Result:

A functional class diagram for a security management system is created.

Experiment 10: Library Management System

Aim:

To draw a use case diagram for a library management system.

Apparatus Required:

- CASE Tool
- UML Diagramming Software

Procedure:

1. Identify actors: Student, Librarian, System.
2. Define use cases like Search Book, Issue Book, Return Book.
3. Draw a use case diagram to show interactions.

Result:

A well-defined use case diagram for the library management system is created.

Experiment 11: VB Program using TextBox, Label, and Command Button Control

Aim:

To develop a Visual Basic (VB) program using TextBox, Label, and Command Button controls.

Apparatus Required:

- Visual Studio IDE
- Windows PC with VB.NET installed

Procedure:

1. **Create Project:**
    - Open **Visual Studio** → Create a new **Windows Forms Application**.
2. **Add Controls:**
    - From the Toolbox, drag and drop:
        - **Label** to display a prompt.
        - **TextBox** for user input.
        - **Command Button** to trigger an action.
3. **Write Code:**
    - Double-click on the Button to open the code editor.
    - Write the following code to display input text using a message box.

Program:

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim userInput As String = TextBox1.Text
        MessageBox.Show("You entered: " & userInput)
    End Sub
End Class
```

Result:

A program was successfully created to display user input using TextBox, Label, and Command Button.

Experiment 12: VB Program using Checkbox and Option Button Control

Aim:

To develop a Visual Basic program using Checkbox and Option Button controls.

Apparatus Required:

- Visual Studio IDE
- Windows PC with VB.NET installed

Procedure:

1. **Create Project:**
    - Open **Visual Studio** → Create a **Windows Forms Application**.
2. **Add Controls:**
    - Add two **CheckBoxes** and two **Option Buttons**.
    - Add a **Command Button** for submitting selections.
3. **Write Code:**
    - Implement logic to read which CheckBox or Option Button is selected.

Program:

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim result As String = "You selected: "

        If CheckBox1.Checked Then
            result &= CheckBox1.Text & ", "
        End If
```

```
        If CheckBox2.Checked Then
            result &= CheckBox2.Text & ", "
        End If

        If RadioButton1.Checked Then
            result &= "Option: " & RadioButton1.Text
        ElseIf RadioButton2.Checked Then
            result &= "Option: " & RadioButton2.Text
        End If

        MessageBox.Show(result)
    End Sub
End Class
```

Result:

Successfully created a program using CheckBox and Option Button to capture user input.

Experiment 13: VB Program using ListBox and ComboBox Control

Aim:

To develop a Visual Basic program using ListBox and ComboBox controls.

Apparatus Required:

- Visual Studio IDE
- Windows PC with VB.NET installed

Procedure:

1. **Create Project:**
    - Create a **Windows Forms Application**.
2. **Add Controls:**
    - Drag and drop a **ListBox**, **ComboBox**, and **Button**.
3. **Populate Controls:**
    - Add items using Items.Add() method in the Form Load event.
4. **Write Code:**
    - Display the selected item using MessageBox.Show().

Program:

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ListBox1.Items.Add("Apple")
        ListBox1.Items.Add("Banana")
        ComboBox1.Items.Add("Mango")
        ComboBox1.Items.Add("Orange")
    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim result As String = "ListBox: " & ListBox1.SelectedItem & vbCrLf & "ComboBox: " & ComboBox1.SelectedItem
        MessageBox.Show(result)
    End Sub
End Class
```

Result:

The program displayed the selected items from ListBox and ComboBox successfully.

Experiment 14: VB Program using Validation Control

Aim:

To create a VB program using Validation Control for input validation.

Apparatus Required:

- Visual Studio IDE
- Windows PC with VB.NET installed

Procedure:

1. **Create Project:**
    - Create a **Windows Forms Application**.
2. **Add Controls:**
    - Add **TextBox** for input and a **Button** for submission.
    - Add an **ErrorProvider** from the toolbox.
3. **Write Code:**

- Validate user input using ErrorProvider.SetError().

Program:

```
Public Class Form1
  Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    If TextBox1.Text = "" Then
      ErrorProvider1.SetError(TextBox1, "Field cannot be empty!")
    Else
      ErrorProvider1.Clear()
      MessageBox.Show("Input is valid: " & TextBox1.Text)
    End If
  End Sub
End Class
```

Result:

The program successfully validated user input using ErrorProvider.

Experiment 15: VB Program for Arithmetic Operation

Aim:

To create a VB program to perform arithmetic operations.

Apparatus Required:

- Visual Studio IDE
- Windows PC with VB.NET installed

Procedure:

1. **Create Project:**
   - Create a **Windows Forms Application**.
2. **Add Controls:**
   - Add two **TextBoxes** for input numbers.
   - Add four **Buttons** for Addition, Subtraction, Multiplication, and Division.
3. **Write Code:**
   - Perform arithmetic operations based on button click.

Program:

```
Public Class Form1
```

```
    Private Sub ButtonAdd_Click(sender As Object, e As EventArgs) Handles ButtonAdd.Click
        Dim num1 As Double = Convert.ToDouble(TextBox1.Text)
        Dim num2 As Double = Convert.ToDouble(TextBox2.Text)
        MessageBox.Show("Addition: " & (num1 + num2))
    End Sub

    Private Sub ButtonSub_Click(sender As Object, e As EventArgs) Handles ButtonSub.Click
        Dim num1 As Double = Convert.ToDouble(TextBox1.Text)
        Dim num2 As Double = Convert.ToDouble(TextBox2.Text)
        MessageBox.Show("Subtraction: " & (num1 - num2))
    End Sub

    Private Sub ButtonMul_Click(sender As Object, e As EventArgs) Handles ButtonMul.Click
        Dim num1 As Double = Convert.ToDouble(TextBox1.Text)
        Dim num2 As Double = Convert.ToDouble(TextBox2.Text)
        MessageBox.Show("Multiplication: " & (num1 * num2))
    End Sub

    Private Sub ButtonDiv_Click(sender As Object, e As EventArgs) Handles ButtonDiv.Click
        Dim num1 As Double = Convert.ToDouble(TextBox1.Text)
        Dim num2 As Double = Convert.ToDouble(TextBox2.Text)
        If num2 <> 0 Then
            MessageBox.Show("Division: " & (num1 / num2))
        Else
            MessageBox.Show("Cannot divide by zero!")
        End If
    End Sub
End Class
```

Result:

The arithmetic operations were successfully performed using Visual Basic.

Experiment 16: VB Program for Web Server Control

Aim:

To create a VB.NET web application using Web Server Controls.

Apparatus Required:

- Visual Studio IDE
- ASP.NET framework
- Windows PC with IIS enabled

Procedure:

1. Open **Visual Studio** → Create a **New ASP.NET Web Application**.
2. Drag and drop **Label, TextBox, Button, and GridView** controls onto the **Web Form**.
3. Write code to process form input and display results using Response.Write().
4. Run the application using the built-in web server and validate functionality.

Program:

```
Public Class WebForm1
   Protected Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
      Label1.Text = "Welcome, " & TextBox1.Text
   End Sub
End Class
```

Result:

A web server control was successfully implemented in VB.NET.

Experiment 17: Create Sample Page Structure Using .NET

Aim:

To design a simple web page structure using .NET.

Apparatus Required:

- Visual Studio IDE
- .NET framework

Procedure:

1. Create an **ASP.NET Web Form**.
2. Design a **simple webpage layout** using HTML and **.NET controls**.
3. Use **Master Pages** for consistent design across multiple pages.

4. Run the application and validate the output.

Program:

```
<%@ Page Language="VB" AutoEventWireup="true" CodeBehind="WebForm1.aspx.vb" Inherits="MyWebsite.WebForm1" %>

<!DOCTYPE html>

<html>

<head>

  <title>Sample Page</title>

</head>

<body>

  <form id="form1" runat="server">

    <h1>Welcome to .NET Web Development</h1>

    <asp:Button ID="Button1" runat="server" Text="Click Me" />

  </form>

</body>

</html>
```

Result:

A structured .NET webpage was successfully created.

Experiment 18: Create an Ad Rotator Control for Hotel Advertisement

Aim:

To develop an **Ad Rotator** control for hotel advertisement using ASP.NET.

Apparatus Required:

- Visual Studio IDE
- .NET framework

Procedure:

1. Create a **New ASP.NET Web Application**.
2. Use the **AdRotator Control** to display hotel advertisements dynamically.
3. Define an **XML file** containing advertisement details.
4. Run the application and validate the ad rotation.

Program:

ads.xml

```
<Advertisements>
  <Ad>
    <ImageUrl>~/images/hotel1.jpg</ImageUrl>
    <NavigateUrl="https://www.hotel1.com"/>
    <AlternateText>Hotel 1</AlternateText>
  </Ad>
  <Ad>
    <ImageUrl>~/images/hotel2.jpg</ImageUrl>
    <NavigateUrl="https://www.hotel2.com"/>
    <AlternateText>Hotel 2</AlternateText>
  </Ad>
</Advertisements>
```

ASP.NET Page:

```
<asp:AdRotator ID="AdRotator1" runat="server" AdvertisementFile="~/ads.xml" />
```

Result:

An **Ad Rotator** control was successfully implemented for hotel advertisement.

Experiment 19: Working with Calendar Control for Checking Voting Eligibility

Aim:

To implement a calendar control to check voter eligibility based on birth date.

Apparatus Required:

- Visual Studio IDE
- .NET framework

Procedure:

1. Drag and drop a **Calendar Control** and **Button** in an ASP.NET Web Form.
2. Write code to calculate the user's age.
3. Display eligibility status in a **Label**.

Program:

```
Public Class WebForm1
  Protected Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
```

```
        Dim birthYear As Integer = Calendar1.SelectedDate.Year
        Dim currentYear As Integer = DateTime.Now.Year
        Dim age As Integer = currentYear - birthYear

        If age >= 18 Then
            Label1.Text = "You are eligible to vote!"
        Else
            Label1.Text = "You are not eligible to vote!"
        End If
    End Sub
End Class
```

Result:

The program correctly determines **voting eligibility** based on the selected birth date.

Experiment 20: Connect Frontend and Backend to Collect a Student Feedback Form

Aim:

To develop a student feedback system by connecting **frontend and backend** using .NET and SQL.

Apparatus Required:

- Visual Studio IDE
- SQL Server

Procedure:

1. Create a **Windows Form** with **TextBoxes** for student details.
2. Use a **Button** to submit the feedback.
3. Connect the form to an **SQL database** and store feedback.

Program:

SQL Table Creation:

```
CREATE TABLE Feedback (
    StudentID INT PRIMARY KEY,
    Name VARCHAR(50),
    Feedback TEXT
);
```

VB.NET Code:

```
Imports System.Data.SqlClient


Public Class Form1
    Dim conn As New SqlConnection("your_connection_string")


    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim cmd As New SqlCommand("INSERT INTO Feedback VALUES (@id, @name, @feedback)", conn)
        cmd.Parameters.AddWithValue("@id", TextBox1.Text)
        cmd.Parameters.AddWithValue("@name", TextBox2.Text)
        cmd.Parameters.AddWithValue("@feedback", TextBox3.Text)


        conn.Open()
        cmd.ExecuteNonQuery()
        conn.Close()
        MessageBox.Show("Feedback Submitted!")
    End Sub
End Class
```

Result:

The **student feedback** form successfully stores data into the **SQL database**.

Experiment 21: Students Course Registration using DML and DQL

Aim:

To implement **DML (Data Manipulation Language) and DQL (Data Query Language)** for course registration.

Apparatus Required:

- SQL Server

Procedure:

1. Create a **Course Registration Table**.
2. Use **INSERT (DML)** to register students.
3. Use **SELECT (DQL)** to retrieve student details.

SQL Queries:

```
-- Create table
CREATE TABLE CourseRegistration (
   StudentID INT PRIMARY KEY,
   Name VARCHAR(50),
   Course VARCHAR(50)
);


-- Insert data (DML)
INSERT INTO CourseRegistration VALUES (101, 'John Doe', 'Computer Science');


-- Retrieve data (DQL)
SELECT * FROM CourseRegistration;
```

Result:

Successfully implemented **course registration** using SQL.

Experiment 22: Develop a Personal Information System for Employees Using DML and DQL

Aim:

To create a personal information management system for employees using SQL with **DML** (Data Manipulation Language) and **DQL** (Data Query Language).

Apparatus Required:

- Visual Studio IDE
- SQL Server

Procedure:

1. Create a table named EmployeeInfo using SQL.
2. Use **INSERT** statements to add data.
3. Use **SELECT** queries to retrieve employee information.
4. Develop a simple VB.NET form for data input and output.

SQL Queries:

```
-- Create Employee Table
CREATE TABLE EmployeeInfo (
```

```
    EmpID INT PRIMARY KEY,
    Name VARCHAR(50),
    Age INT,
    Department VARCHAR(50),
    Salary DECIMAL(10,2)
);

-- Insert Employee Data (DML)
INSERT INTO EmployeeInfo VALUES (101, 'John', 30, 'HR', 50000.00);
INSERT INTO EmployeeInfo VALUES (102, 'Alice', 28, 'IT', 60000.00);

-- Retrieve Data (DQL)
SELECT * FROM EmployeeInfo;
```

VB.NET Program:

```
Imports System.Data.SqlClient

Public Class Form1
    Dim conn As New SqlConnection("your_connection_string")

    Private Sub BtnInsert_Click(sender As Object, e As EventArgs) Handles BtnInsert.Click
        Dim cmd As New SqlCommand("INSERT INTO EmployeeInfo VALUES (@EmpID, @Name, @Age, @Department, @Salary)", conn)
        cmd.Parameters.AddWithValue("@EmpID", TextBox1.Text)
        cmd.Parameters.AddWithValue("@Name", TextBox2.Text)
        cmd.Parameters.AddWithValue("@Age", TextBox3.Text)
        cmd.Parameters.AddWithValue("@Department", TextBox4.Text)
        cmd.Parameters.AddWithValue("@Salary", TextBox5.Text)

        conn.Open()
        cmd.ExecuteNonQuery()
        conn.Close()
```

```
        MessageBox.Show("Data Inserted Successfully")
    End Sub
End Class
```

Result:

Successfully created an employee management system using SQL and VB.NET.

Experiment 23: Create Multiple Databases for Employees with Authorization

Aim:

To implement multiple databases for employee information with authorization using SQL.

Apparatus Required:

- SQL Server

Procedure:

1. Create **multiple databases** for different departments.
2. Define roles: **Partner, Manager, Employee**.
3. Grant access permissions based on roles using GRANT statements.

SQL Queries:

```
-- Create Databases
CREATE DATABASE CompanyDB;
CREATE DATABASE HRDB;

-- Create Roles
CREATE ROLE Partner;
CREATE ROLE Manager;
CREATE ROLE Employee;

-- Grant Permissions
USE CompanyDB;
GRANT SELECT, INSERT, UPDATE ON EmployeeInfo TO Manager;
GRANT SELECT ON EmployeeInfo TO Employee;
GRANT ALL PRIVILEGES ON EmployeeInfo TO Partner;
```

Result:

Multiple databases with proper authorization were successfully created.

Experiment 24: Payroll Processing System Using Data Grid Control

Aim:

To develop a payroll processing system using **Data Grid Control** in VB.NET.

Apparatus Required:

- Visual Studio IDE
- SQL Server

Procedure:

1. Create a table for payroll management.
2. Implement a **DataGridView** in VB.NET.
3. Fetch data using SQL and display it using Data Grid.

SQL Query:

```
CREATE TABLE Payroll (
   EmpID INT PRIMARY KEY,
   Name VARCHAR(50),
   Salary DECIMAL(10,2),
   Deductions DECIMAL(10,2),
   NetSalary AS (Salary - Deductions)
);


INSERT INTO Payroll VALUES (101, 'John', 60000.00, 5000.00);
```

VB.NET Program:

```
Imports System.Data.SqlClient


Public Class Form1
   Dim conn As New SqlConnection("your_connection_string")
   Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
      Dim adapter As New SqlDataAdapter("SELECT * FROM Payroll", conn)
      Dim table As New DataTable()
```

```
        adapter.Fill(table)
        DataGridView1.DataSource = table
    End Sub
End Class
```

Result:

Payroll details were displayed using the Data Grid Control.

Experiment 25: Hospital HR System for Checking Salary

Aim:

To develop a hospital HR system that checks whether an employee's salary exceeds **₹80,000**.

Apparatus Required:

- SQL Server
- Visual Studio IDE

Procedure:

1. Create a table using SQL.
2. Write a query to filter employees with salaries above ₹80,000.

SQL Query:

```
CREATE TABLE HospitalEmployees (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(50),
    Designation VARCHAR(50),
    Salary DECIMAL(10,2)
);

INSERT INTO HospitalEmployees VALUES (201, 'Dr. Smith', 'Surgeon', 90000.00);
INSERT INTO HospitalEmployees VALUES (202, 'Nurse Linda', 'Nurse', 75000.00);

-- Query to check salary > ₹80,000
SELECT * FROM HospitalEmployees WHERE Salary > 80000;
```

Result:

The system successfully filtered employees with salaries greater than ₹80,000.

Experiment 26: Bank Transaction System for Credit Card

Aim:

To simulate a **bank transaction system** for credit card transactions using SQL.

Apparatus Required:

- SQL Server

Procedure:

1. Create a table for transactions.
2. Write SQL queries to insert transactions and check balance.

SQL Queries:

```
CREATE TABLE CreditCardTransactions (
    TransactionID INT PRIMARY KEY,
    CardNumber VARCHAR(16),
    Amount DECIMAL(10,2),
    TransactionType VARCHAR(10) CHECK (TransactionType IN ('Debit', 'Credit')),
    TransactionDate DATE
);


-- Insert a Transaction
INSERT INTO CreditCardTransactions VALUES (1, '1234567890123456', 5000.00, 'Debit', GETDATE());


-- Check Balance
SELECT CardNumber, SUM(CASE WHEN TransactionType = 'Credit' THEN Amount ELSE -Amount END)
AS Balance
FROM CreditCardTransactions
GROUP BY CardNumber;
```

Result:

The credit card transaction was successfully simulated.

✅ **Experiment 27: Electricity Bill Preparation System**

Aim:

To develop an **Electricity Bill Preparation System** using Visual Basic and SQL to calculate customer bills.

Apparatus Required:

- Visual Studio IDE
- SQL Server

Procedure:

1. Create a table named **ElectricityBills** using SQL to store customer details and units consumed.
2. Calculate the electricity bill based on unit consumption using VB.NET.
3. Display the calculated bill using **DataGridView**.

SQL Query:

```
CREATE TABLE ElectricityBills (
   BillID INT PRIMARY KEY,
   CustomerName VARCHAR(50),
   UnitsConsumed INT,
   BillAmount DECIMAL(10,2)
);
```

VB.NET Program:

```
Imports System.Data.SqlClient

Public Class Form1
   Dim conn As New SqlConnection("your_connection_string")

   Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
      Dim units As Integer = Convert.ToInt32(TextBox1.Text)
      Dim amount As Double

      If units <= 100 Then
         amount = units * 3
      ElseIf units <= 300 Then
         amount = 100 * 3 + (units - 100) * 5
```

```
        Else
            amount = 100 * 3 + 200 * 5 + (units - 300) * 7
        End If

        MessageBox.Show("Total Bill: ₹" & amount)
    End Sub
End Class
```

Result:

Electricity bill was calculated based on unit consumption using VB.NET.

✅ **Experiment 28: E-commerce System with Product Management**

Aim:

To develop an **E-commerce System** using VB.NET and SQL for managing product details.

Apparatus Required:

- Visual Studio IDE
- SQL Server

Procedure:

1. Create a **Products** table using SQL.
2. Design a form using VB.NET with fields for Product ID, Name, Price, and Quantity.
3. Implement functions to add, update, delete, and view products.

SQL Query:

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(50),
    Price DECIMAL(10,2),
    Quantity INT
);
```

VB.NET Program:

```
Imports System.Data.SqlClient

Public Class Form1
```

```
    Dim conn As New SqlConnection("your_connection_string")


    Private Sub BtnAdd_Click(sender As Object, e As EventArgs) Handles BtnAdd.Click
        Dim cmd As New SqlCommand("INSERT INTO Products VALUES (@ProductID, @ProductName, @Price, @Quantity)", conn)
        cmd.Parameters.AddWithValue("@ProductID", TextBox1.Text)
        cmd.Parameters.AddWithValue("@ProductName", TextBox2.Text)
        cmd.Parameters.AddWithValue("@Price", TextBox3.Text)
        cmd.Parameters.AddWithValue("@Quantity", TextBox4.Text)

        conn.Open()
        cmd.ExecuteNonQuery()
        conn.Close()

        MessageBox.Show("Product Added Successfully")
    End Sub
End Class
```

Result:

Products were successfully managed using VB.NET and SQL.

✅ Experiment 29: Hotel Management System with Bookings

Aim:

To develop a **Hotel Management System** using Visual Basic and SQL for room booking management.

Apparatus Required:

- Visual Studio IDE
- SQL Server

Procedure:

1. Create a table named **Bookings** using SQL.
2. Design a form in VB.NET to enter booking details.
3. Implement SQL queries for booking, canceling, and checking room availability.

SQL Query:

```
CREATE TABLE Bookings (
   BookingID INT PRIMARY KEY,
   CustomerName VARCHAR(50),
   RoomType VARCHAR(20),
   CheckInDate DATE,
   CheckOutDate DATE
);
```

VB.NET Program:

```
Imports System.Data.SqlClient

Public Class Form1
   Dim conn As New SqlConnection("your_connection_string")

   Private Sub BtnBook_Click(sender As Object, e As EventArgs) Handles BtnBook.Click
      Dim cmd As New SqlCommand("INSERT INTO Bookings VALUES (@BookingID, @CustomerName, @RoomType, @CheckInDate, @CheckOutDate)", conn)
      cmd.Parameters.AddWithValue("@BookingID", TextBox1.Text)
      cmd.Parameters.AddWithValue("@CustomerName", TextBox2.Text)
      cmd.Parameters.AddWithValue("@RoomType", ComboBox1.Text)
      cmd.Parameters.AddWithValue("@CheckInDate", DateTimePicker1.Value)
      cmd.Parameters.AddWithValue("@CheckOutDate", DateTimePicker2.Value)

      conn.Open()
      cmd.ExecuteNonQuery()
      conn.Close()

      MessageBox.Show("Room Booked Successfully")
   End Sub
End Class
```

Result:

The Hotel Management System successfully booked rooms using SQL.

✅ Experiment 30: Railway Reservation System

Aim:

To design a **Railway Reservation System** using Visual Basic and SQL for booking, canceling, and viewing reservations.

Apparatus Required:

- Visual Studio IDE
- SQL Server

Procedure:

1. Create a table named **RailwayReservations** using SQL.
2. Design a form using VB.NET to enter reservation details.
3. Implement booking, canceling, and displaying reservation details using SQL queries.

SQL Query:

```
CREATE TABLE RailwayReservations (
    ReservationID INT PRIMARY KEY,
    PassengerName VARCHAR(50),
    TrainNumber INT,
    Class VARCHAR(20),
    DateOfJourney DATE,
    Status VARCHAR(10) DEFAULT 'Booked'
);
```

VB.NET Program:

```
Imports System.Data.SqlClient

Public Class Form1
    Dim conn As New SqlConnection("your_connection_string")

    ' Book Ticket
    Private Sub BtnBook_Click(sender As Object, e As EventArgs) Handles BtnBook.Click
        Dim cmd As New SqlCommand("INSERT INTO RailwayReservations VALUES (@ReservationID, @PassengerName, @TrainNumber, @Class, @DateOfJourney, 'Booked')", conn)
        cmd.Parameters.AddWithValue("@ReservationID", TextBox1.Text)
```

```
        cmd.Parameters.AddWithValue("@PassengerName", TextBox2.Text)
        cmd.Parameters.AddWithValue("@TrainNumber", TextBox3.Text)
        cmd.Parameters.AddWithValue("@Class", ComboBox1.Text)
        cmd.Parameters.AddWithValue("@DateOfJourney", DateTimePicker1.Value)

        conn.Open()
        cmd.ExecuteNonQuery()
        conn.Close()

        MessageBox.Show("Reservation Confirmed")
    End Sub

    ' Cancel Ticket
    Private Sub BtnCancel_Click(sender As Object, e As EventArgs) Handles BtnCancel.Click
        Dim cmd As New SqlCommand("UPDATE RailwayReservations SET Status='Cancelled' WHERE
ReservationID=@ReservationID", conn)
        cmd.Parameters.AddWithValue("@ReservationID", TextBox1.Text)

        conn.Open()
        cmd.ExecuteNonQuery()
        conn.Close()

        MessageBox.Show("Reservation Cancelled")
    End Sub
End Class
```

Result:

Successfully implemented a **Railway Reservation System** using VB.NET and SQL.
