LAB 06

Problem 1 (a): Enhanced Schema of PVFC Database

To implement RBAC, the database schema required modification to separate authentication credentials from the core customer data. A normalized approach was used, introducing a Roles table to define access levels and a Users table to securely store login credentials and associate them with specific roles via a Foreign Key constraint.

SQL Schema Enhancement:

SQL

```
-- 1. Creating Roles table to define available roles

CREATE TABLE Roles (

   RoleId INT PRIMARY KEY,

   RoleName VARCHAR(50) NOT NULL UNIQUE

);


-- 2. Insert foundational RBAC roles

INSERT INTO Roles (RoleId, RoleName)

VALUES (1, 'Admin'), (2, 'Customer');


-- 3. Creating Users table for authentication

CREATE TABLE Users (

   UserId INT IDENTITY(1,1) PRIMARY KEY,

   Username VARCHAR(50) NOT NULL UNIQUE,

   Password VARCHAR(50) NOT NULL,
```

```
    RoleId INT NOT NULL,

    -- Enforce referential integrity for RBAC
    CONSTRAINT FK_Users_Roles FOREIGN KEY (RoleId) REFERENCES Roles(RoleId)
);

-- 4. Populate test accounts
INSERT INTO Users (Username, Password, RoleId)
VALUES
    ('admin@pvfc.com', 'admin123', 1),
    ('customer@pvfc.com', 'cust123', 2);
```

Problem 1 (b): Updated Code Incorporating RBAC

The RBAC implementation required modifications across both the frontend (to hide unauthorized links dynamically) and the backend (to strictly enforce access control).

1. Frontend Dynamic Routing (Site1.Master)

A Master Page was utilized to centralize the navigation bar. Visual Basic inline tags evaluate the current session's RoleId to dynamically render links based on the user's authorization level.

HTML

```
<div class="navbar">
    <a href="Default.aspx">Home</a>
    <a href="productSearch.aspx">Products</a>

    <%-- Visible ONLY to Admins (Role 1) --%>
```

```
    <% If Session("RoleId") IsNot Nothing AndAlso Convert.ToInt32(Session("RoleId")) = 1 Then %>
        <a href="catalog.aspx">Catalog</a>
    <% End If %>

    <%-- Visible ONLY to Customers (Role 2) --%>
    <% If Session("RoleId") IsNot Nothing AndAlso Convert.ToInt32(Session("RoleId")) = 2 Then %>
        <a href="updateCustomer.aspx">Update Profile</a>
        <a href="payment.aspx">Payment</a>
    <% End If %>

    <%-- Login/Logout Toggle --%>
    <% If Session("RoleId") Is Nothing Then %>
        <a href="register.aspx">Register</a>
        <a href="login.aspx">Login</a>
    <% Else %>
        <a href="login.aspx">Logout</a>
    <% End If %>
</div>
```

2. Backend Authentication Logic (login.aspx.vb)

The login backend was updated to query the database using parameterized inputs to prevent SQL Injection. Upon successful validation, the user's RoleId is stored in a server-side Session variable, which dictates their routing.

VB.Net

```
Protected Sub btnlogin_Click(sender As Object, e As EventArgs) Handles btnlogin.Click
    Dim constr As String =
```

```
ConfigurationManager.ConnectionStrings("PVFCConnString").ConnectionString

   Dim query As String = "SELECT RoleId FROM Users WHERE Username = @Username AND
Password = @Password"

   Try

      Using con As New SqlConnection(constr)          Using cmd As New
SqlCommand(query, con)
cmd.Parameters.AddWithValue("@Username", emailTB.Text.Trim())
cmd.Parameters.AddWithValue("@Password", pwdTB.Text)


            con.Open()


            Using dr As SqlDataReader = cmd.ExecuteReader()
               If dr.Read() Then
                  Dim role As Integer = Convert.ToInt32(dr("RoleId"))


                  ' Establish RBAC Session
                  Session("RoleId") = role


                  ' Route based on role
                  If role = 1 Then
                     Response.Redirect("adminDashboard.aspx", False)
                  ElseIf role = 2 Then
                     Response.Redirect("demo.aspx", False)
                  Else
```

```
                    lblMessage.Text = "Role Not Found"
                End If
            Else
                lblMessage.Text = "Cannot login. Invalid credentials."
            End If
        End Using
    End Using
End Using
Catch ex As Exception        lblMessage.Text = "An error
occurred: " & ex.Message
End Try
End Sub
```

3. Backend Authorization Enforcement (catalog.aspx.vb)

Even if frontend links are hidden, backend routes must be secured against direct URL access. The Page_Load event evaluates the session token before allowing the page to render.

VB.Net

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Load
    ' RBAC ENFORCEMENT: Admin Only (Role 1)
    If Session("RoleId") Is Nothing OrElse Convert.ToInt32(Session("RoleId")) <> 1 Then
        ' Redirect unauthorized users
        Response.Redirect("login.aspx")
        Exit Sub
    End If
```

```
    If Not IsPostBack Then
        BindCatalog()
    End If
End Sub
```

(Note: A similar check for Role = 2 was implemented on updateCustomer.aspx.vb and payment.aspx.vb).

Problem 1 (c): Test Cases for Role-Based Access Control

Test Case ID	Scenario	Input Data	Expected Output
1	**Valid Admin Login**	Username: admin@pvfc.com Password: admin123 *(RoleId: 1)*	System authenticates user, establishes session, redirects to adminDashboard.aspx, and displays Admin navbar links.
2	**Valid Customer Login**	Username: customer1@pvfc.com Password: cust123 *(RoleId: 2)*	System authenticates user, establishes session, redirects to demo.aspx, and displays Customer navbar links.
3	**Invalid Credentials**	Username: wrong_user Password: wrong_pass	Authentication fails. System denies access and displays: "Cannot login. Invalid credentials."
4	**Unauthorized Route Access**	User attempts to access catalog.aspx via direct URL input while logged out.	Authorization fails. The backend Page_Load check intercepts the request and redirects to login.aspx.

5	**SQL Injection Attempt**	Username: ' OR '1'='1 Password: password	Authentication fails. Parameterized query sanitizes the malicious string, resulting in "Cannot login."

Implementation Note regarding Logout Mechanism

Since comprehensive state session management has not been fully implemented for this application scope, the logout functionality operates on a simplified mechanism.

Currently, when a user clicks "Logout" in the navigation bar, they are simply redirected to login.aspx. Within the Page_Load subroutine of that login page, the commands Session.Clear() and Session.Abandon() are triggered. This forces the server to drop the in-memory RoleId key tied to that user's temporary browser footprint. Consequently, any subsequent attempts to load secure pages will fail the backend If Session("RoleId") Is Nothing checks, successfully locking the user out.