

Security Features Report - SaaS Platform

Introduction:

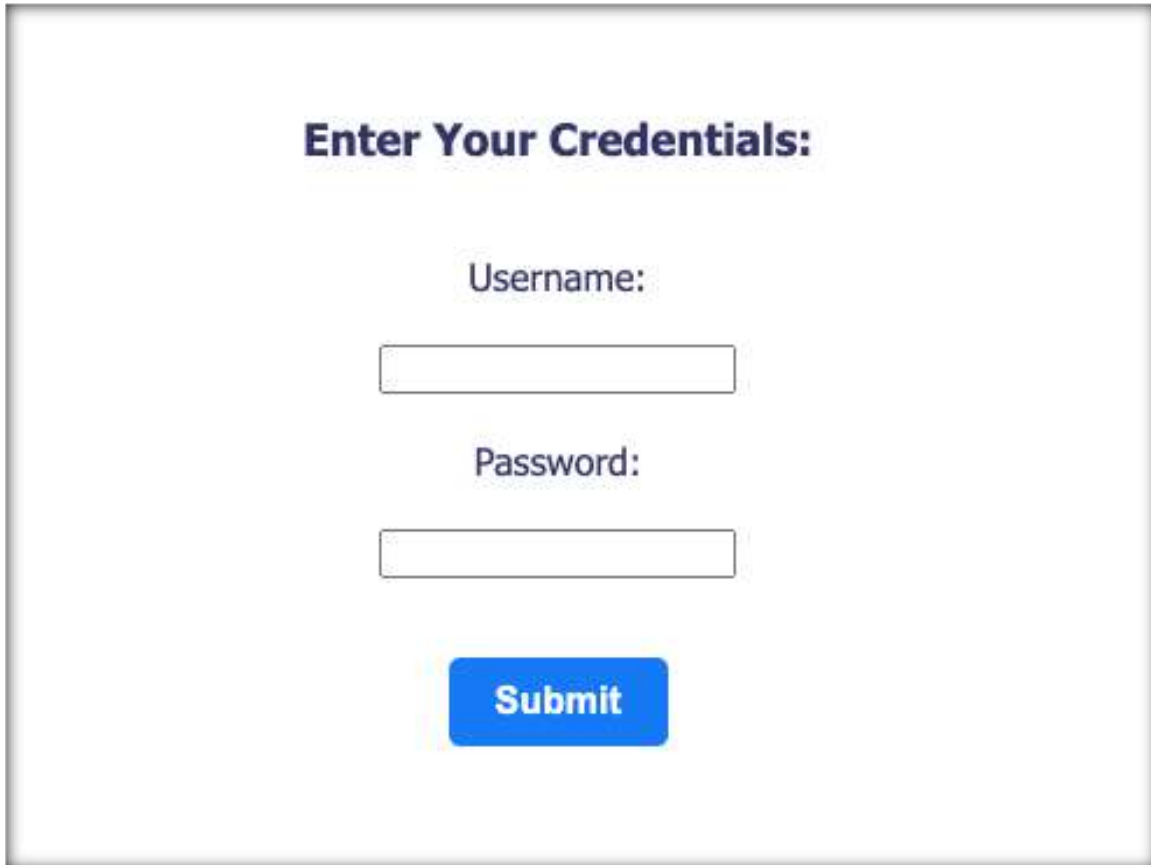
In this project, several essential security features have been implemented to safeguard user accounts, data, and access to functionalities. Below is a list of the key security mechanisms integrated into our SaaS application.

Summary Table of Security Features I Implemented in my Application:

Security Feature	Purpose
User Authentication	Verify user identity using username and password.
Session Timeout (Customer)	Terminate inactive customer sessions after 5 minutes.
Cookie Invalidation (Admin)	Invalidate admin session cookies after 5 minutes of inactivity.
Role-Based Access Control	Restrict access based on roles (Admin/Customer).
Input Validation	Ensure valid input for passwords, email, phone, etc.
Parameterized Queries	Prevent SQL Injection attacks.
SSL Certificate	Encrypt communication between client and server.
Encrypted ViewState	Protect page state data from tampering.

1. User Authentication (Username & Password)

Authentication ensures that only authorized users can access the application. A login system using usernames and passwords has been implemented. Each user's credentials are verified against stored data before granting access. This mechanism forms the foundation of the application's security and prevents unauthorized access.



The image shows a user authentication form titled "Enter Your Credentials:". It contains two input fields: "Username:" and "Password:". Below the "Password:" field is a blue "Submit" button. The form is enclosed in a light gray border.

Enter Your Credentials:

Username:

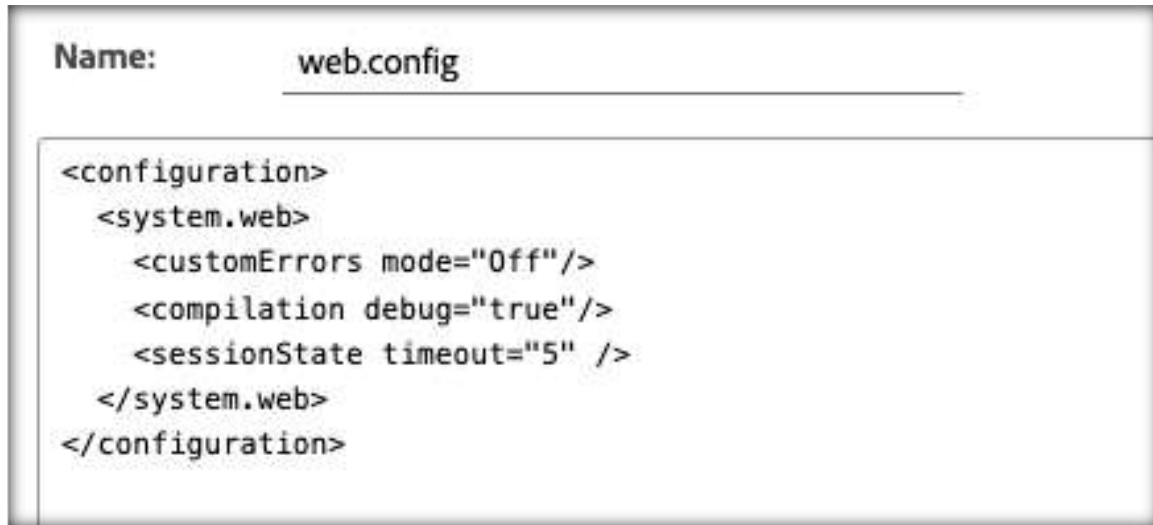
Password:

Submit

2. Session Timeout for All Customer Pages (5 Minutes)

Session-based logout technique is enforced on all customer pages to prevent unauthorized access due to unattended sessions. If a customer remains inactive for 5 minutes (timeout is set to ease testing for sir), their session is automatically terminated.

Below is the screenshot of web.config in root folder where session timeout is set to be 5 minutes for all pages.

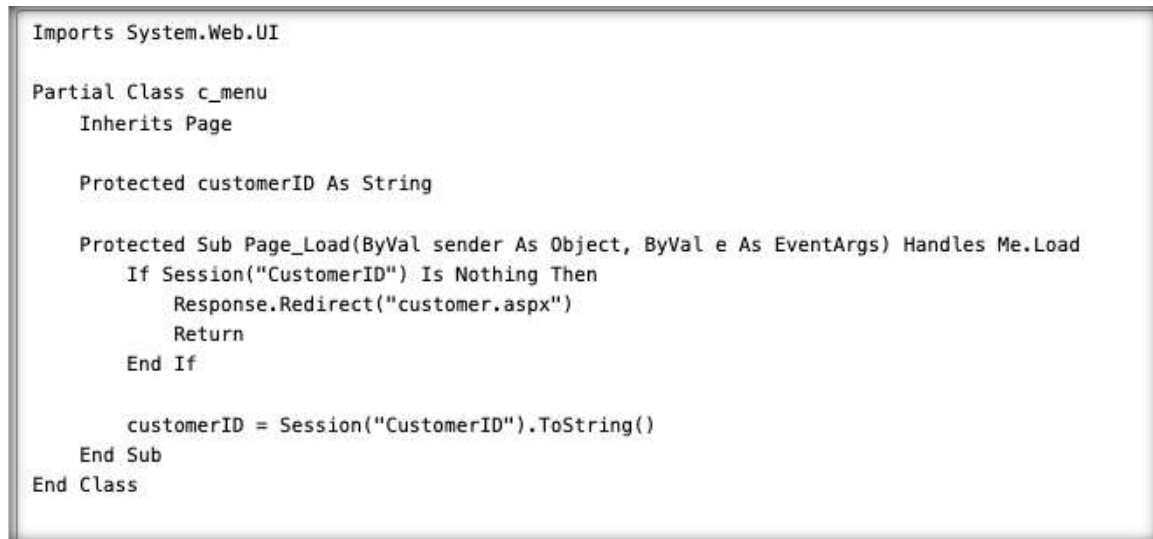


The screenshot shows a web.config file with the following XML configuration:

```
Name: web.config

<configuration>
  <system.web>
    <customErrors mode="Off"/>
    <compilation debug="true"/>
    <sessionState timeout="5" />
  </system.web>
</configuration>
```

Below is the screenshot of code behind for one of the pages where session timeout is checked, if it is timed out, redirected to login page to login again.



The screenshot shows the code behind for a page, with the following C# code:

```
Imports System.Web.UI

Partial Class c_menu
  Inherits Page

  Protected customerID As String

  Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs) Handles Me.Load
    If Session("CustomerID") Is Nothing Then
      Response.Redirect("customer.aspx")
      Return
    End If

    customerID = Session("CustomerID").ToString()
  End Sub
End Class
```

3. Cookie Invalidation for Admin Pages (5 Minutes):

Cookie-based timeout system has been implemented for admin pages. Admins are automatically logged out if no activity is detected for 5 minutes. This method invalidates the authentication cookie, ensuring that privileged access is not misused after a session is left idle.

Here's how I have stored cookies and set their timeout to be 5 minutes.

```
Protected Sub btnClick_Me(ByVal sender as Object, ByVal e As EventArgs)
    Dim username As String =txtUserName.Text
    Dim password As String =txtPassword.Text
    If username = "Admin" AND password = "1234" Then
        Dim Cookie As New HttpCookie("Admin_Info")
        Cookie("Admin_Name")=username
        Cookie("Admin_Password")=password
        Cookie.Expires=DateTime.Now.AddMinutes(5)
        Response.Cookies.Add(Cookie)

        Response.Redirect("a_menu.aspx")
    Else
        lblMessage.Text="Invalid Credentials"
        lblMessage.ForeColor=Drawing.Color.Red
    End If
```

Here's how I have retrieved them.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs) Handles Me.Load

    If Not IsPostBack Then
        Dim Cookie As HttpCookie
        Cookie=Request.Cookies("Admin_Info")
        If Cookie IsNot Nothing Then
            Cookie=Request.Cookies("Admin_Info")
            If Request.Cookies("MethodID") IsNot Nothing Then
                txtMethodId.Text = Request.Cookies("MethodID").Value
            End If
            If Request.Cookies("MethodName") IsNot Nothing Then
                txtMethodName.Text = Request.Cookies("MethodName").Value
            End If
        Else
            Response.Redirect("admin.aspx")
        End If
    End If
End Sub
```

4. Parameterized SQL Queries

All database interactions use parameterized SQL queries. This prevents SQL Injection attacks, one of the most common security vulnerabilities in web applications. Instead of inserting raw user input directly into SQL statements, parameters are bound securely, protecting the database against manipulation.

Here is one sample chunk of code where I have applied the concept of parameterized SQL queries to prevent SQL injection:

```
Dim cmd As New SqlCommand("INSERT INTO PAYMENTMETHOD (Method_ID, MethodName) VALUES (@MethodID, @MethodName)", con)
cmd.Parameters.AddWithValue("@MethodID", txtMethodId.Text)
cmd.Parameters.AddWithValue("@MethodName", txtMethodName.Text)

Try
    con.Open()
    cmd.ExecuteNonQuery()
    lblMessage.ForeColor = Drawing.Color.Green
    lblMessage.Text = "Method Added"
```

Note: This format of parameterized SQL queries is used for all databases' codes in almost all pages.

5. Input Validation & Strong Password Enforcement

Input fields of **SIGN UP PAGE** are validated using ASP.NET validators:

- Regular Expression Validator ensures strong password patterns (e.g., length and complexity).

Additionally,

- Required Field Validator ensures critical fields are not left empty.
- Format Validators ensure that the email and phone numbers conform to standard formats.

Below is the screenshot of password length and strength validation code along with email and phone number formats validation codes.

```
Password: <br/>
<asp:TextBox ID="txtCustomerId" runat="server"></asp:TextBox><br />
<asp:RequiredFieldValidator runat="server" ControlToValidate="txtCustomerId" ErrorMessage="Password is required!" ForeColor="Red"
Display="Dynamic" /><br />
<asp:RegularExpressionValidator runat="server" ControlToValidate="txtCustomerId" ValidationExpression="^(?=.*[A-Za-z])(?=.*\d).{8,}$"
ErrorMessage="Password must be 8+ characters with at least 1 letter and 1 digit" ForeColor="Red" Display="Dynamic" />

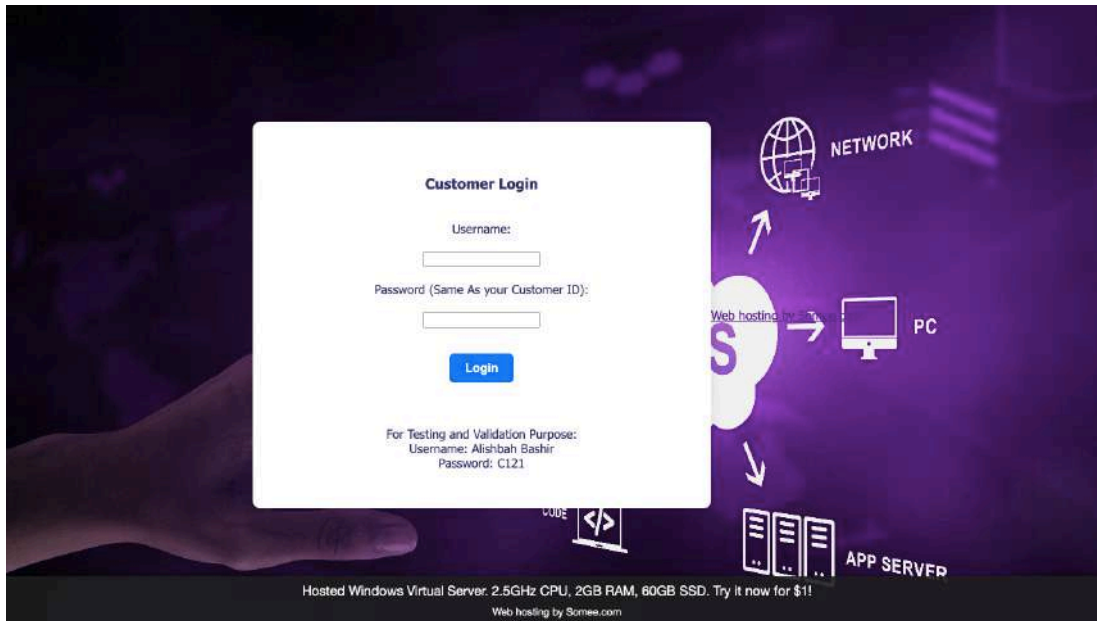
Email: <br/>
<asp:TextBox ID="txtEmail" runat="server"></asp:TextBox><br />
<asp:RequiredFieldValidator runat="server" ControlToValidate="txtEmail" ErrorMessage="Email is required!" ForeColor="Red"
Display="Dynamic" /><br />
<asp:RegularExpressionValidator runat="server" ControlToValidate="txtEmail" ValidationExpression="\S+@\S+\.\S+" ErrorMessage="Invalid
email!" ForeColor="Red" /><br/>

Cell No: <br/>
<asp:TextBox ID="txtCellNo" runat="server"></asp:TextBox><br />
<asp:RequiredFieldValidator runat="server" ControlToValidate="txtCellNo" ErrorMessage="Cell Number is required!" ForeColor="Red"
Display="Dynamic" /><br />
<asp:RegularExpressionValidator runat="server" ControlToValidate="txtCellNo" ValidationExpression="^\+92[0-9]{10}$"
ErrorMessage="Enter a valid Pakistan number (e.g., +923001234567)" ForeColor="Red" Display="Dynamic" /><br/>
```

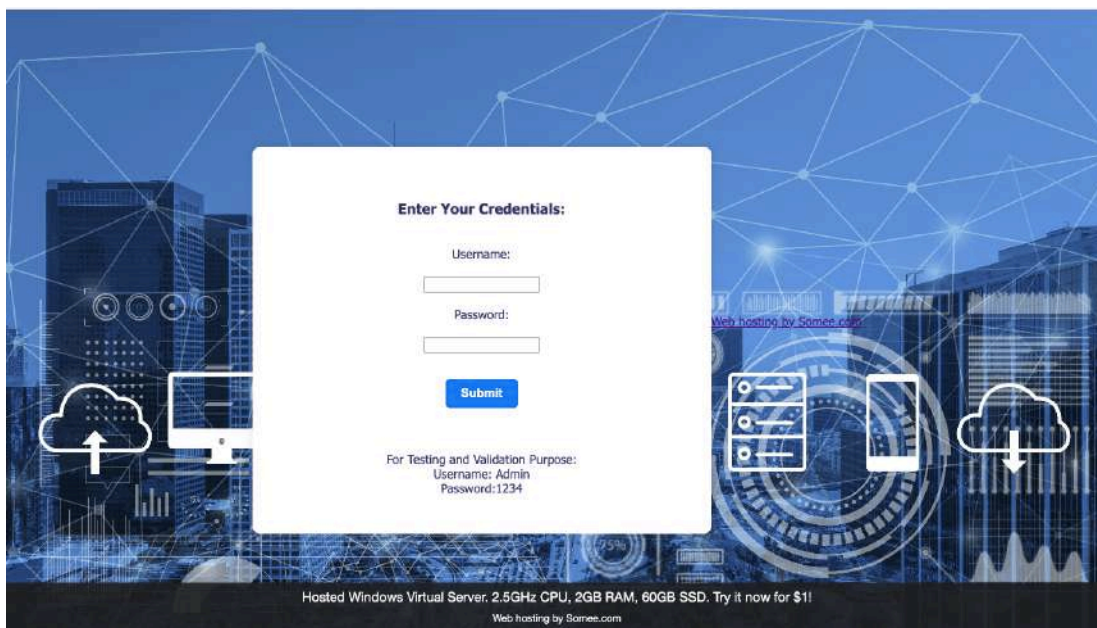
6. Role-Based Access Control (RBAC):

RBAC ensures that users can only access functionality appropriate to their role. Two roles have been defined: Admin and Customer. Admins have access to backend and management features, while customers can only interact with subscription and software tool features relevant to them. This prevents unauthorized actions and enhances data isolation.

For Customers:



For Admin:



7. Encrypted ViewState

In ASP.NET, ViewState is used to preserve page and control values between postbacks. In this project, the `EnableViewStateMac` and `ViewStateEncryptionMode` directives are configured in all pages to encrypt the ViewState. This protects the application against ViewState tampering and disclosure of sensitive page-level data.

Name: a_menu.aspx

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="a_menu.aspx.vb" Inherits="a_menu" EnableViewState="true" ViewStateEncryptionMode="Always"%>
```

8. SSL Certificate (HTTPS):

An SSL certificate has been purchased and installed to enforce HTTPS across the platform. This encrypts the data transmission between the user and the server, protecting sensitive data (like login credentials and payment details) from being intercepted or tampered with by third parties.