IAD

PROBLEM 1: SECURITY FEATURES OF MOVIE RENTAL SYSTEM

inayat arshad
BS-22-IB-105732

SECURITY FEATURES USED IN MOVIE RENTAL SYSTEM

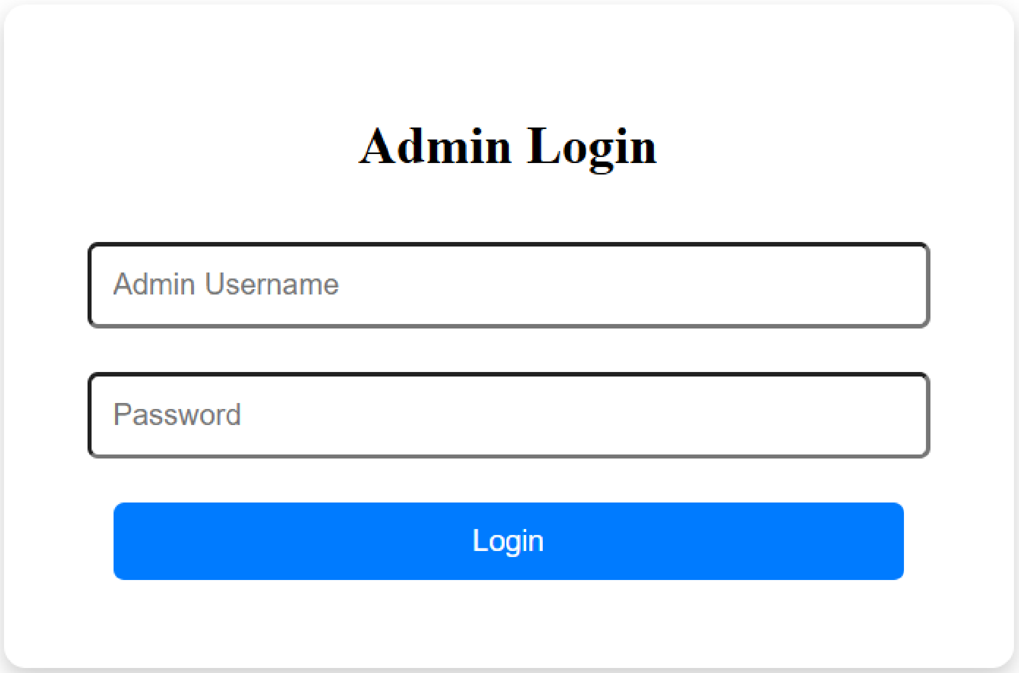
INTRODUCTION

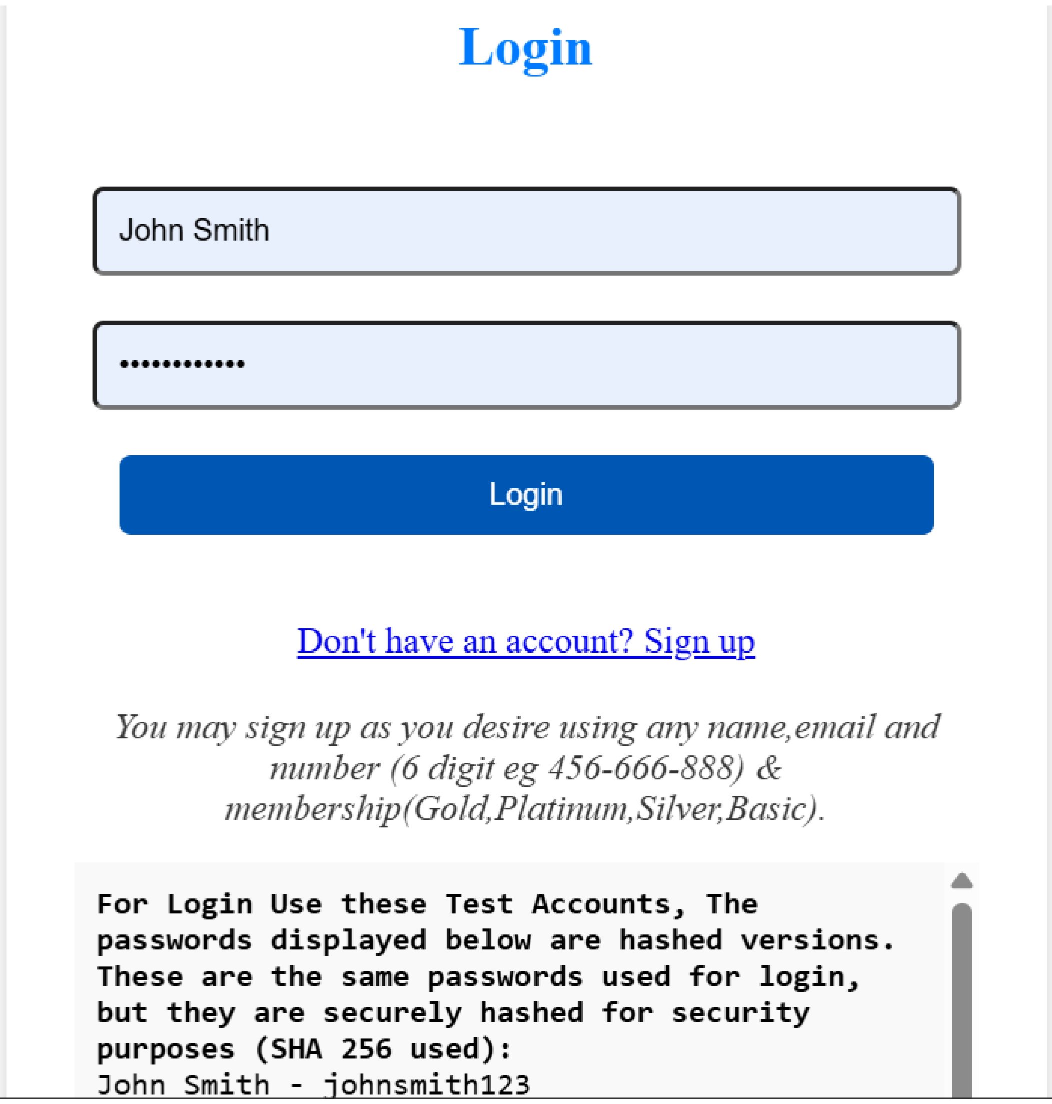
In this report, important security features have been implemented to protect user data , accounts and access to different functionalities.

Security Features	***Functionality***
User Authentication	Verify user identity using username and password.
Cookie Invalidation (Admin)	Invalidate admin session cookies after 5 minutes of inactivity
Role-Based Access Control	Restrict access based on roles (Admin/Customer).
Secure Password Storage	Hashing passwords to prevent attacks .
Session Timeout (Customer)	Terminate inactive customer sessions after 20 minutes.
Parametrized Queries	Prevent SQL Injection attacks.
Encrypted View state	Protect page state data from tampering.
SSl usage	Encrypt communication between client and server.

1)USER AUTHENTICATION:

Authentication guarantees that only legitimate users can access the application. A login mechanism based on usernames and passwords has been incorporated, where user credentials are validated against the stored database records before access is granted. This process serves as a critical security layer, helping to block unauthorized users from entering the system.


Admin Login
Admin Username
Password
Login

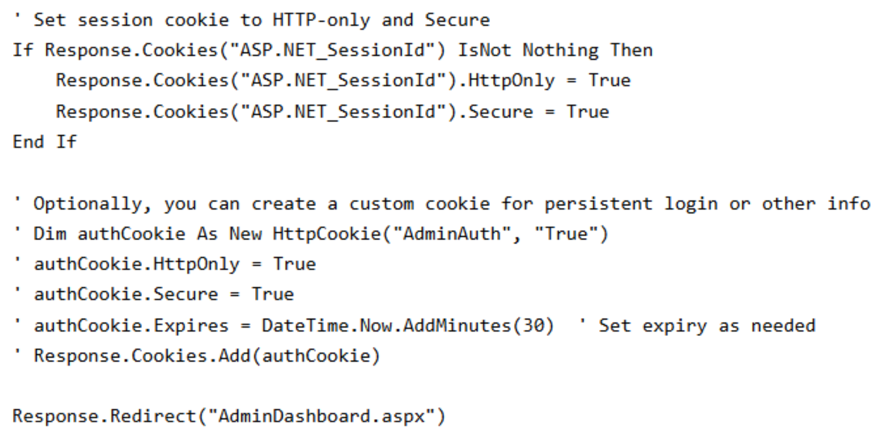


You may sign up as you desire using any name,email and number (6 digit eg 456-666-888) & membership(Gold,Platinum,Silver,Basic).

```
For Login Use these Test Accounts, The
passwords displayed below are hashed versions.
These are the same passwords used for login,
but they are securely hashed for security
purposes (SHA 256 used):
John Smith - johnsmith123
```

2)COOKIE INVALIDATION(ADMIN):

Cookie invalidation enhances security by ensuring that admin session cookies expire after a period of inactivity. In this application, admin cookies are automatically invalidated after 20 minutes of no activity. This prevents unauthorized access if the admin leaves the session unattended. It helps reduce the risk of session hijacking.



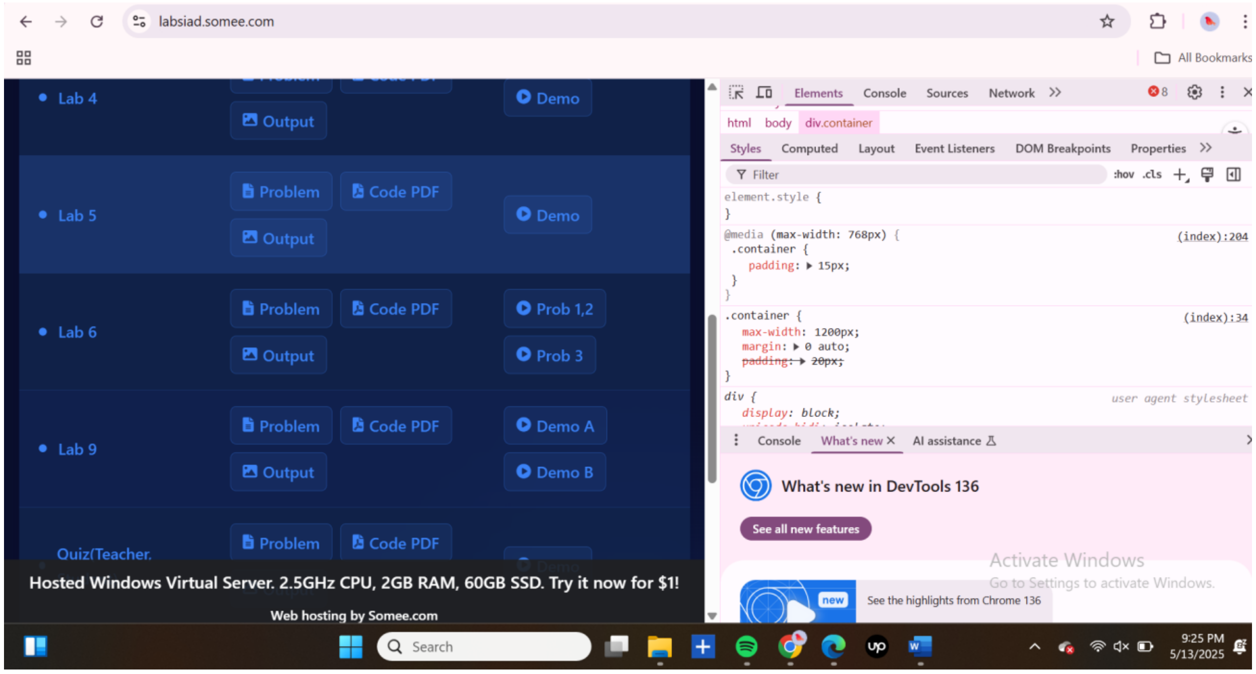
```
' Set session cookie to HTTP-only and Secure
If Response.Cookies("ASP.NET_SessionId") IsNot Nothing Then
    Response.Cookies("ASP.NET_SessionId").HttpOnly = True
    Response.Cookies("ASP.NET_SessionId").Secure = True
End If

' Optionally, you can create a custom cookie for persistent login or other info
' Dim authCookie As New HttpCookie("AdminAuth", "True")
' authCookie.HttpOnly = True
' authCookie.Secure = True
' authCookie.Expires = DateTime.Now.AddMinutes(30)  ' Set expiry as needed
' Response.Cookies.Add(authCookie)

Response.Redirect("AdminDashboard.aspx")
```

We can check cookies on our page by ;

Testing Secure Cookies:

- **Step 1:** Log in to the application using valid credentials.
- **Step 2:** Open your browser's developer tools (right-click > Inspect > Application tab).
- **Step 3:** Look for the session cookie (ASP.NET_SessionId) under the "Cookies" section.
- **What happens?** The session cookie should have both the HttpOnly and Secure flags enabled, ensuring that the cookie is not accessible via JavaScript and is sent only over HTTPS.

PROBLEM 1: SECURITY FEATURES OF MOVIE RENTAL SYSTEM

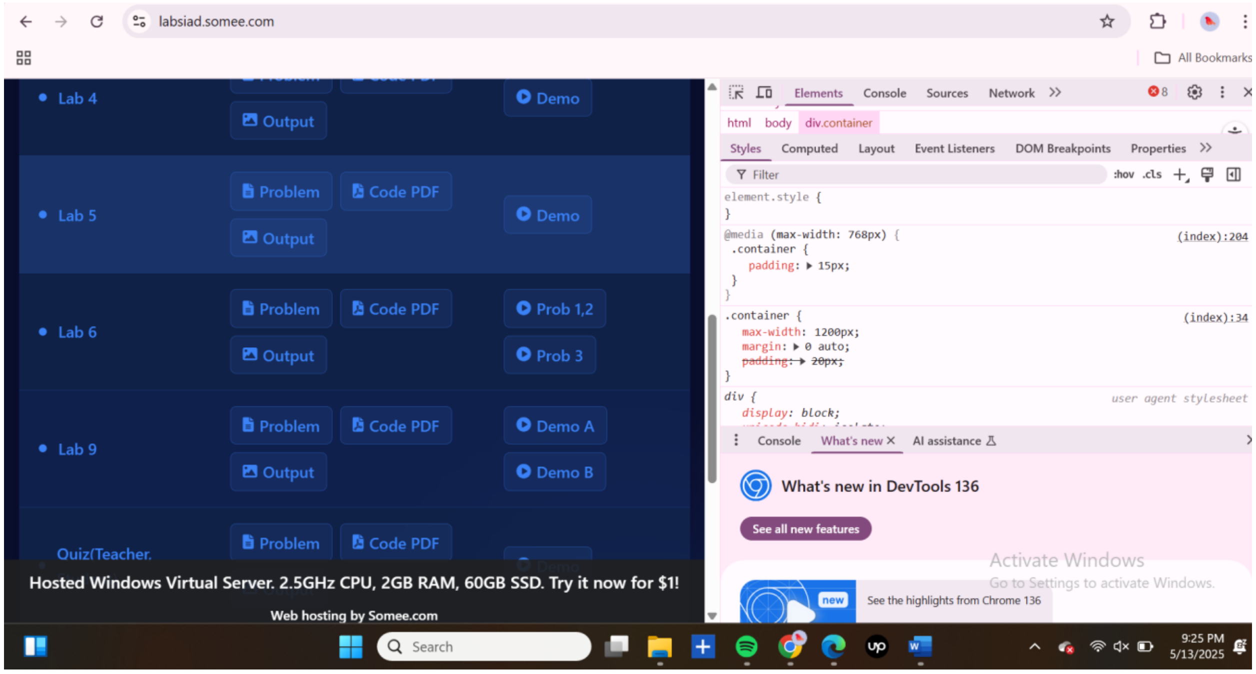


Hosted Windows Virtual Server. 2.5GHz CPU, 2GB RAM, 60GB SSD. Try it now for $1!

Web hosting by Somee.com

```
element.style {
}
@media (max-width: 768px) {                    (index):204
  .container {
    padding: ▶ 15px;
  }
}
.container {                                   (index):34
    max-width: 1200px;
    margin: ▶ 0 auto;
    ~~padding: ▶ 20px;~~
}
div {                           user agent stylesheet
    display: block;
```

What's new in DevTools 136

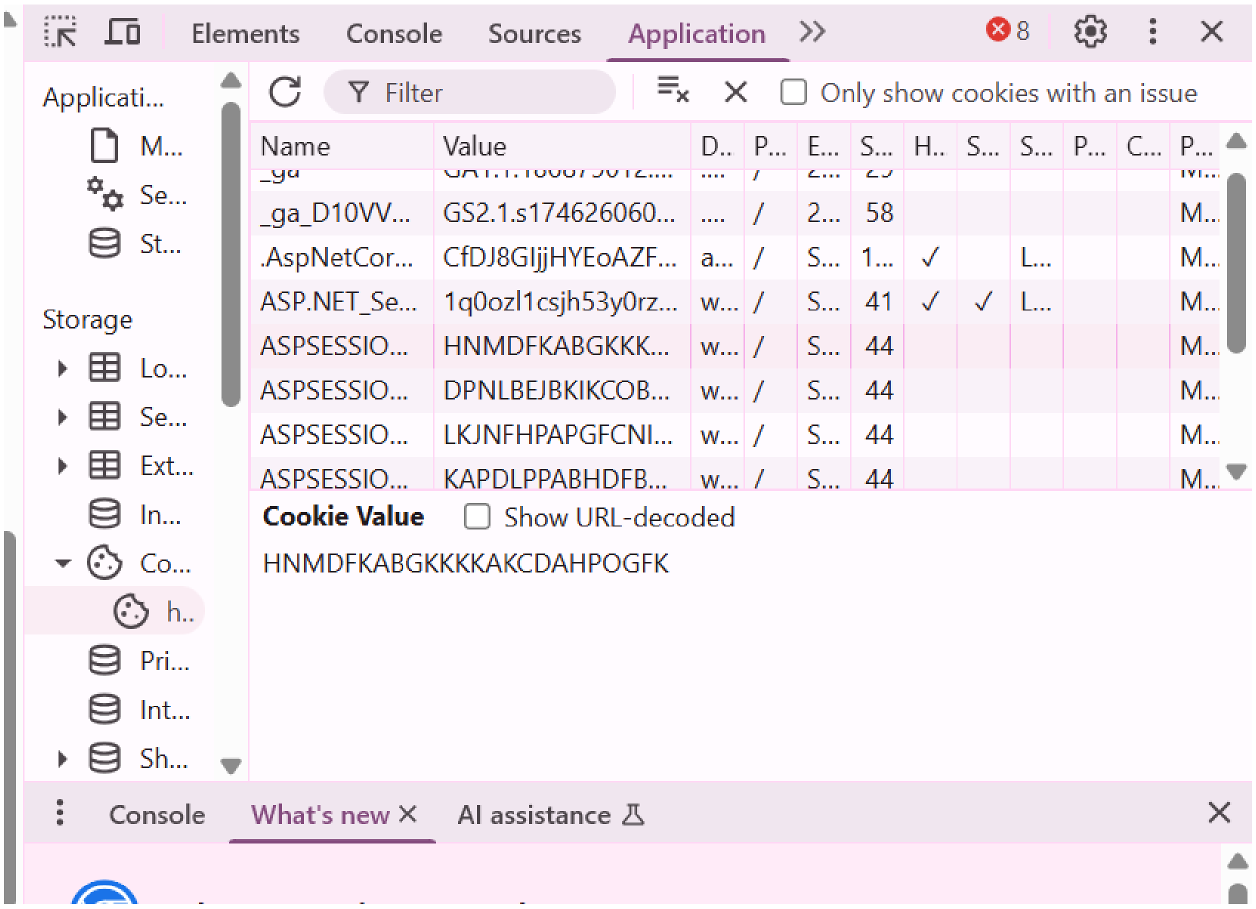


Hosted Windows Virtual Server. 2.5GHz CPU, 2GB RAM, 60GB SSD. Try it now for $1!

Web hosting by Somee.com

```
element.style {
}
@media (max-width: 768px) {                    (index):204
  .container {
    padding: ▶ 15px;
  }
}
.container {                                   (index):34
    max-width: 1200px;
    margin: ▶ 0 auto;
    ~~padding: ▶ 20px;~~
}
div {                           user agent stylesheet
    display: block;
```

What's new in DevTools 136



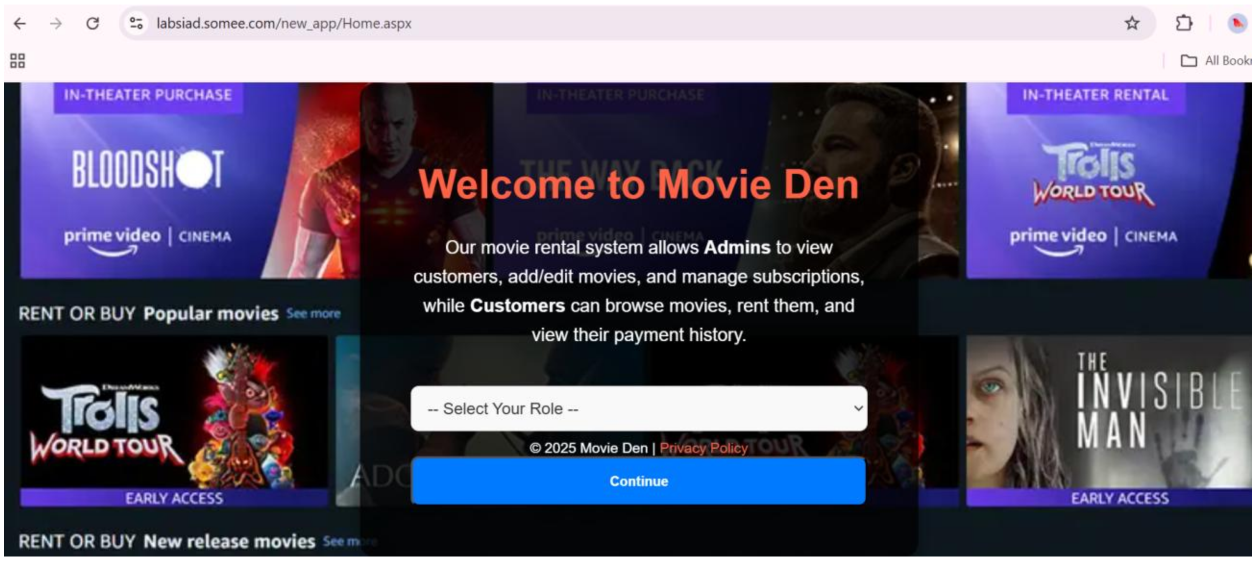
Name	Value	D..	P...	E...	S...	H..	S...	S...	P...	C...	P...
_ga_D10VV...	GS2.1.s174626060...	....	/	2...	58						M...
.AspNetCor...	CfDJ8GIjjHYEoAZF...	a...	/	S...	1...	✓		L...			M...
ASP.NET_Se...	1q0ozl1csjh53y0rz...	w...	/	S...	41	✓	✓	L...			M...
ASPSESSIO...	HNMDFKABGKKK...	w...	/	S...	44						M...
ASPSESSIO...	DPNLBEJBKIKCOB...	w...	/	S...	44						M...
ASPSESSIO...	LKJNFHPAPGFCNI...	w...	/	S...	44						M...
ASPSESSIO...	KAPDLPPABHDFB...	w...	/	S...	44						M...

Cookie Value Show URL-decoded

HNMDFKABGKKKKAKCDAHPOGFK

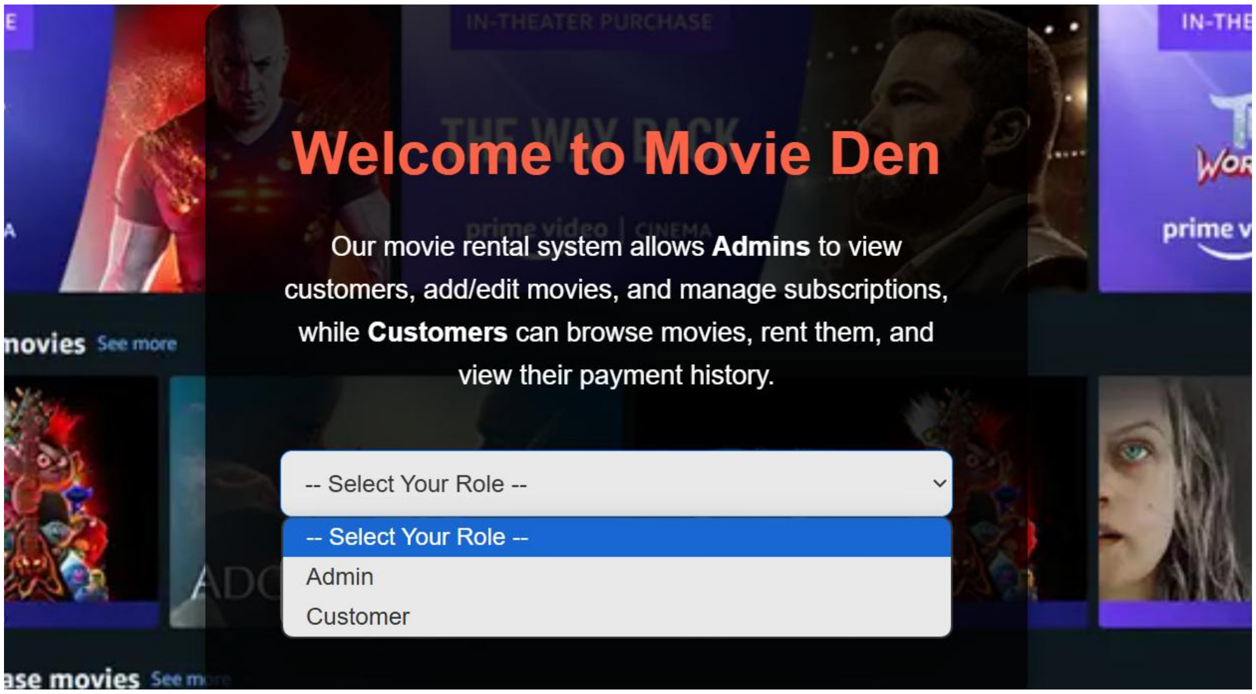
3)RBAC(Role based Access Control)

Role-Based Access Control (RBAC) restricts access to different parts of the application based on user roles. In this system, Admins can manage movies, customers, and subscriptions, while Customers can only browse, rent movies, and view payment history. This ensures users access only the features relevant to their role. RBAC enhances security and organizes functionality efficiently.


labsiad.somee.com/new_app/Home.aspx

Welcome to Movie Den

Our movie rental system allows **Admins** to view customers, add/edit movies, and manage subscriptions, while **Customers** can browse movies, rent them, and view their payment history.


Welcome to Movie Den

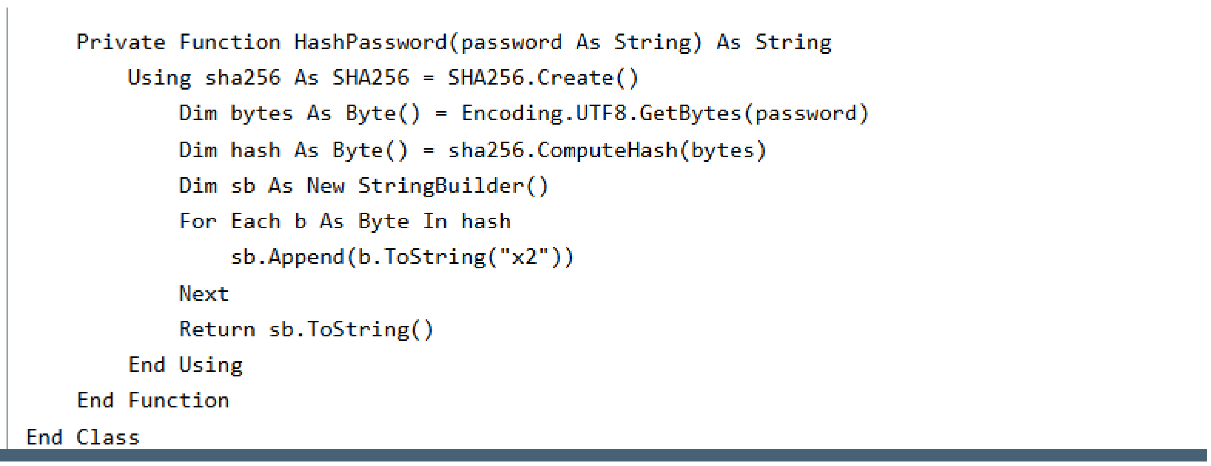
Our movie rental system allows **Admins** to view customers, add/edit movies, and manage subscriptions, while **Customers** can browse movies, rent them, and view their payment history.

4)SECURE PASSWORD STORAGE(sha 256 used)

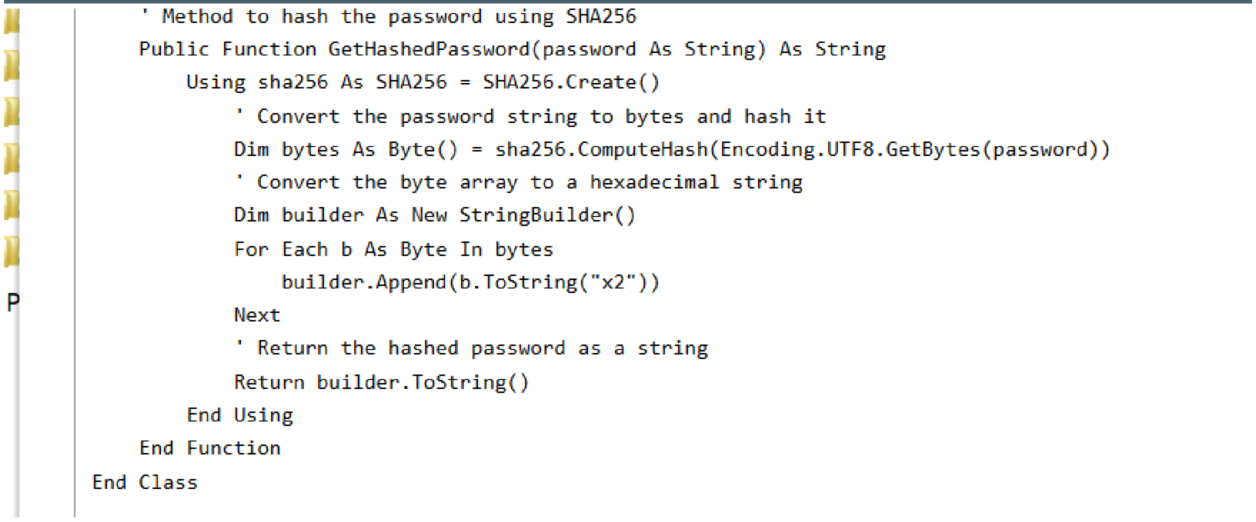
When a user types a password like inayat123 during login:

1. Login.aspx.vb **hashes** that input using SHA-256.
2. The hashed version (e.g., 25f9e794323b453885f5181f1b624d0b) is compared with the **stored hashed password** in the database.
3. If it **matches**, login is successful.

You just type the plain password (inayat123) as usual — no change in user behavior.



```
    Private Function HashPassword(password As String) As String
        Using sha256 As SHA256 = SHA256.Create()
            Dim bytes As Byte() = Encoding.UTF8.GetBytes(password)
            Dim hash As Byte() = sha256.ComputeHash(bytes)
            Dim sb As New StringBuilder()
            For Each b As Byte In hash
                sb.Append(b.ToString("x2"))
            Next
            Return sb.ToString()
        End Using
    End Function
End Class
```



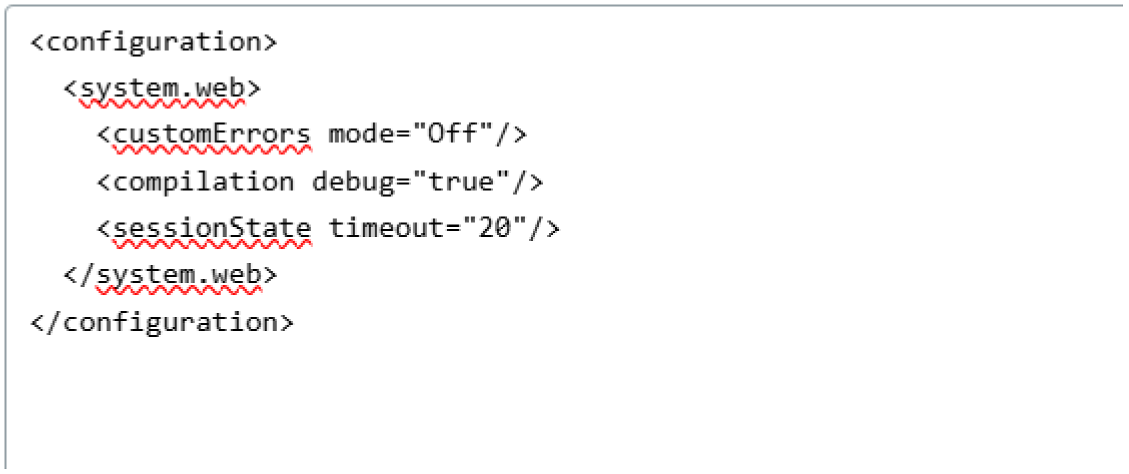
```
    ' Method to hash the password using SHA256
    Public Function GetHashedPassword(password As String) As String
        Using sha256 As SHA256 = SHA256.Create()
            ' Convert the password string to bytes and hash it
            Dim bytes As Byte() = sha256.ComputeHash(Encoding.UTF8.GetBytes(password))
            ' Convert the byte array to a hexadecimal string
            Dim builder As New StringBuilder()
            For Each b As Byte In bytes
                builder.Append(b.ToString("x2"))
            Next
            ' Return the hashed password as a string
            Return builder.ToString()
        End Using
    End Function
End Class
```

After adding HashPasswords.aspx ,I Deployed and visit https://labsiad.somee.com/HashPasswords.aspx once, It will hash all non-hashed passwords in the customer_t table. After confirming login works:
Delete the page immediately
In my web application, I implemented **password hashing** to enhance the security of the login system. Specifically, I used the **SHA-256 hashing algorithm** in VB.NET to hash the admin password before verifying it during login.
This had the following functions and advantages:

- **Prevented plain-text password storage**: Instead of storing or comparing passwords in plain text, I used a hashing function to transform the password into a fixed-length, irreversible string.
- **Improved security in case of data breaches**: Even if someone gains unauthorized access to the application or database, they won't be able to see the actual password.
- **Resisted reverse-engineering**: The one-way nature of the SHA-256 algorithm ensures that it's practically impossible to retrieve the original password from the hashed version.
- **Protected against common attacks**: Hashing helps defend against **brute force** and **rainbow table attacks**, especially when used with additional techniques like salting (which can be added later for even stronger protection).

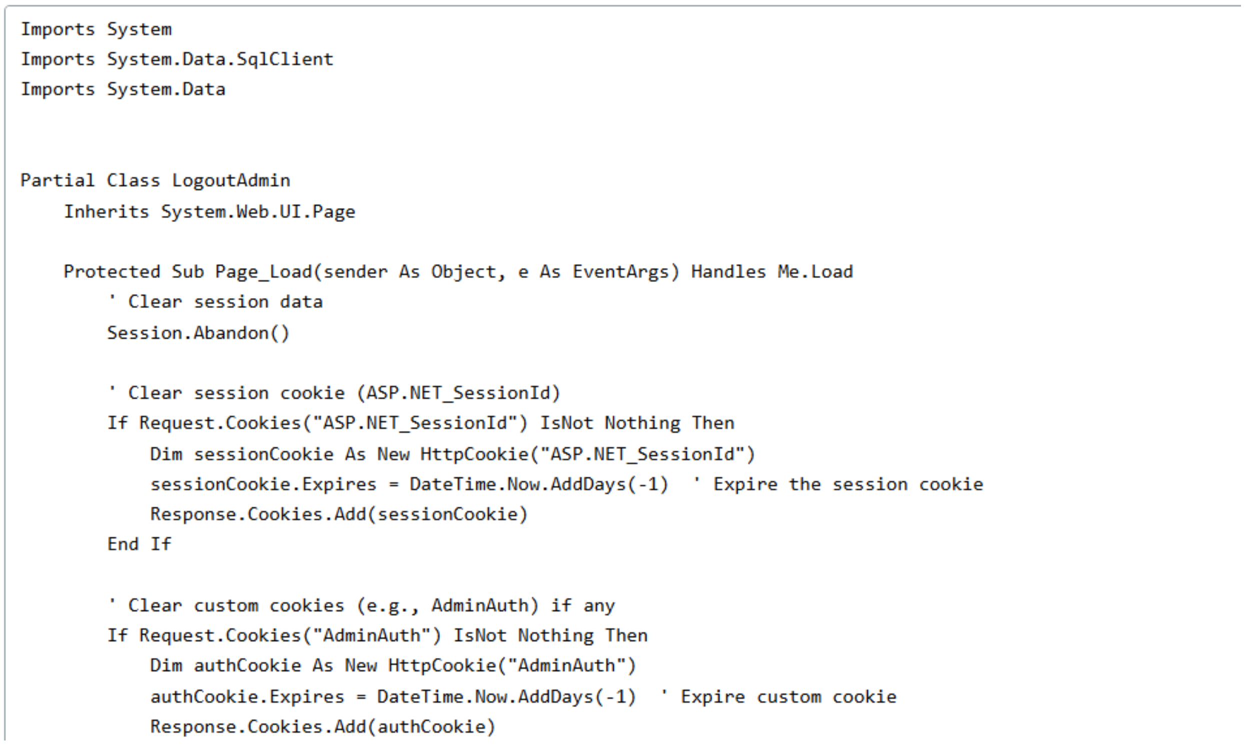
By doing this, I ensured that sensitive login information, like admin credentials, is handled in a secure and industry-standard way.

5)SESSION TIMEOUT(20 min)



```
<configuration>
  <system.web>
    <customErrors mode="Off"/>
    <compilation debug="true"/>
    <sessionState timeout="20"/>
  </system.web>
</configuration>
```

As soon as timeout , logout page redirects to home .



```
Imports System
Imports System.Data.SqlClient
Imports System.Data


Partial Class LogoutAdmin
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(sender As Object, e As EventArgs) Handles Me.Load
        ' Clear session data
        Session.Abandon()

        ' Clear session cookie (ASP.NET_SessionId)
        If Request.Cookies("ASP.NET_SessionId") IsNot Nothing Then
            Dim sessionCookie As New HttpCookie("ASP.NET_SessionId")
            sessionCookie.Expires = DateTime.Now.AddDays(-1)  ' Expire the session cookie
            Response.Cookies.Add(sessionCookie)
        End If

        ' Clear custom cookies (e.g., AdminAuth) if any
        If Request.Cookies("AdminAuth") IsNot Nothing Then
            Dim authCookie As New HttpCookie("AdminAuth")
            authCookie.Expires = DateTime.Now.AddDays(-1)  ' Expire custom cookie
            Response.Cookies.Add(authCookie)
```

Session abandon() functionality used.

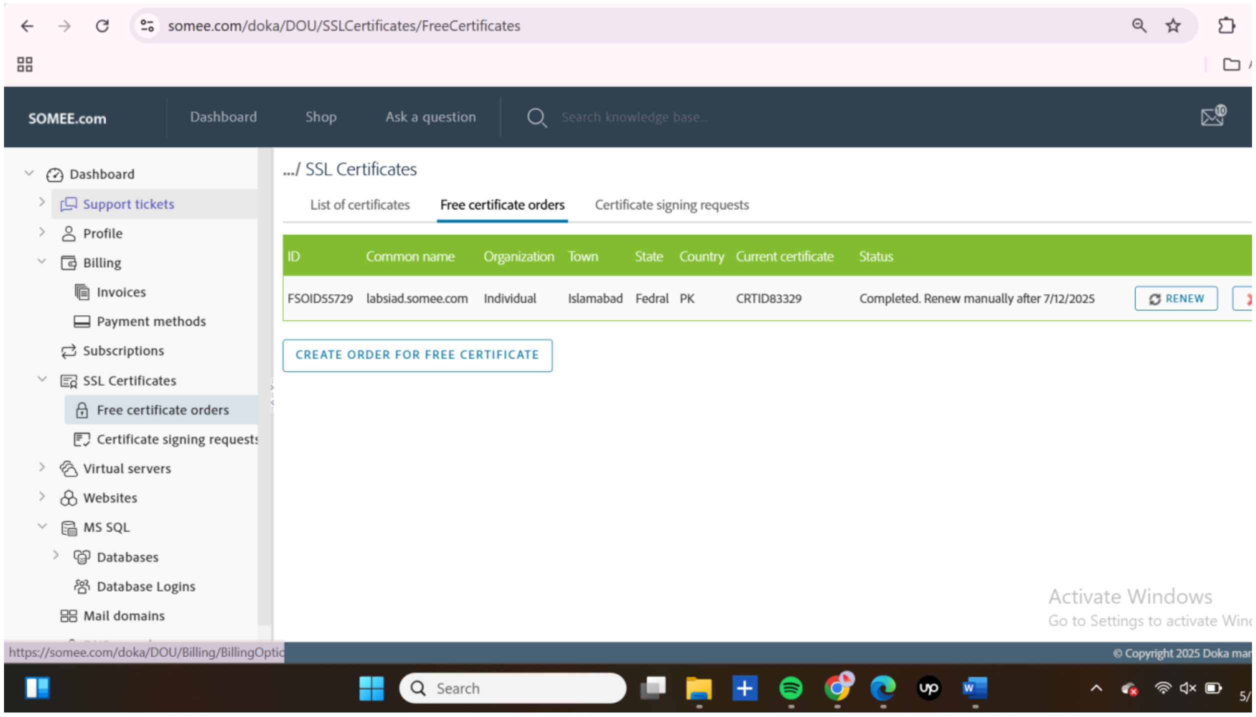
6)PARAMETRIZED SQL QUERY

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.

```
        Using conn As New SqlConnection(connStr)
            conn.Open()

            Dim cmd As New SqlCommand("SELECT customer_id, customer_name FROM customer_t WHERE customer_name = @name AND password = @password",
conn)
            cmd.Parameters.AddWithValue("@name", txtUsername.Text)
            cmd.Parameters.AddWithValue("@password", hashedInputPassword)

            Dim reader As SqlDataReader = cmd.ExecuteReader()
```

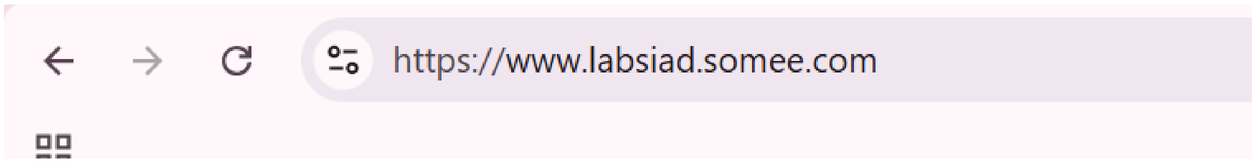
7)SSL USED

To enhance the security of my ASP.NET web application, I implemented HTTPS redirection to ensure that all client-server communication is encrypted. Firstly , my site was not secured due to http only , no SSL , the URL was http://labsiad.somee.com but after that I bought a free certificate order from some ,



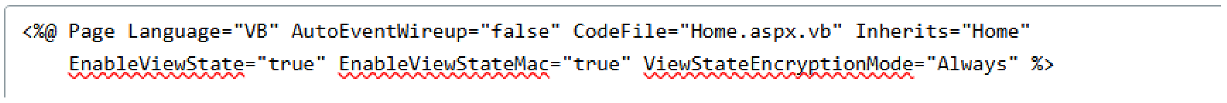
ID	Common name	Organization	Town	State	Country	Current certificate	Status
FSOID55729	labsiad.somee.com	Individual	Islamabad	Fedral	PK	CRTID83329	Completed. Renew manually after 7/12/2025

And then I was able to , use Https and system got secured , now my URL is https://www.labsiad.somee.com/



8)ENCRYPTED VIEW STATE

In this ASP.NET project, ViewState is utilized to retain control and page data during postbacks. To enhance security, each page is configured with EnableViewStateMac and ViewStateEncryptionMode settings, ensuring that the ViewState is encrypted and tamper-proof. This prevents attackers from modifying or accessing sensitive data stored in the ViewState. By securing ViewState, the application maintains integrity and confidentiality across user interactions.



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

----------------------end----------------------