SUBHAN AMJAD

IAD LAB 13

To

Mr. IRFAN HAMEED

SECURITY FEATURES IMPLEMENTED

1	Password hashing with salt (PBKDF2)	Implemented
2	Secure session timeout	Implemented
3	Http Only and Secure cookies	Implemented
4	Parameterized queries (SQLi prevention)	Implemented
5	Login audit logging	Implemented
6	Role-based access foundation	Implemented
7	User validity with unique username check	Implemented

➢ Password Hashing with Salt using PBKDF2

Both the *Login* and *AddUser* pages use PBKDF2 (Rfc2898DeriveBytes) to hash passwords. Unique 128-bit salts are generated for each user using a secure RNG (RNGCryptoServiceProvider). Hashing is done with 10,000 iterations, which significantly slows down brute force and rainbow table attacks.

- Prevents password theft even if the database is compromised.
- Resistant to common attacks like brute force, dictionary attacks, and precomputed hashes.

➢ SESSION MANAGEMENT

- **Implementation Details:**
  - The web.config file sets the session timeout to 5 minutes:

```
<!-- Session Timeout -->
<sessionState timeout="5" />
```

- **Security Impact:**
  - Limits the time window for session hijacking.
  - Enforces automatic logout after inactivity, reducing unauthorized access risks.

➢ **SECURE AND HttpOnly Cookies**

```
<!-- Secure and HttpOnly Cookies -->
<httpCookies httpOnlyCookies="true" requireSSL="true" />
```

- **Security Impact:**
  - HttpOnly prevents JavaScript from accessing cookies, mitigating **XSS attacks**.
  - RequireSSL ensures cookies are sent only over secure HTTPS connections, protecting them from being intercepted.

➢ **SQL INJECTION PREVENTION VIA PARAMETRIZED QUERIES**

- **Implementation Details:**
  - All SQL queries use SqlCommand with Parameters.AddWithValue() or .Add():

```
Dim cmd As New SqlCommand("SELECT User_ID, Password_Salt, Password_Hash FROM User_t WHERE Username=@uname", conn)
cmd.Parameters.AddWithValue("@uname", username)
```

- **Security Impact:**
  - Effectively blocks SQL injection attacks, as user input is not directly injected into SQL queries.

➢ **AUDIT LOGGING FOR LOGIN ATTEMPTS**

- **Implementation Details:**
  - Function RecordLogin logs:

User ID	IP Address	User Agent	Login Time	Login Status

- **Security Impact:**
  - Provides traceability and helps in intrusion detection.
  - Useful for security audits and account breach investigations.

- **Role-Based Access Preparation (RBAC)**

  - **Implementation Details:**
    - Role ID is retrieved after successful login.
    - Session variable Session("RoleID") is set.
    - Role-based redirection is applied:

```
' Example role redirection (customize this)
If Session("RoleID").ToString() = "1" Then
    Response.Redirect("Dashboard.aspx")
Else
    Response.Redirect("Dashboard.aspx")
End If
```

  - **Security Impact:**
    - Foundation for implementing **role-based access control (RBAC)**, ensuring users access only authorized areas.

- **USER VALIDITY AND UNIQUE USERNAME ENFOCEMENT**

- **Implementation Details:**
  - Before creating a new user, the system checks if the username already exists in the database:
    - If a user with the same username exists, it prevents duplicate creation and shows a relevant message.
  - This ensures username uniqueness at the database level, reducing data conflicts and improving account integrity.

```
' Check if username already exists
Dim checkCmd As New SqlCommand("SELECT COUNT(*) FROM User_t WHERE Username = @username", conn)
checkCmd.Parameters.Add("@username", SqlDbType.NVarChar).Value = username
```

- **Security and Functional Impact:**
    - Prevents account spoofing or confusion between users.
    - Strengthens identity verification and maintains consistency across login and access control.
    - Lays the foundation for clean and secure user management in multi-role environments.