DEPARTMENT OF COMPUTER & INFORMATION SCIENCES

INTERNET APPLICATION DEVELOPMENT LAB 13

MUHAMMAD ALI QADRI

TABLE OF CONTENTS

PROBLEM 01 (SECURITY FEATURES) 1

Password Hashing 1

Purpose: 1

Implementation: 1

Example: 1

Benefit: 1

Role-Based Access Control (RBAC) 1

Purpose: 1

Implementation: 1

Example: 1

Benefit: 1

Input Validation (Client-side & Server-side) 2

Purpose: 2

Implementation: 2

Example: 2

Benefit: 2

Session Management 2

Purpose: 2

Implementation: 2

Example: 2

Benefit: 2

SQL Injection Protection 3

Purpose: 3

Implementation: 3

Example: 3

Benefit: 3

Access Control to Admin Panel 3

Purpose: 3

Implementation: 3

Example: 3

Benefit: 3

Secure Logout Mechanism 4

Purpose: 4

Implementation: 4

Example: 4

Benefit: 4

Summary Table 4

PROBLEM 03 (TEST CASES) 5
Password Hashing – Test Cases 5
Positive Case Input: 5
Negative Case 5
Role-Based Access Control (RBAC) 6
Positive Case 6
Negative Case 6
Input Validation 6
Positive Case 6
Negative Case 6
Session Management 7
Positive Case 7
Negative Case 7
SQL Injection Protection 7
Positive Case 7
Negative Case 7
Admin Panel Access Control 8
Positive Case 8
Negative Case 8
Secure Logout 8
Positive Case 8
Negative Case 8

PROBLEM 01 (SECURITY FEATURES)

Password Hashing

Purpose:

To protect user passwords from being stolen, especially in case of database breach.

Implementation:

- When a user registers or updates their password, the password is not stored as plain text.
- Instead, it's hashed using SHA-256 a secure hashing algorithm.
- During login, the password entered is hashed and compared with the stored hash.

Example:

```
Dim hashedPassword As String = ComputeSHA256Hash(txtPassword.Text)

' Save hashedPassword to database
```

Benefit:

Even if an attacker gains access to the database, they cannot see the actual passwords.

Role-Based Access Control (RBAC)

Purpose:

To ensure that different types of users (Admin, Receptionist, Patient, Doctor) only access the data and features they're allowed to.

Implementation:

- After login, the user's role is stored in session.
- Each page or feature checks the user's role before granting access.
- UI elements are also shown/hidden based on role.

Example:

```
If Session("role") <> "Admin" Then

Response.Redirect("Unauthorized.aspx")

End If
```

Benefit:

Prevents unauthorized users from accessing sensitive operations (e.g., only Admin can manage users).

Input Validation (Client-side & Server-side)

Purpose:

To prevent invalid or malicious data from entering the system.

Implementation:

- **Client-side** validation using JavaScript and ASP.NET Validators.
- **Server-side** validation in VB.NET code-behind to double-check input before inserting into the database.

Example:

- Use RequiredFieldValidator, RegularExpressionValidator
- In server-side code:

```
If txtEmail.Text = "" Or Not txtEmail.Text.Contains("@") Then
lblError.Text = "Invalid Email"
End If
```

Benefit:

Prevents data inconsistency, accidental errors, and some forms of injection.

Session Management

Purpose:

To manage user authentication and preserve user state across multiple pages.

Implementation:

- When the user logs in, their ID and role are stored in session variables.
- Pages validate session existence and redirect login if expired.

Example:

```
If Session("username") Is Nothing Then
Response.Redirect("Login.aspx")
End If
```

Benefit:

Keeps unauthorized users from accessing the system after logout or timeout.

SQL Injection Protection

Purpose:

To prevent attackers from manipulating SQL queries through form input.

Implementation:

- Use parameterized SQL queries instead of string concatenation.
- Avoid direct user input inside SQL strings.

Example:

```
Dim cmd As New SqlCommand("SELECT * FROM Users WHERE username = @username
AND password = @password", conn)

cmd.Parameters.AddWithValue("@username", txtUsername.Text)

cmd.Parameters.AddWithValue("@password", hashedPassword)
```

Benefit:

Prevents attacks like ' OR '1'='1 from bypassing authentication.

Access Control to Admin Panel

Purpose:

To prevent non-admin users from entering admin pages (like user management or employee addition).

Implementation:

- Check the session role before loading any sensitive pages.
- Optionally, hide the Admin navigation panel for other users.

Example:

```
If Session("role") <> "Admin" Then

Response.Redirect("NoPermission.aspx")

End If
```

Benefit:

Secures high-privilege features of your HMS from regular users or receptionists.

Secure Logout Mechanism

Purpose:

To ensure user sessions are properly terminated to prevent reuse by unauthorized parties.

Implementation:

- On logout, clear all session variables and redirect to login page.
- Prevent access to previous pages using the back button after logout.

Example:

```
Session.Clear()

Session.Abandon()

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

Benefit:

Protects against session hijacking and ensures security after logout.

Summary Table

#	Security Feature	Purpose	Key Implementation
1	Password Hashing	Protect stored passwords	Use SHA-256 or stronger hashing
2	Role-Based Access	Different rights for each user	Session-based role checks
3	Input Validation	Stop invalid/malicious input	ASP.NET + server-side checks
4	Session Management	Manage login state securely	Session timeout + page protection
5	SQL Injection Protection	Prevent SQL manipulation	Parameterized queries
6	Admin Access Control	Restrict access to admin panel	Role check in page load
7	Secure Logout	Properly terminate session	Session.Abandon() and redirect

PROBLEM 03 (TEST CASES)

Password Hashing – Test Cases

Positive Case

Input:

- A new patient registers using the HMS signup form with the following credentials:
    - Username: aliansari
    - Password: ali1223

Result*:*

- In the database, the password is stored as a hashed value (e.g., SHA-256 or salted hash).
- It is not readable and not the same as the original password.
- Login with ali1223 works correctly.
- The stored hash differs from the hash of another user using the same password as salting is applied.

Negative Case

Input*:*

- Attempts to store the password directly (plaintext) into the database (e.g., bypassing the application logic).
    - Username: attacker
    - Password: plaintext123

Result*:*

- The system automatically applies the hashing algorithm through the server-side code.
- If password is found stored as plaintext, test fails.
- The system rejects or logs any bypass attempt without hashing.
- Login using plaintext123 does not work unless it is hashed and stored properly.

Role-Based Access Control (RBAC)

Positive Case

Input:

- Log in as a Receptionist (valid credentials).
- Navigate to the Patient Management page.

Result:

- Receptionist is allowed to view and edit patient records.

Negative Case

Input:

- Log in as a Receptionist.
- Manually enter the URL for the Admin User Management page.

Result:

- Access is denied (redirected to “Access Denied” or Login page).

Input Validation

Positive Case

Input:

- In the patient registration form, enter a valid email: abc@gmail.com.
- Enter age: 45.
- Leave no required field blank.

Result:

- Form submits successfully; data is accepted.

Negative Case

Input:

- In the same form, enter email: not-an-email and age: -5.
- Try injecting script: <script>alert('X')</script> into the name field.

Result:

- Validation errors displayed: “Enter a valid email,” “Age must be between 1 and 120,” and script tags are rejected or sanitized.

Session Management

Positive Case

Input:

- Log in as any user.
- Navigate several pages within 20 minutes without logging out.

Result:

- Session remains active; user can access their dashboard without re-login.

Negative Case

Input:

- Log in, then wait 30 minutes (session timeout).
- Click the browser Back button to return to a protected page.

Result:

- User is redirected to the Login page due to the expired session.

SQL Injection Protection

Positive Case

Input:

- In the login form, enter normal credentials:
- username: aliansari
- password: ali1223

Result:

- Login succeeds or fails normally, no error.

Negative Case

Input:

- In the username field enter:
- ' OR '1'='1' --
- Any password.

Result:

- Injection attempt is neutralized; login fails with “Invalid credentials.”

Admin Panel Access Control

Positive Case

Input:

- Log in as Admin.
- Navigate to User Management page.

Result:

- Admin may view, add, edit, or delete user accounts.

Negative Case

Input:

- Log in as Doctor.
- Attempt to click “Manage Users” or enter its URL.

Result:

- Access is denied; user is redirected away with an error message.

Secure Logout

Positive Case

Input:

- Log in, then click the **Logout** button.

Result:

- Session is cleared, user is redirected to the Login page, and cannot access any protected pages without logging in again.

Negative Case

Input:

- After logging out, click the browser’s Back button to revisit a protected page.

Result:

- User remains logged out and is redirected back to the Login page (no cached content shown).

Feature	Positive Input	Positive Result	Negative Input	Negative Result
Password Hashing	Register with password ali1223	Password stored as a hash; login with ali1223 succeeds	Attempt to store or view plaintext password	System stores hash only; plaintext not accepted; login fails un-hashed
Role-Based Access Control	Receptionist logs in and navigates Patient page	Access granted	Receptionist tries to access Admin User Management URL	Access denied/redirected
Input Validation	Valid email abc@gmail.com, age 45	Form submits successfully	Email not-an-email, age -5, script tag in name field	Validation errors shown; malicious input rejected
Session Management	User navigates within session timeout period	Session remains active; pages accessible	Wait past session timeout then click Back on protected page	Redirected to Login (session expired)
SQL Injection Protection	Normal login credentials	Login process runs normally	Username ' OR '1'='1' --, any password	Injection neutralized; login fails
Admin Panel Access Control	Admin logs in and opens User Management	Admin may view/add/edit/delete users	Doctor logs in and attempts to open User Management URL	Access denied/redirected
Secure Logout	User clicks Logout	Session cleared; redirected to Login; protected pages blocked	After logout, click Back button	Remains logged out; redirected to Login; no cached protected content shown