WEB SYSTEM

Name: John Drex F. Cantor

Scenario 1 — Using $_POST instead of $_GET

☑ **Fixed Code**

```
<?php
$conn = mysqli_connect("localhost","root","","class_db");

$id = $_GET['id'];

$sql = "SELECT * FROM students WHERE student_id = $id";
$res = mysqli_query($conn, $sql);
$r = mysqli_fetch_assoc($res);

echo $r['first_name'];
?>
```

The original used $_POST even though the value came from the URL. I changed it to $_GET so the script correctly reads ?id= from the URL.

Scenario 2 — Missing Quotes in SQL

☑ **Fixed Code**

```
<?php
$conn = mysqli_connect("localhost","root","","class_db");

$fname = $_POST['fname'];
$sql = "SELECT * FROM students WHERE first_name = '$fname'";
```

```
$res = mysqli_query($conn, $sql);
?>
```

String values in SQL must be inside quotes. I added ' ' around $fname.

Scenario 3 — SQL Injection Vulnerability

☑ **Fixed Code**

```
<?php
$conn = mysqli_connect("localhost","root","","class_db");

$age = $_GET['age'];

$stmt = $conn->prepare("SELECT * FROM students WHERE age = ?");
$stmt->bind_param("i", $age);
$stmt->execute();
?>
```

I replaced the direct SQL with a prepared statement to prevent SQL injection.

Scenario 4 — Validate Empty POST Fields

☑ **Fixed Code**

```
<?php
$conn = mysqli_connect("localhost","root","","class_db");

if (!empty($_POST['fname']) && !empty($_POST['lname'])) {
```

```
    $first = $_POST['fname'];
    $last  = $_POST['lname'];

    $sql = "INSERT INTO students (first_name,last_name) VALUES
('$first','$last')";
    mysqli_query($conn, $sql);

    echo "Inserted!";

} else {
    echo "Please fill out both first and last name.";
}
?>
```

I added a check to ensure first and last name are not empty before inserting into the database.

Scenario 5 — Wrong POST Key Name

☑ **Fixed Code**

```
<?php
$conn = mysqli_connect("localhost","root","","class_db");

$email = $_POST['email'];

$sql = "SELECT * FROM students WHERE email='$email'";
$res = mysqli_query($conn, $sql);
?>
```

The original used a misspelled key emial. I changed it to email.

Scenario 6 — Unsafe DELETE Using GET

☑ **Fixed Code**

```
<?php
$conn = mysqli_connect("localhost","root","","class_db");

$id = intval($_GET['id']);

$sql = "DELETE FROM students WHERE student_id = $id";
mysqli_query($conn, $sql);
?>
```

I used intval() to prevent the user from injecting text like 1 OR 1=1.

Scenario 7 — Query Fails but Script Continues

☑ **Fixed Code**

```
<?php
$conn = mysqli_connect("localhost","root","","class_db");

$id = $_POST['id'];
$email = $_POST['email'];

$sql = "UPDATE students SET email='$email' WHERE
student_id=$id";
```

```
if (mysqli_query($conn, $sql)) {
    echo "Updated!";
} else {
    echo "Error updating!";
}
?>
```

I added error checking to only display “Updated!” when the query is successful.

Scenario 8 — Missing Loop for Fetching Multiple Records

☑ **Fixed Code**

```
<?php
$conn = mysqli_connect("localhost","root","","class_db");

$res = mysqli_query($conn, "SELECT * FROM students");

while ($row = mysqli_fetch_assoc($res)) {
    echo $row['email'] . "<br>";
}
?>
```

mysqli_fetch_assoc only returns one row, so I added a loop to display all rows.

Scenario 9 — Using POST but Link Sends GET

☑ **Fixed Code**

```
<?php
```

```
$id = $_GET['id'];
?>


<a href="view.php?id=3">View Student</a>
```

Links send data using GET, not POST, so I changed POST to GET.

Scenario 10 — Wrong Variable in SQL

✅ **Fixed Code**

```
<?php
$age = $_POST['age'];
$sql = "SELECT * FROM students WHERE age = $age";
?>
```

The original used $aeg which does not exist. I corrected it to $age.

Scenario 11 — Mismatched Method

✅ **Fixed Code**

HTML

```
<form method="GET" action="save.php">
    <input name="email">
</form>
```

PHP

```
<?php
$email = $_GET['email'];
?>
```

I matched the form method (GET) to the PHP superglobal.

Scenario 12 — Numeric GET Used Inside Quotes

☑ **Fixed Code**

```
<?php
$id = $_GET['id'];
$sql = "SELECT * FROM students WHERE id = $id";
?>
```

IDs are numbers, so they do not need quotes in SQL.

Scenario 13 — Missing WHERE Clause in UPDATE

☑ **Fixed Code**

```
<?php
$conn = mysqli_connect("localhost","root","","class_db");

$newEmail = $_POST['email'];
$id = $_POST['id'];

$sql = "UPDATE students SET email='$newEmail' WHERE
student_id=$id";
mysqli_query($conn, $sql);
?>
```

Without a WHERE clause, the update changes all rows. I added the condition.

Scenario 14 — Incorrect POST Array Usage

☑ **Fixed Code**

```
<?php
$data = $_POST;

$sql = "INSERT INTO students (first_name, last_name, email)
        VALUES ('{$data['first_name']}', '{$data['last_name']}',
'{$data['email']}')";
?>
```

I added quotes around array values and fixed the missing ' ' characters.

Scenario 15 — Unsafe Page Number in LIMIT

☑ **Fixed Code**

```
<?php
$page = $_GET['page'];

$page = intval($page);

if ($page < 0) {
    $page = 0;
}

$limit = 5;
$offset = $page * $limit;

$sql = "SELECT * FROM students LIMIT $offset, $limit";
?>
```

I converted the page to integer and prevented negative numbers to avoid errors.