Name: Rico, Donalen Grace C.

Subject: Web Systems

Course & Block: BSIS – 2

Professor: Reymar Llagas

Troubleshooting

SCENARIO **CORRECT**

1	Using $_POST instead of $_GET	Use $_GET['id'] because the value comes from the URL.
2	Missing quotes in SQL when using POST	Add quotes around the string: first_name = '$fname'.
3	SQL injection vulnerability	Use prepared statements to prevent SQL injection.
4	Forgetting to validate empty POST field	Validate the input first (empty()) before inserting to avoid blank records.
5	Wrong key name in POST	Correct the typo by using $_POST['email'] to match the form field name.
6	Unsafe direct use of GET in DELETE	Sanitize the ID using intval() before deleting.
7	Query fails but script continues	Add quotes around the email and include an error check so the script only prints "Updated" if successful.
8	Missing mysqli_fetch_assoc loop	Use a while (mysqli_fetch_assoc()) loop to display all records instead of only one.
9	Using GET but link sends POST	Replace $_POST['id'] with $_GET['id'] because links always send GET requests.
10	Wrong variable used in SQL	Change $aeg to $age so the correct variable

		is used in the SQL query.
11	Mismatched method (expects POST but form sends GET)	Match the form method and PHP by either changing the form to POST or reading the value using $_GET.
12	Numeric GET used inside quotes	Remove the quotes around the numeric ID or cast it as an integer for proper SQL handling.
13	Missing WHERE clause in UPDATE	Add WHERE student_id=$id so only one specific record is updated.
14	Using POST array incorrectly	Use proper array syntax ($data['email']) and add quotes for the SQL string values.
15	GET parameter used inside SQL without sanitization	Validate or limit the page number using intval() to prevent huge offsets from slowing the database.

EXPLAINATION:

Scenario 1 — Using $_POST instead of $_GET

- Kaya nagkaka-error kasi ang value ay galing sa URL, hindi sa form. Ang $_POST is gumagana lang kapag may form na nag-submit. Dapat $_GET kasi URL parameter ang binabasa.

Scenario 2 — Missing quotes in SQL when using POST

- Hindi naiintindihan ng MySQL na string siya kung walang quotes. Iisipin niyang pangalan ng column si “Ana”. Kaya need lagyan ng (' ')para maging literal text.

Scenario 3 — SQL injection vulnerability

- If directly mong isasabay ang GET value sa query, possible na mag-input ang attacker ng “1 OR 1=1” kaya makukuha ang lahat na data. Prepared statements is a "secured way" para hindi mabasa ng SQL bilang command ang input ng user.

Scenario 4 — Forgetting to validate empty POST field

- If you don’t check the fields, puwedeng magcontinue ang blank or empty na data papunta sa database. Para maiwasan ang “empty rows”, kailangan i-check muna kung may laman ang fields bago mag-insert.

Scenario 5 — Wrong key name in POST

- Simpleng typo lang pero grabe ang impact. Naghanap si PHP ng $_POST ['emial'] pero the form name is email. Dapat tugma yumg "name" sa form and ang ginamit sa PHP.

Scenario 6 — Unsafe direct use of GET in DELETE

- Dangerous pag hindi mo ginamitan ng sanitization ang GET value. Puwedeng mag-input ang user ng ?id=0 OR 1=1 kaya mabura ang lahat na students. Using intval() ensures na number sana ang ina-accept.

Scenario 7 — Query fails but script continues

- String ang email kaya dapat naka-quote. If hindi naka-quote, failed ang query pero nagpapakita pa man "Updated!" Dapat laging may error checking: kung failed, ipapakita ang error and kung success, saka palang magprint ng message.

Scenario 8 — Missing mysqli_fetch_assoc loop

- Maski may 100 records, pirmeng 1 row lang ang marireturn pag hindi naka-while(). Kaya dapat naka-loop para ma-display ang lahat na results.

Scenario 9 — Using GET but link sends POST

- Ang link hindi kaya mag-send ng POST, GET sana dapat. Kaya pag ginamit mo $_POST['id'], siguradong empty or undefined index. So dapat $_GET.

Scenario 10 — Wrong variable used in SQL

- $age dapat, pero $aeg ang nagamit. Hindi ito marerecognize ng PHP kaya failed ang SQL.

Scenario 11 — Mismatched method (expects POST but form sends GET)

- If GET ang nasa form, GET man dapat ang basahin kang PHP. Kung POST ang PHP, POST man dapat ang form. Dapat pareho sinda para magkapareho ang data flow.

Scenario 12 — Numeric GET used inside quotes

- Ang ID number, hindi dapat i-treat na string. Pag naka-quotes, nagiging string comparison siya kaya minsan mas bagal or wrong index ginagamit.

Scenario 13 — Missing WHERE clause in UPDATE

- Kung walang WHERE, lahat ng students magbabago ang email. Lagi dapat may condition.

Scenario 14 — Using POST array incorrectly

- Kailangan (' ') sa array keys at sa SQL string values. Kapag kulang, error agad.

Scenario 15 — GET parameter used inside SQL without sanitization

- Kung hindi mo limitahan ang input, puwedeng maglagay ang user ng ?page=99999999 at magka-crash ang database dahil sobrang laki ng offset.