



NAME: SAMANTHA LOUISE V. SOLA

SUBJECT: WEB SYSTEM

COURSE/YEAR&BLOCK: BSIS – 2

INSTRUCTOR: REYMARK LLAGAS

TROUBLESHOOTING

SCENARIO 1:

| PROBLEM | ANSWER | EXPLANATION |
|---------------------------|--|---|
| \$_POST instead of \$_GET | Use \$_GET['id'] instead of \$_POST['id'] | The URL parameter comes through GET, not POST, so using \$_POST gives “undefined index.” Using \$_GET reads the value correctly from the URL. |

SCENARIO 2:

| PROBLEM | ANSWER | EXPLANATION |
|-----------------------|---|--|
| Missing quotes in SQL | \$sql = "SELECT * FROM students WHERE first_name = '\$fname'" | Kailangang naka-quote ang strings sa SQL. Without quotes, MySQL thinks the value is a column name, causing SQL errors. |

SCENARIO 3:

| PROBLEM | ANSWER | EXPLANATION |
|---------------|--------------------------------------|--|
| SQL injection | Use prepared statements with binding | Kung diretso ang user input sa SQL, pwedeng ma-exploit tulad ng 1 OR 1=1 . Prepared statements ensure input is treated as value only, preventing injection. |

SCENARIO 4:

| PROBLEM | ANSWER | EXPLANATION |
|-------------------|----------------------------------|---|
| Empty POST fields | Validate inputs before inserting | Prevents blank or invalid rows from being inserted. |

SCENARIO 5:

| PROBLEM | ANSWER | EXPLANATION |
|----------------|-------------------------------------|--|
| Wrong POST key | Correct key to: \$_POST['email'] | Key must match form input to avoid errors. |

SCENARIO 6:

| PROBLEM | ANSWER | EXPLANATION |
|------------------------|--|---|
| Unsafe DELETE with GET | Use intval(\$_GET['id']) or prepared statement | Direct use ng GET value nag allow ng mga malicious input to delete multiple rows. Casting or prepared statements prevent SQL injection. |



SCENARIO 7:

| PROBLEM | ANSWER | EXPLANATION |
|-------------------|--|---|
| No error checking | Check if query succeeded before printing | Kapag walang check, kahit fail ang SQL, magpi-print ng "Updated!". Always verify query success bago mag-display ng message. |

SCENARIO 8:

| PROBLEM | ANSWER | EXPLANATION |
|--------------------------|--|---|
| Only first record prints | Use while loop with mysqli_fetch_assoc() | mysqli_fetch_assoc() returns one row at a time. Looping ensures lahat ng rows ay ma-display. |

SCENARIO 9:

| PROBLEM | ANSWER | EXPLANATION |
|-------------------------------|-------------------------------|--|
| Using POST but link sends GET | Use <code>\$_GET['id']</code> | Hyperlinks send data via GET , kaya POST will cause " Undefined index ". Align the superglobal with the method. |

SCENARIO 10:

| PROBLEM | ANSWER | EXPLANATION |
|---------------------|---|--|
| Wrong variable name | Use correct variable <code>\$age</code> | Correct variable name prevents undefined errors. |

SCENARIO 11:

| PROBLEM | ANSWER | EXPLANATION |
|-------------------|--|---|
| Mismatched method | Match form method with PHP superglobal (GET ↔ POST) | Method ng form at PHP array ay dapat match para hindi mag-throw ng "Undefined index" error. |

SCENARIO 12:

| PROBLEM | ANSWER | EXPLANATION |
|---------------------------|---|---|
| Numeric GET inside quotes | Remove quotes or cast to int: WHERE id = \$id | IDs are integers. Quotes are unnecessary and may cause indexing inefficiency. |

SCENARIO 13:

| PROBLEM | ANSWER | EXPLANATION |
|-------------------------|---------------------------------|---|
| Missing WHERE in UPDATE | Add WHERE student_id = ? | Para di ma-update lahat ng rows accidentally. |

SCENARIO 14:

| PROBLEM | ANSWER | EXPLANATION |
|----------------------------|---|--|
| Incorrect POST array usage | Use <code>\$data['first_name']</code> and wrap values in quotes | Array keys need quotes; SQL strings need quotes. |



SCENARIO 15:

| PROBLEM | ANSWER | EXPLANATION |
|--------------------|---|---|
| Unsafe page number | Validate and cast page: \$page = max(0, intval(\$_GET['page'])) | Para safe, no super huge page numbers na pwedeng mag-crash SQL. |