KEIRA LATASHA CREOLLO **BSIS2**

TROUBLESHOOTING EXERCISE PHP

Scenario 1 — Using $_POST instead of $_GET

$id = $_GET['id'];

$sql = "SELECT * FROM students WHERE student_id = $id";

ExplanationDapat $_GET ang gamitin pag galing sa URL. ‘Pag po kasi $_POST nilagay, magiging undefined index siya.

Scenario 2 — Missing quotes in SQL

$fname = $_POST['fname'];

$sql = "SELECT * FROM students WHERE first_name = '$fname'";

ExplanationName is a string (ex. "Ana"), kaya dapat po surrounded siya by quotesWHERE first_name = '$fname'.

Scenario 3 — SQL injection vulnerability

$age = $_GET['age'];

$stmt = $conn->prepare("SELECT * FROM students WHERE age = ?");

$stmt->bind_param("i", $age);

$stmt->execute();

$result = $stmt->get_result();

ExplanationDirectly ginamit ang $_GET['age'] sa query. Pwede mag-input ng 1 OR 1=1 mabubuksan lahat ng data. The solution is we need Prepared statements.

Scenario 4 — No validation of empty fields

```
$first = trim($_POST['fname']);

$last  = trim($_POST['lname']);


if ($first != "" && $last != "") {

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

  mysqli_query($conn, $sql);

  echo "Inserted!";

} else {

  echo "Error: fields cannot be empty.";

}
```

ExplanationKahit po empty yung form, nag-iinsert pa rin. Pero neeed pa rin nating icheck muna kung may laman po yung fname at lname bago mag INSERT.

Scenario 5 — Wrong key name in POST

```
$email = $_POST['email']; // correct spelling
```

ExplanationGinamit $_POST['emial'] (mali spelling). Dapat pareho Sila sa form field name (email).

Scenario 6 — Unsafe use of GET in DELETE

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

$sql = "DELETE FROM students WHERE student_id = $id";

mysqli_query($conn, $sql);
```

Explanation

DELETE ... WHERE id = $_GET['id']Pwede ilagay ang ?id=0 OR 1=1 - mabubura siya lahat. So dapat gumamit ng **intval()** or prepared statement.

Scenario 7 — Query error but still prints “Updated!”

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

$res = mysqli_query($conn, $sql);


if ($res) {

  echo "Updated!";

} else {

  echo "SQL Error: " . mysqli_error($conn);

}
```

ExplanationWalang quotes sa email but may SQL error pero nag-e-echo pa rin ng success. So we need to double check kung successful gaya po neto if (!$res) echo "Error";

Scenario 8 — Missing fetch loop

```
while ($row = mysqli_fetch_assoc($res)) {

  echo $row['email'] . "<br>";

}
```

Explanationmysqli_fetch_assoc() once – means isa lang mapprint. We need to use loop like while loop to show all records.

Scenario 9 — Using GET but link sends POST

$id = $_GET['id'];

ExplanationPHP expects $_POST['id'] but link uses GET (view.php?id=3). So dapat gamitin $_GET['id'] kasi po link siya.

Scenario 10 — Wrong variable name

$sql = "SELECT * FROM students WHERE age = $age";

Explanation$aeg instead of $age. Typo siya so nagiging undefined variable kaya dapat we need to always check database for correct spelling.

Scenario 11 — POST expected but form sends GET

$email = $_GET['email'];Explanation Form uses method="GET", pero PHP reads $_POST['email']. Dapat pareho sila either change yung form to POST or change PHP to GET.

Scenario 12 — Numeric GET used inside quotes

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

$sql = "SELECT * FROM students WHERE student_id = $id";**Explanation**The proble is WHERE id = '$id' (ID is integer). Works, pero mas efficient ‘pag walang quotes for numeric values.

Scenario 13 — Missing WHERE clause in UPDATE

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

ExplanationUPDATE students SET email='...' without WHERE.Lahat ng rows ma-uupdate. Dapat may WHERE to target only one student.

Scenario 14 — Using POST array incorrectly

$first = $_POST['first_name'];

$last = $_POST['last_name'];

$email = $_POST['email'];

$sql = "INSERT INTO students (first_name, last_name, email)

VALUES ('$first', '$last', '$email')";

Explanation

$data[first_name] - walang quotes, wrong indexing.Dapat ganito po siya$data['first_name'] may quotes sa SQL.

Scenario 15 — GET parameter without validation

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

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

if ($page > 100) $page = 100; // limit

$limit = 5;

$offset = $page * $limit;**Explanation**User can input huge number like ?page=9999999 it will slow or crash the system. Dapat ivalidate yung limit range to safe numbers like for example (max 100).

Note:

Hello po sir hindi ko na po sinama yung may buggy code tapos yung ibang bug/problem na hindi kasama sa pdf po.