REPUBLIC OF THE PHILIPPINES

BICOL UNIVERSITY
POLANGUI
Polangui, Albay

Email: bupc-dean@bicol-u.edu.ph

NAME: *Enoch Andrew J. Querol* ***SUBJECT:*** *Web System*

COURSE, YEAR & SECTION: *BSIS 2- A* ***PROFESSOR:*** *Sir Reymar* A. Llagas

#Troubleshooting 101 Answers

Scenario 1 — Using $_POST instead of $_GET

✔ Corrected Code

$id = $_GET[‘id’];

$sql = “SELECT * FROM students WHERE student_id = $id”;

✔ Explanation (Humanized)

The value comes from the URL, not a form.

So we must use $_GET, not $_POST.

Using the wrong superglobal causes an “Undefined index” error.

Scenario 2 — Missing quotes in SQL (POST)

✔ Corrected Code

$fname = $_POST[‘fname’];

$sql = “SELECT * FROM students WHERE first_name = ‘$fname’”;

✔ Explanation

String values like names need to be inside quotes in SQL.

Without quotes, MySQL thinks Ana is a column name, not a value.

Scenario 3 — SQL Injection Vulnerability

✔ Corrected (Prepared Statement)

$age = $_GET[‘age’];

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

$stmt->bind_param(“i”, $age);

$stmt->execute();

$res = $stmt->get_result();

✔ Explanation

Prepared statements prevent hackers from typing things like

1 OR 1=1 to break your database.

Scenario 4 — No validation on POST fields

✔ Corrected Code

```
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 in all fields.”;
}
```

✔ Explanation

We must check if the input boxes are not empty before inserting.

This prevents blank or invalid rows

Scenario 5 — Wrong POST key

✔ Corrected Code

```
$email = $_POST[‘email’];
```

✔ Explanation

The original code had emial, a simple typo.

PHP searched for the wrong index → Undefined index error.

Scenario 6 — Unsafe DELETE using GET

✔ Corrected Code

```
$id = intval($_GET[‘id’]);

$sql = “DELETE FROM students WHERE student_id = $id”;
```

✔ Explanation

Intval() prevents dangerous inputs like 0 OR 1=1,

Which could delete all rows.

Scenario 7 — Query fails but script still prints success

✔ Corrected Code

```
$sql = “UPDATE students SET email=’$email’ WHERE student_id=$id”;
$res = mysqli_query($conn, $sql)
If ($res) {
   Echo “Updated!”;
} else {
   Echo “Update failed: “ . mysqli_error($conn)
}
```

✔ Explanation

We added error checking so the script doesn’t lie and

Say “Updated!” when the query actually failed

Scenario 8 — Missing loop

✔ Corrected Code

```
$res = mysqli_query($conn, “SELECT * FROM students”)
While ($row = mysqli_fetch_assoc($res)) {
   Echo $row[‘email’] . “<br>”;
}
```

✔ Explanation

Mysqli_fetch_assoc only retrieves one row.

A while loop is needed to print all emails.

Scenario 9 — GET vs POST mismatch

✔ Corrected Code

```
$id = $_GET[‘id’];
```

✔ Explanation

The link uses GET, so PHP must also read using GET,

Otherwise it will show “Undefined index

Scenario 10 — Wrong variable name

✔ Corrected Code

```
$age = $_POST[‘age’];
$sql = “SELECT * FROM students WHERE age = $age”;
```

✔ Explanation

They accidentally used $aeg instead of $age.

This causes an “Undefined variable” error.

Scenario 11 — Method mismatch

✔ Corrected Fix Option A (Fix HTML)

```
<form method=”POST” action=”save.php”>
   <input name=”email”>

</form>
```

✔ Fix Option B (Fix PHP)

$email = $_GET[‘email’];

✔ Explanation

The form sends GET but PHP expects POST.

HTML and PHP must match.

Scenario 12 — Numeric GET inside quotes

✔ Corrected Code

$id = intval($_GET[‘id’]);

$sql = “SELECT * FROM students WHERE student_id = $id”;

✔ Explanation

ID is a number, so it should not be inside quotes.

Using intval() ensures it is always numeric.

Scenario 13 — UPDATE missing WHERE

✔ Corrected Code

$newEmail = $_POST[‘email’];

$sql = “UPDATE students SET email=’$newEmail’ WHERE student_id = 1”;

✔ Explanation

Missing WHERE updates all rows, which is very dangerous.

Scenario 14 — Using POST array incorrectly

✔ Corrected Code

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

VALUES (‘{$data[‘first_name’]}’, ‘{$data[‘last_name’]}’, ‘{$data[‘email’]}’)”;

✔ Explanation

Array keys need ‘ ‘ and array values need quotes in SQL.

Scenario 15 — Unsafe page number

✔ Corrected Code

$page = isset($_GET[‘page’]) ? intval($_GET[‘page’]) :

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

If ($page > 1000) $page = 1000; // limit

$limit = 5;

$offset = $page * $limit;

$sql = “SELECT * FROM students LIMIT $offset, $limit”;

✔ Explanation

We restrict the page number so users cannot crash MySQL by requesting

Billions of rows.