**504 – WEB FRAMEWORK AND SERVICES**

**Assignment – 1**

**Q1) Explain the difference between client-side and server-side scripting languages.**

**Answer:**

| **Aspect** | **Client-Side Scripting** | **Server-Side Scripting** |
| --- | --- | --- |
| **Execution** | Runs on the client’s browser. | Runs on the server before sending response. |
| **Examples** | JavaScript, HTML, CSS. | PHP, ASP.NET, Python (Django), Node.js. |
| **Usage** | Enhances UI, form validation, interactivity. | Handles database operations, authentication. |
| **Security** | Less secure (code visible to users). | More secure (executed on server). |
| **Speed** | Faster (no server request needed). | Slower (requires server interaction). |
| **Output** | Produces dynamic effects on web pages. | Produces dynamic content (HTML/JSON/XML). |

➡️ **Conclusion:** Both client-side and server-side scripting complement each other to build dynamic, interactive, and secure web applications.

**Q2) What is GET and POST? Explain in detail with Examples.**

**Answer:**  
Both **GET** and **POST** are HTTP request methods used to send data between client and server.

* **GET Method:**
  + Sends data in the URL (query string).
  + Limited to small amounts of data.
  + Less secure (data visible in URL).
  + Example:
  + http://example.com/login?user=zack&password=123
  + $user = $\_GET['user'];
* **POST Method:**
  + Sends data in the body of the request.
  + Can send large and secure data.
  + Not visible in URL.
  + Example:
  + <form method="POST" action="login.php">
  + <input type="text" name="user">
  + <input type="password" name="password">
  + <input type="submit" value="Login">
  + </form>
  + $user = $\_POST['user'];

➡️ **GET** is best for fetching data, while **POST** is best for secure transactions.

**Assignment – 2**

**Q1) Define array. What are the types of arrays? Give Examples.**

**Answer:**  
An **array** is a collection of multiple values stored in a single variable.

**Types of Arrays in PHP:**

1. **Indexed Array:** Uses numeric indexes.
2. $fruits = ["Apple", "Banana", "Mango"];
3. echo $fruits[1]; // Banana
4. **Associative Array:** Uses named keys.
5. $student = ["name"=>"Zack", "age"=>22];
6. echo $student["name"]; // Zack
7. **Multidimensional Array:** Array of arrays.
8. $marks = [
9. ["Maths", 90],
10. ["Science", 85]
11. ];
12. echo $marks[0][1]; // 90

➡️ Arrays help in managing large sets of data efficiently.

**Q2) Explain cookies and session management functions.**

**Answer:**

* **Cookies:**
  + Small pieces of data stored on the client’s browser.
  + Functions:
  + setcookie("user", "Zack", time()+3600); // create cookie
  + echo $\_COOKIE["user"]; // access cookie
* **Sessions:**
  + Store data on the server, more secure.
  + Functions:
  + session\_start();
  + $\_SESSION["user"] = "Zack"; // store session
  + echo $\_SESSION["user"]; // retrieve session

➡️ Cookies are client-side, while sessions are server-side. Both are used for maintaining user state.

**Assignment – 3**

**Q1) Explain the different file handling functions in PHP such as fopen(), fread(), fwrite(), and fclose() with Examples.**

**Answer:**

1. **fopen()** – Opens a file.
2. $file = fopen("test.txt", "r");
3. **fread()** – Reads content from a file.
4. $content = fread($file, filesize("test.txt"));
5. **fwrite()** – Writes data into a file.
6. $file = fopen("test.txt", "w");
7. fwrite($file, "Hello PHP!");
8. **fclose()** – Closes a file.
9. fclose($file);

➡️ These functions allow reading/writing files, useful in logs and data storage.

**Q2) Write PHP code to perform CRUD operations on a MySQL database using mysqli.**

**Answer:**

<?php

$server = "localhost";

$user = "root";

$pass = "";

$db = "college";

$conn = mysqli\_connect($server, $user, $pass, $db);

// (a) Insert

mysqli\_query($conn, "INSERT INTO students (name, age) VALUES ('Zack', 22)");

// (b) Update

mysqli\_query($conn, "UPDATE students SET age=23 WHERE name='Zack'");

// (c) Delete

mysqli\_query($conn, "DELETE FROM students WHERE name='Zack'");

// (d) Display

$result = mysqli\_query($conn, "SELECT \* FROM students");

while($row = mysqli\_fetch\_assoc($result)) {

echo $row['name']." - ".$row['age']."<br>";

}

?>

➡️ This shows **CRUD** operations (Create, Read, Update, Delete) using **mysqli**.

**Assignment – 4**

**Q1) What is AJAX? Discuss the core features of CodeIgniter.**

**Answer:**  
**AJAX (Asynchronous JavaScript and XML):**

* A technique for updating web pages asynchronously without reloading.
* Example: Auto-suggestions in a search bar.

**Core Features of CodeIgniter (PHP Framework):**

1. Lightweight and fast.
2. Follows **MVC architecture**.
3. Built-in libraries for sessions, form validation, database.
4. Secure (XSS filtering, SQL injection prevention).
5. Supports caching for performance.
6. Easy configuration and deployment.

**Q2) Explain the role of MVC architecture in modern web development using CodeIgniter.**

**Answer:**  
**MVC (Model-View-Controller):** A design pattern used in frameworks like CodeIgniter.

1. **Model:** Handles database logic (CRUD operations).
2. class StudentModel extends CI\_Model {
3. public function getStudents() {
4. return $this->db->get("students")->result();
5. }
6. }
7. **View:** Handles UI (HTML, CSS, JS).
8. <h1><?php echo $student->name; ?></h1>
9. **Controller:** Connects Model and View.
10. class StudentController extends CI\_Controller {
11. public function index() {
12. $this->load->model("StudentModel");
13. $data['students'] = $this->StudentModel->getStudents();
14. $this->load->view("studentView", $data);
15. }
16. }

➡️ MVC separates concerns: **Model = Data**, **View = Presentation**, **Controller = Logic**, making apps structured and maintainable.