

Experiment - 07.

- **Aim:** Build a dynamic web application using PHP and MySQL.
 - a. Create database tables in MySQL and create connection with PHP.
 - b. Create the add, update, delete and retrieve functions in the PHP web app interacting with MySQL database.

- **Theory** ⇒

- 1] Introduction to PHP -

- PHP (Hypertext preprocessor) is a server side scripting language used for creating dynamic web applications.
 - It is widely used for database driven web apps due to its seamless integration with MySQL, an open source relational database management system.
 - A PHP my-sql web application enables users to add, update, delete and retrieve data from a MySQL database dynamically.

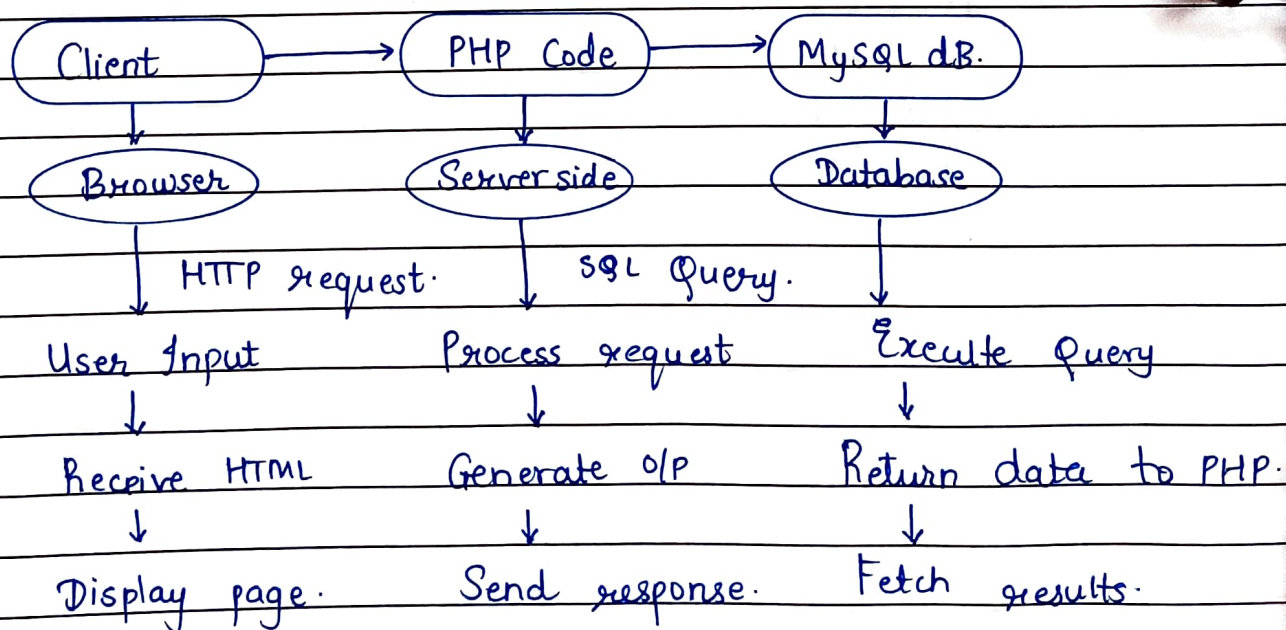
2) PHP and MySQL web application architecture.

A typical php web app follows a three-tier architecture.

Components of PHP-MySQL web application -

1. Client side (Front end) -
 - Users interact with the application via a Web browser.
 - HTML, CSS, JS are used for UI
2. Server side (PHP Backend) -
 - PHP processes requests, executes business logic and interacts with mysql.
3. Database layer (MySQL) - Stores and retrieves application data.

Block diagram -



3] Steps to Build a dynamic PHP MySQL Web application

STEP 1: Set up MySQL Database and Tables.

1. Install MySQL database server.
 - Use XAMPP, WAMP or LAMP
 - Start MySQL service.
2. Create MySQL database.
3. Create required tables, insert sample data.

STEP 2: Establish connection between PHP and MySQL.

1. Enable MySQL Extension in PHP.
 - Ensure it is enabled in php.ini
2. Write PHP Script to connect to database.
 - Use `mysqli_connect()` or new `PDO()` to establish connection.
 - Store database credentials (hostname, username, password, dbname).
3. Handle Connection Errors.
 - Use `die()` or `try-catch` to manage error.

STEP 3: Implement CRUD operations (Create, read, update, delete).

3.1 Create: (Add data to database).

- Users enter details in HTML form.
- PHP processes form data and inserts using `INSERT INTO` Query.
- Validate user input to prevent SQL injection.

3.2 Read (Retrieve data from database and display).

- PHP executes an SQL SELECT query to fetch.
- Retrieved data is displayed using HTML table.
- Use `mysqli-fetch-assoc()` or `PDO::fetch()` to extract data.

3.3 Update (Modify existing records)

- Use UPDATE query to modify existing records.

3.4 Delete (Remove records from database).

- Use the DELETE FROM query to remove the record.

STEP 4 : Handling forms and User input in PHP.

- To collect user input php uses the `$POST` and `$GET` superglobal variables.

1. HTML form -

- Use `<form method="POST">` for data submit.
- `<input type=" " >` for user inputs.

2. PHP forms processing -

- Check if form is submitted using `isset($_POST['submit'])`
- Validate inputs (eg. `empty()`, `filter_var()`).

STEP 5: Displaying data in HTML table.

1. Use a while loop for data fetching.
 - Fetch rows using `mysqli-fetch-assoc()`.
 - Display details inside a `<table>`
2. Enable sorting and filtering.
 - Use SQL `order by` for sorting.
 - Use search filters for refining results dynamically

STEP 6: Deploy PHP application.

- Run PHP scripts via localhost using XAMPP / WAMP.
- Check database connectivity.
- Access your website through server, localhost.

You can also implement user authentication and sessions -

* Implement login and session management -

1. Create user Table in MySQL.
 - Store username, password, email.
2. Implement login system.
 - Use `session-start()` to maintain login state.
 - Validate user credentials using select query.
3. Secure password storage.
 - Use `password-hash()` and `password-verify()` for secure authentication.

* Security Best Practices -

1. Preventing SQL Injection.
 - Always use prepared statements instead of raw SQL queries.
 - Escape inputs using `htmlspecialchars()`.
2. Avoiding Cross site Scripting (XSS).
 - Sanitize user input before displaying.
3. Restricting unauthorized access.
 - Implement session management and role based authentication.
 - Restrict access to sensitive files using `htaccess`.

* **Conclusion:** Thus we successfully implemented CRUD operations using PHP and MySQL.

* * * * *