FSD Laboratory 04

Arin Khopkar, TY B.Tech CSE, Panel I, I2, 62, 1032221073

Aim: Write server-side script in PHP to perform form validation and create database application using PHP and MySQL to perform insert, update, delete and search operations.

Objectives:

1. To understand Server-side Scripting.
2. To learn database connectivity using PHP-MySQL.
3. To perform insert, update, delete and search operations on database.

Theory:

1. PHP Architecture.

PHP (Hypertext Preprocessor) is a widely-used open-source scripting language that is especially suited for web development. It runs on the server-side and generates dynamic content, often interacting with databases to produce custom outputs for users.

Key Components of PHP Architecture:

1. **PHP Interpreter**: This is the core of PHP, responsible for interpreting and executing the PHP code.
2. **Web Server**: PHP is executed on a web server like Apache, Nginx, or IIS, which processes HTTP requests and serves PHP pages to users.
3. **PHP Scripts**: These are files with `.php` extensions that contain the logic for the web application. They can interact with databases, perform operations, and render HTML.
4. **Database (MySQL, PostgreSQL, etc.)**: PHP often interacts with databases to store or retrieve information. MySQL is the most popular database used with PHP.
5. **Client**: The browser (client-side) interacts with the server-side PHP scripts through HTTP requests. PHP generates HTML content based on the client's requests.
6. **Middleware/Frameworks**: PHP can use frameworks like Laravel, CodeIgniter, or Symfony to organize code and apply design patterns like MVC (Model-View-Controller).

Steps for Database connectivity in PHP.

1. Install and Configure Database (MySQL):

- Install a MySQL database on the server or use a hosted database.
- Create a database and user in MySQL with appropriate privileges.

2. Ensure PHP Has Database Extensions:

- Ensure that PHP has the necessary MySQL extensions installed. Usually, **MySQLi** (MySQL improved) or **PDO** (PHP Data Objects) extensions are used for database interaction.
- Check this in the `php.ini` file or by running `phpinfo();` in a PHP script.

3. Connecting to the Database Using MySQLi:

You can connect to the MySQL database using **MySQLi** (Procedural or Object-Oriented approach):

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "test_db";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully!";
?>
```

4. Connecting to the Database Using PDO:

Another method is using the **PDO** (PHP Data Objects) extension. PDO is more flexible as it supports multiple databases (e.g., MySQL, PostgreSQL, etc.).

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "test_db";

try {
    // Create a PDO instance (connect to the database)
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

    // Set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    echo "Connected successfully!";
} catch (PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```

5. Perform Database Queries:

Once connected, you can perform queries such as **SELECT**, **INSERT**, **UPDATE**, and **DELETE**.

MySQLi Example (Procedural):

```
<?php
$sql = "SELECT id, name FROM users";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
   // Output data of each row
   while($row = mysqli_fetch_assoc($result)) {
      echo "id: " . $row["id"]. " - Name: " . $row["name"]. "<br>";
   }
} else {
   echo "0 results";
}
mysqli_close($conn);
?>
```

6. Closing the Connection:

It's good practice to close the database connection after your script completes.

- For **MySQLi**: `mysqli_close($conn);`
- For **PDO**: The connection is automatically closed when the object is destroyed, but you can also set `$conn = null;` to explicitly close it.

7. Error Handling:

Proper error handling is essential to ensure your PHP application can deal with any issues, such as connection failures.

- In **MySQLi**, you can use `mysqli_connect_error()` or check for errors using `mysqli_error($conn)`.
- In **PDO**, exceptions are automatically thrown in the event of an error when you set the error mode to `PDO::ERRMODE_EXCEPTION`.

FAQ:

1. What are the advantages of Server-side Scripting?

A1) Server-side scripting refers to scripts executed on a web server to generate dynamic content before it is sent to the client's browser. PHP is a common server-side scripting language.

1. Dynamic Content Generation:

- Server-side scripts can generate dynamic content based on user inputs, database data, or session information, providing personalized and interactive user experiences.

2. Database Interaction:

- It allows seamless communication with databases to store, retrieve, and manipulate data. For example, fetching user data, updating profiles, and processing form submissions.

3. Security:

- Sensitive information (e.g., passwords, database credentials) is processed on the server, which prevents exposure to the client-side and reduces security risks. Data validation and authorization checks can also be enforced server-side.

4. Cross-Browser Compatibility:

- Server-side scripts output standardized HTML, CSS, and JavaScript, so they work consistently across different browsers without needing compatibility adjustments.

5. Session Management:

- Server-side scripting enables session handling (e.g., login sessions, user tracking) by storing session data on the server, which helps with managing state and user authentication.

6. Reduced Client-Side Load:

- Since most of the processing is done on the server, the client (browser) only receives the final output, reducing the load on the client-side.

7. Code Reusability and Maintenance:

- Server-side scripting languages allow developers to create reusable modules, libraries, and frameworks, improving maintainability and reducing the need for code duplication.

2. What is XAMPP and phpMyAdmin?

A2) XAMPP is a free, open-source cross-platform web server solution stack package developed by **Apache Friends**. It consists of:

- **X**: Cross-platform (can run on multiple platforms: Windows, Linux, Mac OS)
- **A**: Apache (the web server)
- **M**: MySQL or MariaDB (the database)
- **P**: PHP (the server-side scripting language)
- **P**: Perl (another scripting language)

Key Features of XAMPP:

1. **Ease of Installation**: XAMPP bundles Apache, MySQL/MariaDB, and PHP together in a single installation package, simplifying the setup process for developers.
2. **Local Development Environment**: It allows developers to create and test web applications on their local machine without needing a live web server.
3. **phpMyAdmin Integration**: XAMPP comes pre-packaged with phpMyAdmin, a web-based tool for managing MySQL databases.

phpMyAdmin is a free and open-source web-based tool designed to handle the administration of MySQL or MariaDB databases. It provides an intuitive graphical user interface (GUI) for managing databases without needing to use command-line SQL commands.

Key Features of phpMyAdmin:

1. **Database Management**: Create, modify, and delete databases, tables, and fields.
2. **Query Execution**: You can write and execute SQL queries directly within phpMyAdmin.
3. **Data Import/Export**: Import and export data in various formats such as SQL, CSV, XML, etc.
4. **User Management**: Add or manage database users, assign privileges, and set permissions.
5. **Backup and Restore**: phpMyAdmin allows easy database backup and restoration.
6. **Data Manipulation**: Perform operations like inserting, updating, deleting, and browsing table data.

3. What are the two ways to connect to a database in PHP?

A3) In PHP, there are primarily two ways to connect to a MySQL or MariaDB database:

1. MySQLi (MySQL Improved):

MySQLi is an improved version of the original MySQL extension, offering more features and better security.

- **Supports both Procedural and Object-Oriented Syntax**: Developers can choose between a procedural or object-oriented approach to work with databases.
- **Prepared Statements**: MySQLi supports prepared statements, which provide better security against SQL injection.

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "my_database";

// Create a connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully!";
?>
```

2. PDO (PHP Data Objects):

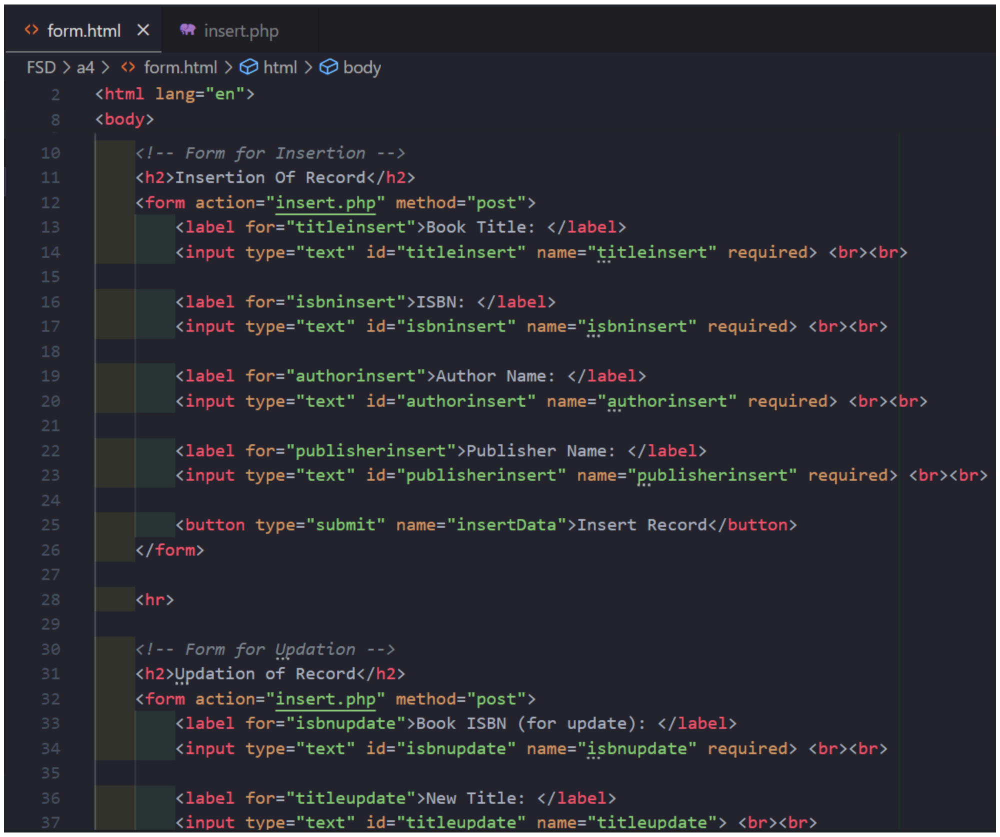
PDO (PHP Data Objects) is a database access layer that supports multiple database types (MySQL, PostgreSQL, SQLite, etc.) using the same API. It provides a consistent way to interact with different databases.

- **Database Flexibility**: PDO supports various databases, not just MySQL.
- **Prepared Statements**: It offers strong protection against SQL injection by using prepared statements.
- **Object-Oriented Only**: PDO is purely object-oriented, and it doesn't offer a procedural API like MySQLi.

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "my_database";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // Set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully!";
} catch (PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```

Code:

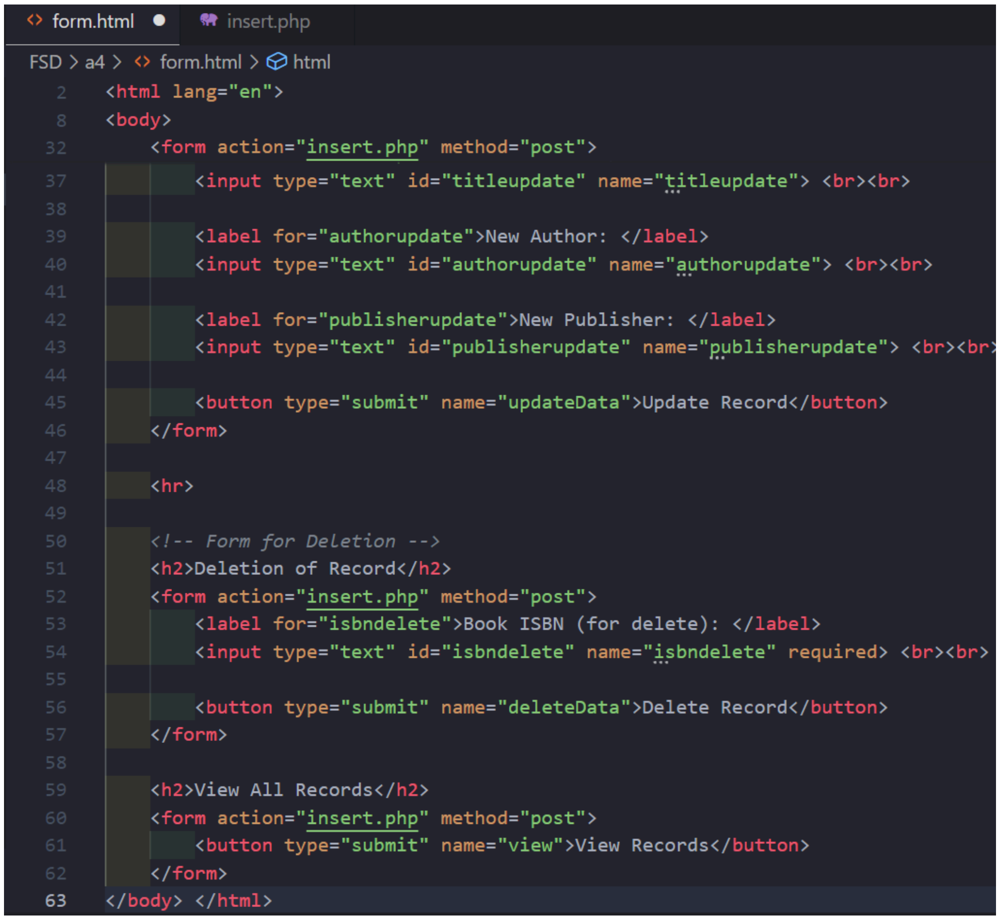
1. **Html file**
2. **PHP file**



form.html × | insert.php

FSD > a4 > form.html > html > body

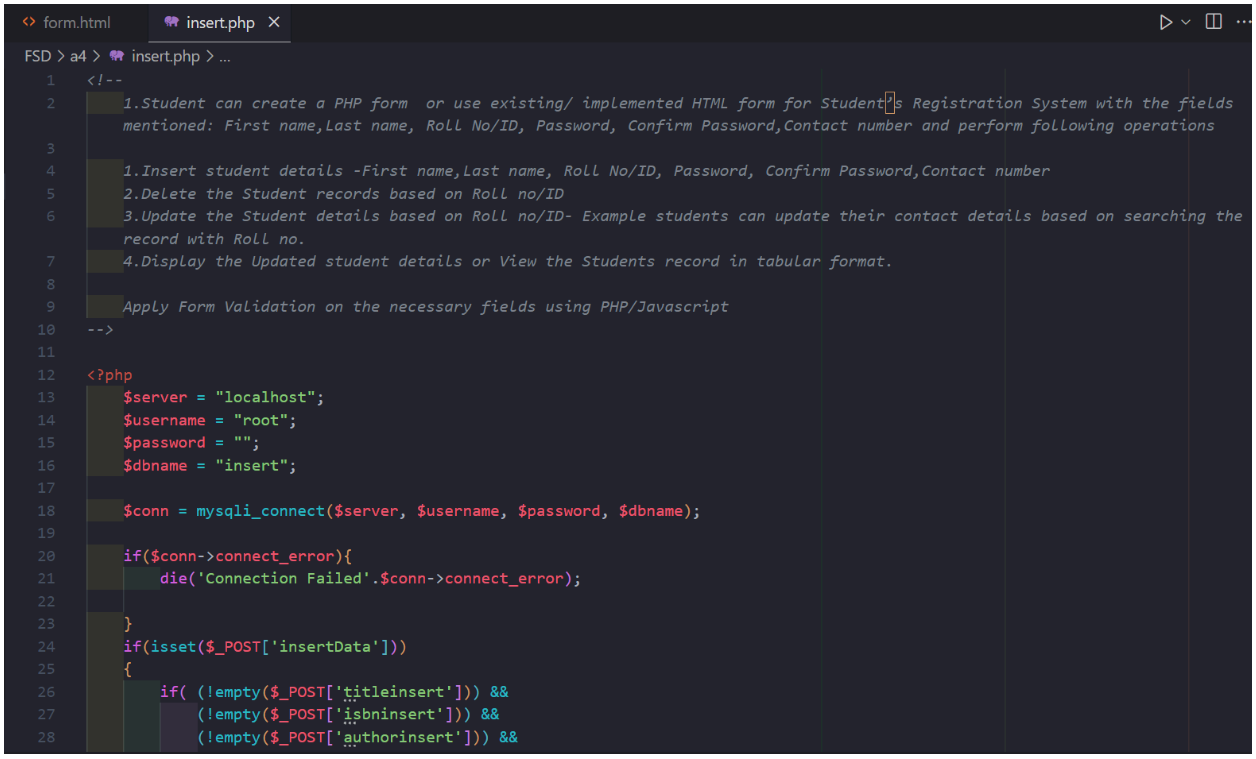
```html
<html lang="en">
<body>
    <!-- Form for Insertion -->
    <h2>Insertion Of Record</h2>
    <form action="insert.php" method="post">
        <label for="titleinsert">Book Title: </label>
        <input type="text" id="titleinsert" name="titleinsert" required> <br><br>

        <label for="isbninsert">ISBN: </label>
        <input type="text" id="isbninsert" name="isbninsert" required> <br><br>

        <label for="authorinsert">Author Name: </label>
        <input type="text" id="authorinsert" name="authorinsert" required> <br><br>

        <label for="publisherinsert">Publisher Name: </label>
        <input type="text" id="publisherinsert" name="publisherinsert" required> <br><br>

        <button type="submit" name="insertData">Insert Record</button>
    </form>

    <hr>

    <!-- Form for Updation -->
    <h2>Updation of Record</h2>
    <form action="insert.php" method="post">
        <label for="isbnupdate">Book ISBN (for update): </label>
        <input type="text" id="isbnupdate" name="isbnupdate" required> <br><br>

        <label for="titleupdate">New Title: </label>
        <input type="text" id="titleupdate" name="titleupdate"> <br><br>
```



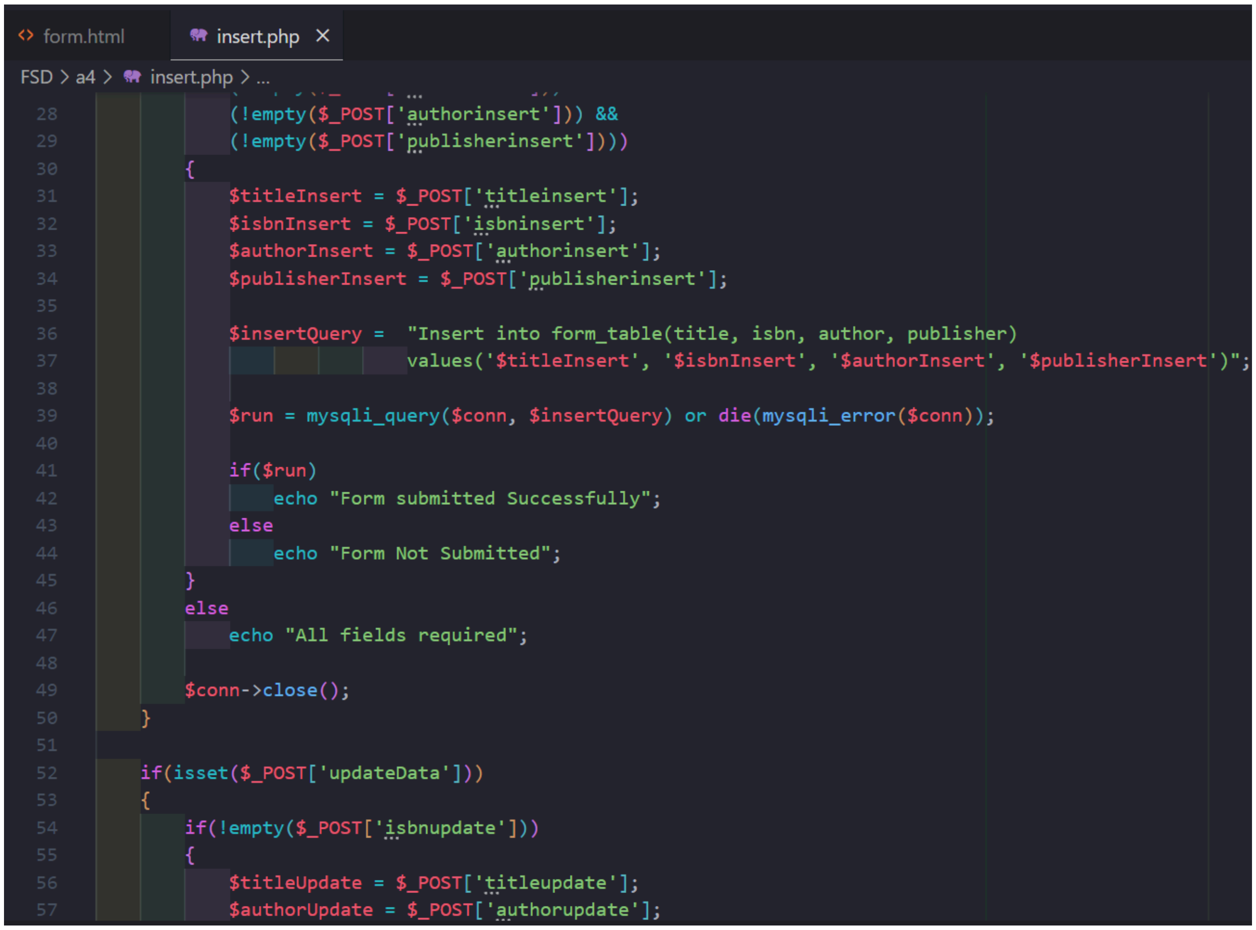
form.html ● | insert.php

FSD > a4 > form.html > html

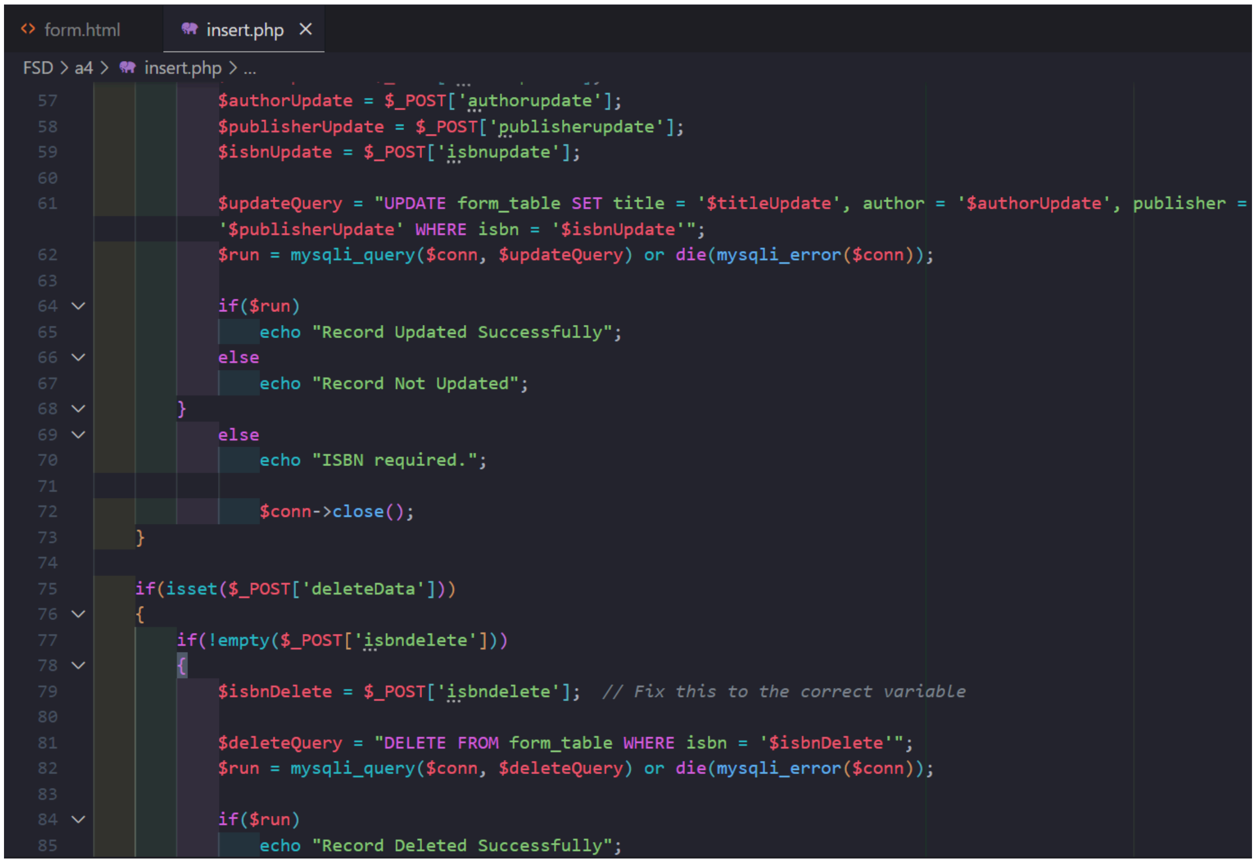
```html
<html lang="en">
<body>
    <form action="insert.php" method="post">
        <input type="text" id="titleupdate" name="titleupdate"> <br><br>

        <label for="authorupdate">New Author: </label>
        <input type="text" id="authorupdate" name="authorupdate"> <br><br>

        <label for="publisherupdate">New Publisher: </label>
        <input type="text" id="publisherupdate" name="publisherupdate"> <br><br

        <button type="submit" name="updateData">Update Record</button>
    </form>

    <hr>

    <!-- Form for Deletion -->
    <h2>Deletion of Record</h2>
    <form action="insert.php" method="post">
        <label for="isbndelete">Book ISBN (for delete): </label>
        <input type="text" id="isbndelete" name="isbndelete" required> <br><br>

        <button type="submit" name="deleteData">Delete Record</button>
    </form>

    <h2>View All Records</h2>
    <form action="insert.php" method="post">
        <button type="submit" name="view">View Records</button>
    </form>
</body> </html>
```



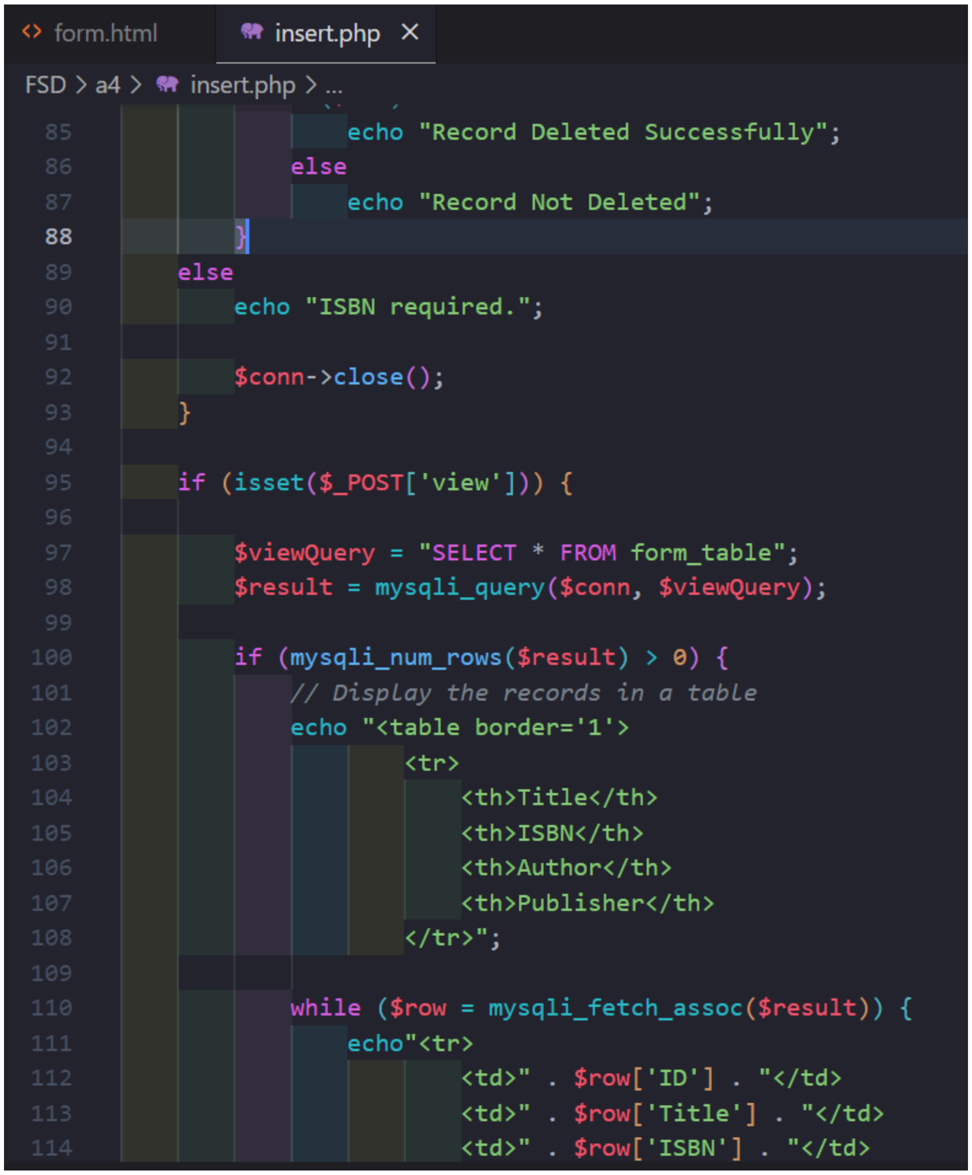
```php
<!--
    1.Student can create a PHP form  or use existing/ implemented HTML form for Student's Registration System with the fields
    mentioned: First name,Last name, Roll No/ID, Password, Confirm Password,Contact number and perform following operations

    1.Insert student details -First name,Last name, Roll No/ID, Password, Confirm Password,Contact number
    2.Delete the Student records based on Roll no/ID
    3.Update the Student details based on Roll no/ID- Example students can update their contact details based on searching the
    record with Roll no.
    4.Display the Updated student details or View the Students record in tabular format.

    Apply Form Validation on the necessary fields using PHP/Javascript
-->

<?php
    $server = "localhost";
    $username = "root";
    $password = "";
    $dbname = "insert";

    $conn = mysqli_connect($server, $username, $password, $dbname);

    if($conn->connect_error){
        die('Connection Failed'.$conn->connect_error);

    }
    if(isset($_POST['insertData']))
    {
        if( (!empty($_POST['titleinsert'])) &&
            (!empty($_POST['isbninsert'])) &&
            (!empty($_POST['authorinsert'])) &&
```



```php
            (!empty($_POST['authorinsert'])) &&
            (!empty($_POST['publisherinsert'])))
        {
            $titleInsert = $_POST['titleinsert'];
            $isbnInsert = $_POST['isbninsert'];
            $authorInsert = $_POST['authorinsert'];
            $publisherInsert = $_POST['publisherinsert'];

            $insertQuery =  "Insert into form_table(title, isbn, author, publisher)
                            values('$titleInsert', '$isbnInsert', '$authorInsert', '$publisherInsert')";

            $run = mysqli_query($conn, $insertQuery) or die(mysqli_error($conn));

            if($run)
                echo "Form submitted Successfully";
            else
                echo "Form Not Submitted";
        }
        else
            echo "All fields required";

        $conn->close();
    }

    if(isset($_POST['updateData']))
    {
        if(!empty($_POST['isbnupdate']))
        {
            $titleUpdate = $_POST['titleupdate'];
            $authorUpdate = $_POST['authorupdate'];
```



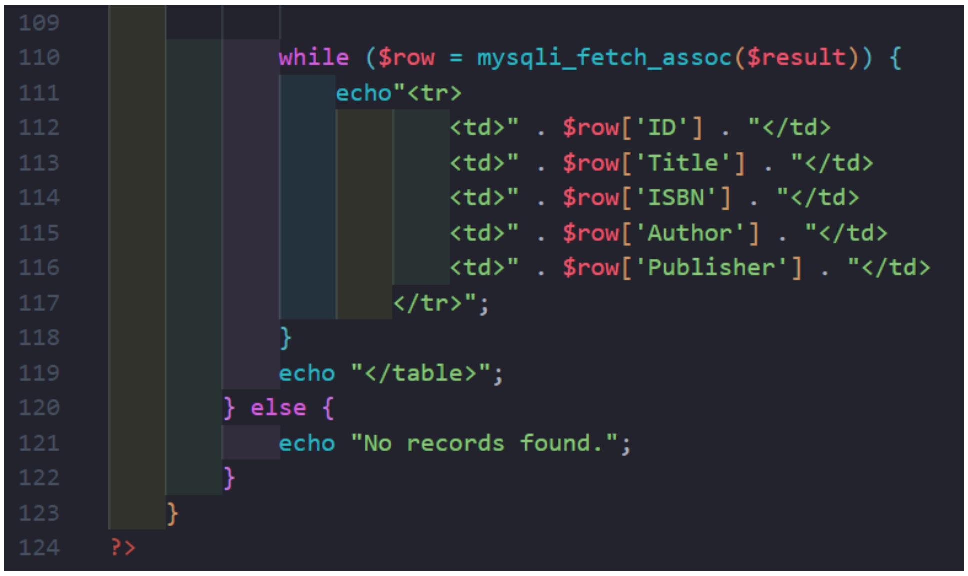
```php
            $authorUpdate = $_POST['authorupdate'];
            $publisherUpdate = $_POST['publisherupdate'];
            $isbnUpdate = $_POST['isbnupdate'];

            $updateQuery = "UPDATE form_table SET title = '$titleUpdate', author = '$authorUpdate', publisher = '$publisherUpdate' WHERE isbn = '$isbnUpdate'";
            $run = mysqli_query($conn, $updateQuery) or die(mysqli_error($conn));

            if($run)
                echo "Record Updated Successfully";
            else
                echo "Record Not Updated";
        }
            else
                echo "ISBN required.";

                $conn->close();
    }

    if(isset($_POST['deleteData']))
    {
        if(!empty($_POST['isbndelete']))
        {
            $isbnDelete = $_POST['isbndelete'];  // Fix this to the correct variable

            $deleteQuery = "DELETE FROM form_table WHERE isbn = '$isbnDelete'";
            $run = mysqli_query($conn, $deleteQuery) or die(mysqli_error($conn));

            if($run)
                echo "Record Deleted Successfully";
```



form.html   insert.php ×

FSD > a4 > insert.php > ...

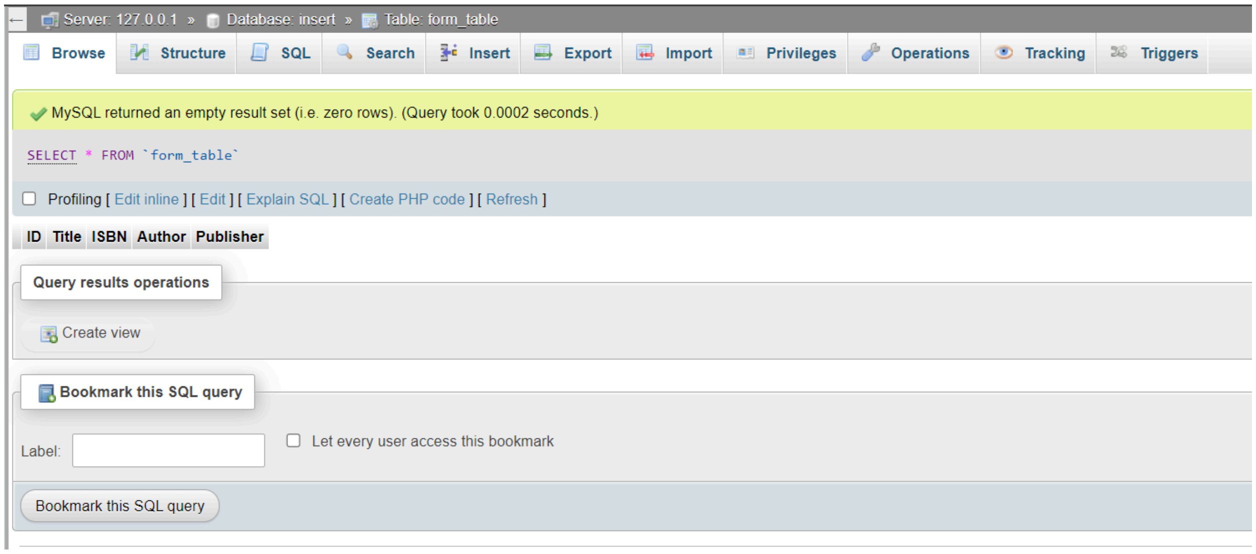
```
            echo "Record Deleted Successfully";
        else
            echo "Record Not Deleted";
        }
    else
        echo "ISBN required.";

        $conn->close();
    }

    if (isset($_POST['view'])) {

        $viewQuery = "SELECT * FROM form_table";
        $result = mysqli_query($conn, $viewQuery);

        if (mysqli_num_rows($result) > 0) {
            // Display the records in a table
            echo "<table border='1'>
                    <tr>
                        <th>Title</th>
                        <th>ISBN</th>
                        <th>Author</th>
                        <th>Publisher</th>
                    </tr>";

            while ($row = mysqli_fetch_assoc($result)) {
                echo"<tr>
                        <td>" . $row['ID'] . "</td>
                        <td>" . $row['Title'] . "</td>
                        <td>" . $row['ISBN'] . "</td>
```



```

            while ($row = mysqli_fetch_assoc($result)) {
                echo"<tr>
                        <td>" . $row['ID'] . "</td>
                        <td>" . $row['Title'] . "</td>
                        <td>" . $row['ISBN'] . "</td>
                        <td>" . $row['Author'] . "</td>
                        <td>" . $row['Publisher'] . "</td>
                    </tr>";
            }
            echo "</table>";
        } else {
            echo "No records found.";
        }
    }
?>
```

Output: Screenshots of the output to be attached.

Database before any operation:



Server: 127.0.0.1 » Database: insert » Table: form_table
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0002 seconds.)

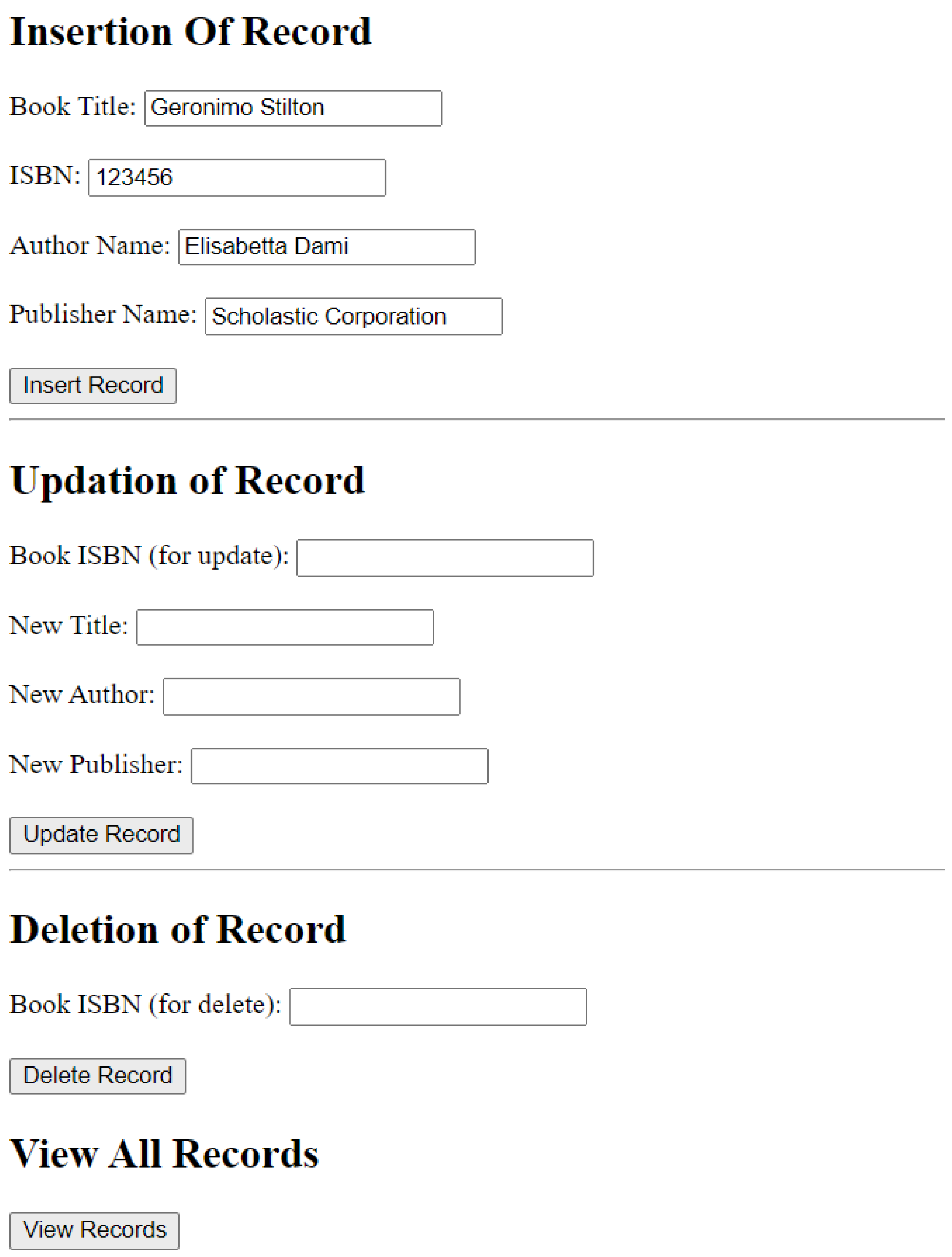
```
SELECT * FROM `form_table`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

ID	Title	ISBN	Author	Publisher

Query results operations
Create view
Bookmark this SQL query
Label:
Let every user access this bookmark
Bookmark this SQL query

Html file for Library Management System:



Insertion Of Record

Book Title: Geronimo Stilton

ISBN: 123456

Author Name: Elisabetta Dami

Publisher Name: Scholastic Corporation

Updation of Record

Book ISBN (for update):

New Title:

New Author:

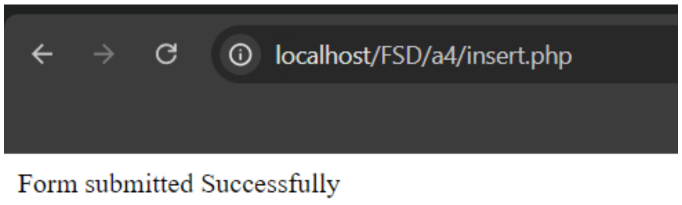
New Publisher:

Deletion of Record

Book ISBN (for delete):

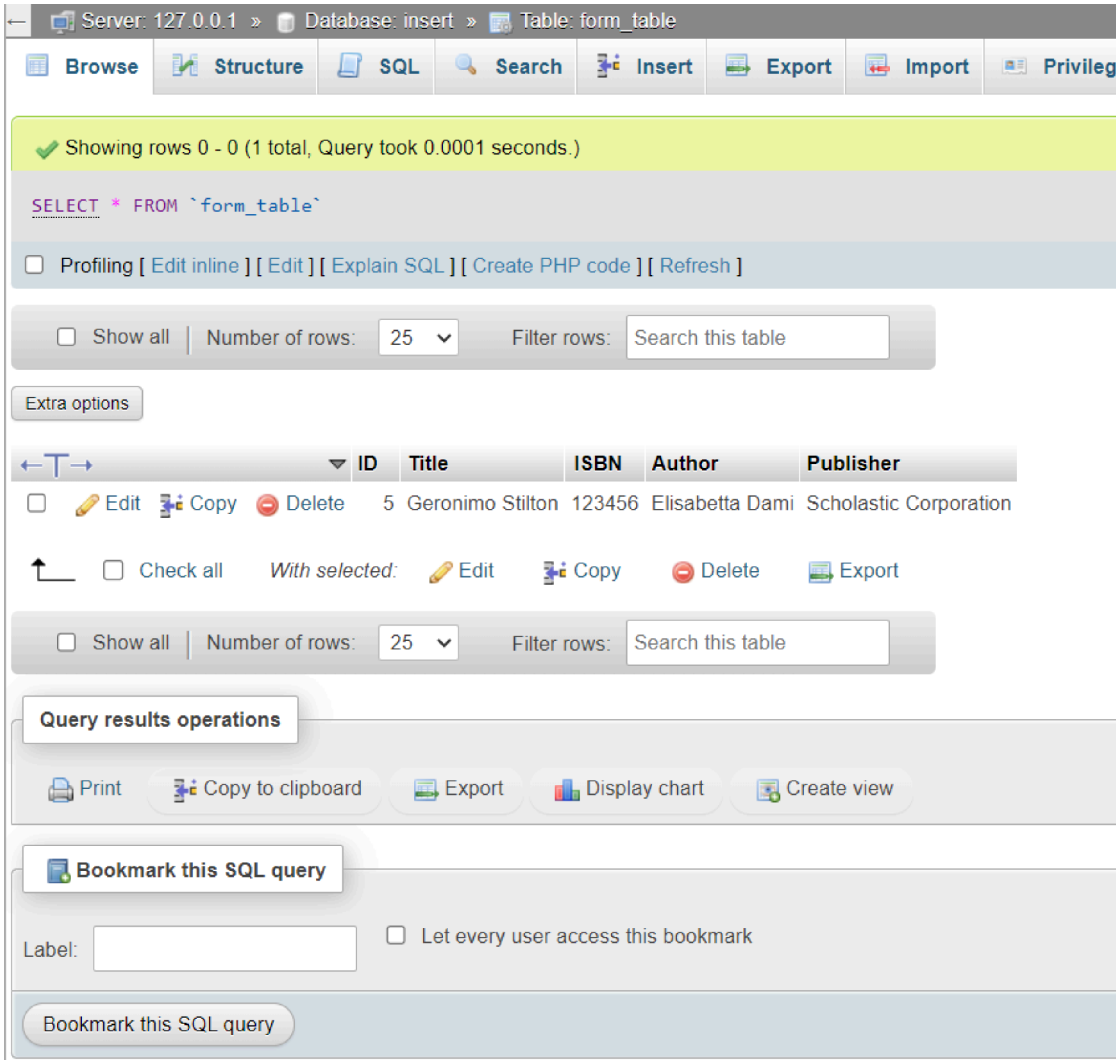
View All Records

After Insertion of record:


localhost/FSD/a4/insert.php

Form submitted Successfully

Database after insertion:


Server: 127.0.0.1 » Database: insert » Table: form_table

Browse | Structure | SQL | Search | Insert | Export | Import | Privileg

Showing rows 0 - 0 (1 total, Query took 0.0001 seconds.)

```
SELECT * FROM `form_table`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

	ID	Title	ISBN	Author	Publisher
Edit Copy Delete	5	Geronimo Stilton	123456	Elisabetta Dami	Scholastic Corporation

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table

Query results operations

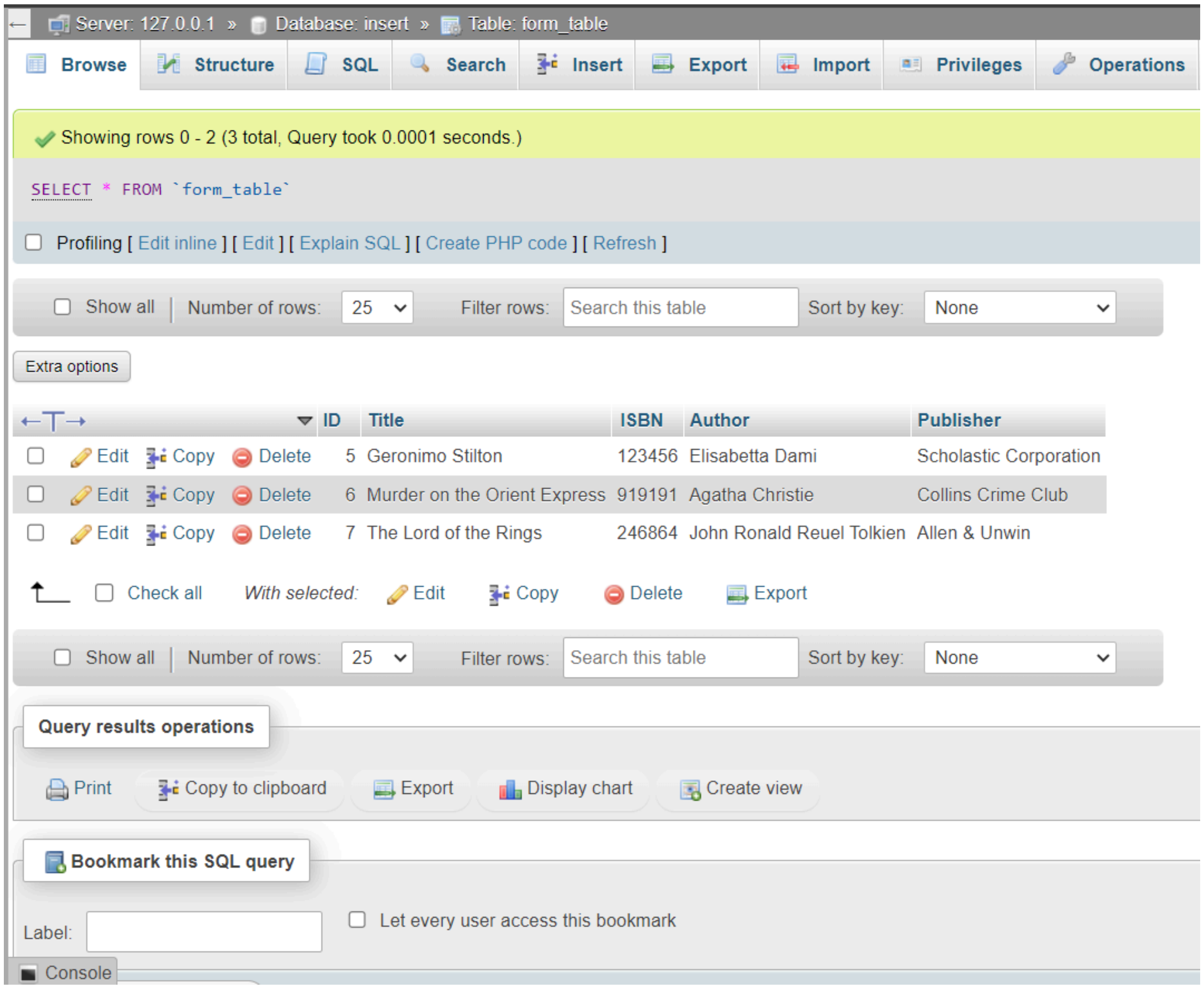
Print Copy to clipboard Export Display chart Create view

Bookmark this SQL query

Label: Let every user access this bookmark

Bookmark this SQL query

After inserting two more entries:



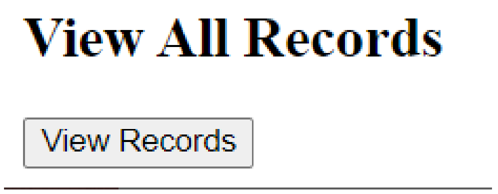
Server: 127.0.0.1 » Database: insert » Table: form_table

Showing rows 0 - 2 (3 total, Query took 0.0001 seconds.)

```
SELECT * FROM `form_table`
```

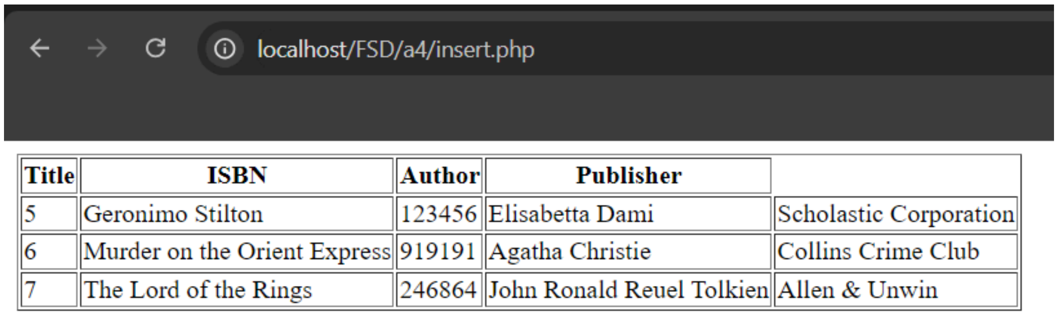
ID	Title	ISBN	Author	Publisher
5	Geronimo Stilton	123456	Elisabetta Dami	Scholastic Corporation
6	Murder on the Orient Express	919191	Agatha Christie	Collins Crime Club
7	The Lord of the Rings	246864	John Ronald Reuel Tolkien	Allen & Unwin

For viewing the entries:



View All Records

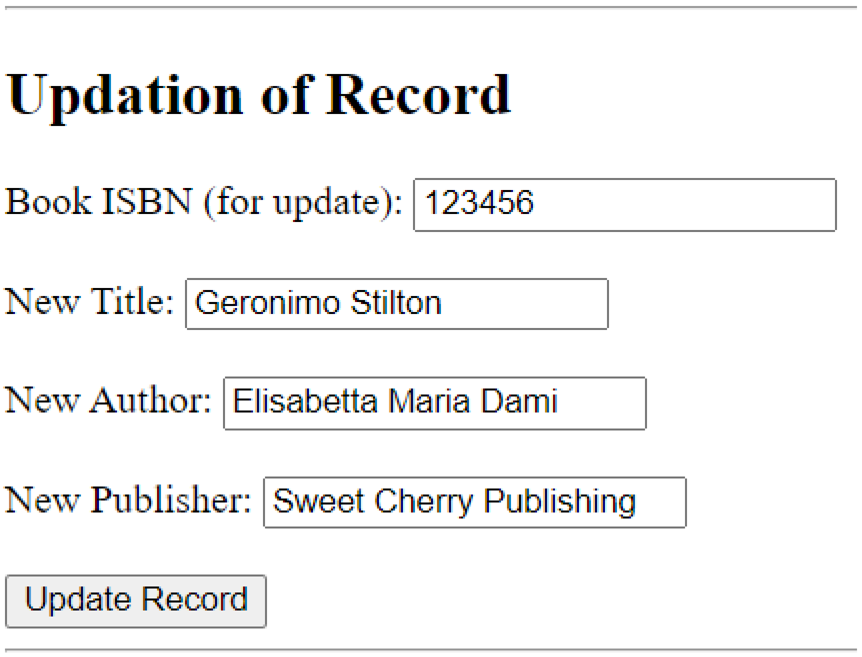
View Records



localhost/FSD/a4/insert.php

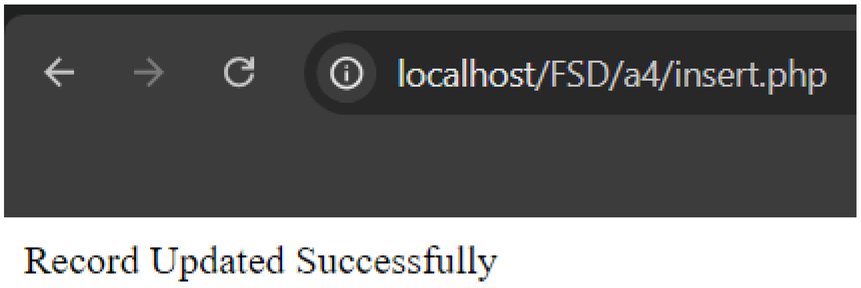
Title	ISBN	Author	Publisher	
5	Geronimo Stilton	123456	Elisabetta Dami	Scholastic Corporation
6	Murder on the Orient Express	919191	Agatha Christie	Collins Crime Club
7	The Lord of the Rings	246864	John Ronald Reuel Tolkien	Allen & Unwin

Updation of record:

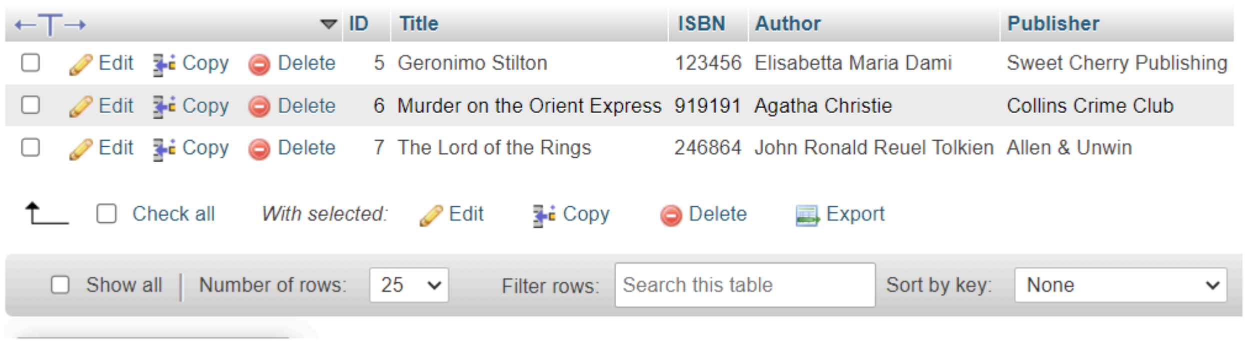


Updation of Record

Book ISBN (for update):	123456
New Title:	Geronimo Stilton
New Author:	Elisabetta Maria Dami
New Publisher:	Sweet Cherry Publishing

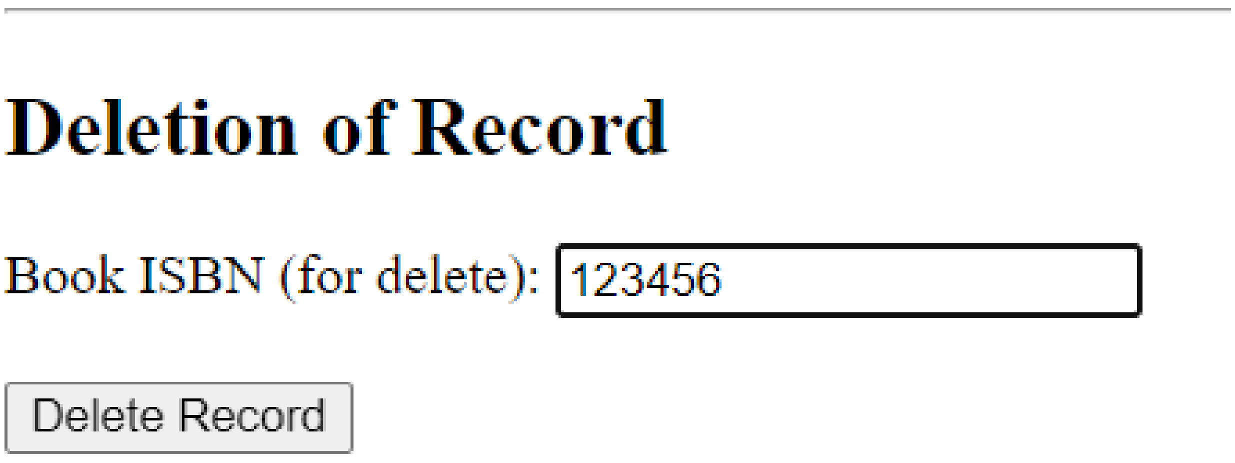


Record Updated Successfully



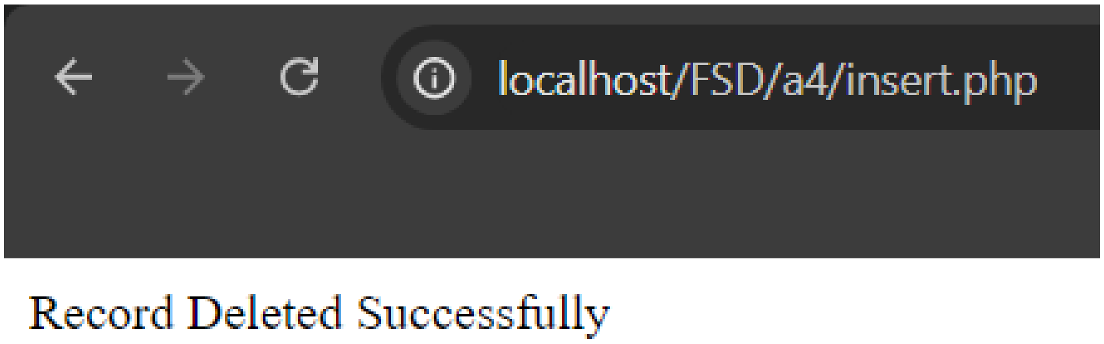
ID	Title	ISBN	Author	Publisher
5	Geronimo Stilton	123456	Elisabetta Maria Dami	Sweet Cherry Publishing
6	Murder on the Orient Express	919191	Agatha Christie	Collins Crime Club
7	The Lord of the Rings	246864	John Ronald Reuel Tolkien	Allen & Unwin

Deletion of record:



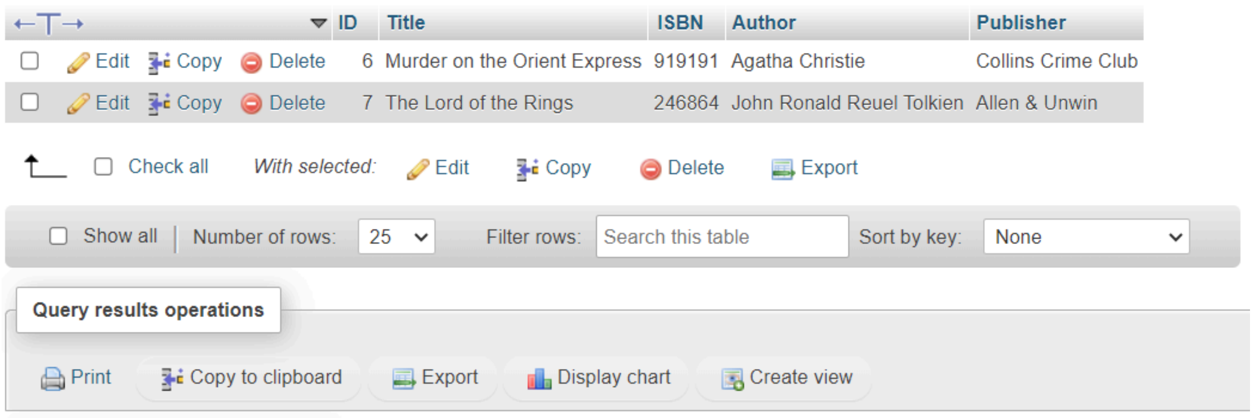
Deletion of Record

Book ISBN (for delete): 123456



localhost/FSD/a4/insert.php

Record Deleted Successfully



	ID	Title	ISBN	Author	Publisher
Edit Copy Delete	6	Murder on the Orient Express	919191	Agatha Christie	Collins Crime Club
Edit Copy Delete	7	The Lord of the Rings	246864	John Ronald Reuel Tolkien	Allen & Unwin

Sample Problem Statements:

PHP CRUD Operations

1.Student can create a PHP form or use existing/ implemented HTML form for Student's Registration System with the fields mentioned: First name,Last name, Roll No/ID, Password, Confirm Password,Contact number and perform following operations

1.Insert student details -First name,Last name, Roll No/ID, Password, Confirm Password,Contact number
2.Delete the Student records based on Roll no/ID
3.Update the Student details based on Roll no/ID- Example students can update their contact details based on searching the record with Roll no.
4.Display the Updated student details or View the Students record in tabular format.

Apply Form Validation on the necessary fields using PHP/Javascript

2. Student can create a PHP form or use existing/ implemented HTML form for Library Management System with the fields mentioned: Book name, ISBN No, Book title, Author name , Publisher name and perform following operations

1.Insert Book details -Book name, ISBN No, Book title, Author name, Publisher name
2.Delete the Book records based on ISBN No
3.Update the Book details based on ISBN No- Example students can update wrong entered book details based on searching the record with ISBN No.
4.Display the Updated Book details or View the Book Details records in tabular format.

Apply Form Validation on the necessary fields using PHP/Javascript

3. Student can create a PHP form or use existing/ implemented HTML form for Employee Management System with the fields mentioned: Employee name, Employee ID, Department_name, Phone number , Joining Date and perform following operations

1.Insert Employee details -Employee name, Employee ID, Department_name, Phone number , Joining Date
2.Delete the Employee records based on Employee ID
3.Update the Employee details based on Employee ID- Example students can update Employee details based on searching the record with Employee ID.
4.Display the Updated Employee details or View the Employee Details records in tabular format.

Apply Form Validation on the necessary fields using PHP/Javascript

4. Student can create a PHP form or use existing/ implemented HTML form for Flight Booking Management System with the fields mentioned: Passenger name, From, to, date,Departure date,Arrival date, Phone number , Email ID and perform following operations

1.Insert Passenger details -Passenger name, From, to, date,Departure date,Arrival date, Phone number , Email ID
2.Delete the Passenger records based on Phone Number

3.Update the Passenger details based on Phone Number - Example students can update Flight Booking details based on searching the record with Phone Number.
4.Display the Updated Flight Booking details or View the Flight Booking Details records in tabular format.

Apply Form Validation on the necessary fields using PHP/Javascript.

Technologies Student Should Use:
XAMPP
PHP for Server-side Scripting
MySQL as a backend Database