MIT World Peace University

Full Stack Development
Third Year B. Tech, Semester 5

Server Side PHP Scripting and Database Management.

Lab Assignment 4

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

September 22, 2023

Contents

1 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.

2 Objectives

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

3 Problem Statement

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

Apply Form Validation on the necessary fields using PHP/Javascript

4 Theory

5 PHP

Definition 1 *PHP stands for* ***Hypertext Preprocessor****. It is a widely-used open-source scripting language primarily designed for web development. PHP is embedded within HTML code and executed on the server-side to generate dynamic web pages.*

5.1 Features

1. **Server-Side Scripting:** PHP is a server-side scripting language, which means it is executed on the web server before the web page is sent to the client's browser.
2. **Cross-Platform Compatibility:** PHP is compatible with various operating systems, including Windows, Linux, macOS, making it highly versatile.

3. **Database Integration:** PHP offers robust support for database integration, allowing developers to interact with databases like MySQL, PostgreSQL, and more.

4. **Extensive Library Support:** PHP has a vast standard library and a thriving community of developers, resulting in a rich ecosystem of extensions and libraries.

5. **Open Source:** PHP is open-source software, which means it's free to use, modify, and distribute, contributing to its widespread adoption.

5.2 Advantages

1. **Ease of Learning:** PHP has a relatively low learning curve, making it accessible to beginners in web development.

2. **Wide Adoption:** PHP is one of the most commonly used languages for web development, ensuring ample resources and community support.

3. **Rapid Development:** PHP's simplicity and numerous built-in functions facilitate rapid web application development.

4. **Platform Independence:** PHP can run on various platforms, ensuring compatibility with different hosting environments.

5. **Active Community:** A large and active PHP community continually updates and enhances the language and its ecosystem.

5.3 History and Significance

1. **Origins:** PHP was created by Rasmus Lerdorf in 1994 as a set of Common Gateway Interface (CGI) binaries written in C. It was initially a simple tool for tracking visits to his online resume.

2. **PHP 3 and Beyond:** PHP 3, released in 1997, introduced a new parser written in C and marked the transition to a more robust scripting language. Subsequent versions, like PHP 4 and PHP 5, added essential features and improvements.

3. **PHP 7:** PHP 7, released in 2015, brought substantial performance improvements, making PHP even more attractive for web development.

4. **Significance:** PHP has played a crucial role in the development of dynamic web applications and websites. It powers a substantial portion of the internet, from small personal blogs to large-scale e-commerce platforms.

6 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers or Interpreters: Brave Browser (Chromium v117.0.5938.88.)

7 Input and Output

1. A Library Interface called 'Novel Tea' Library was set up on the front end usign React. Its Back end was then written in PHP, and the database was set up using Mariadb.
2. The Front end was hosted on another locolhost port, while the php server was hosted using httpd, and default php server.
3. The Front end was then connected to the backend using the simple FETCH API or Axios calls, and the data was sent to the backend using JSON. The responses were also sent from the backend to the Frontend using JSON.
4. The Features of the Library include, and therefore extend to the backend:
   (a) Adding a Book to the Library
   (b) Removing a Book from the Library
   (c) Updating a Book in the Library
   (d) Searching for a Book in the Library
   (e) Getting all the Books from the library to display.
5. These features are demonstrated in the screenshots below.

8 Screenshots

8.1 React Frontend

Figure 1: The Home Page of the Novel Tea Library

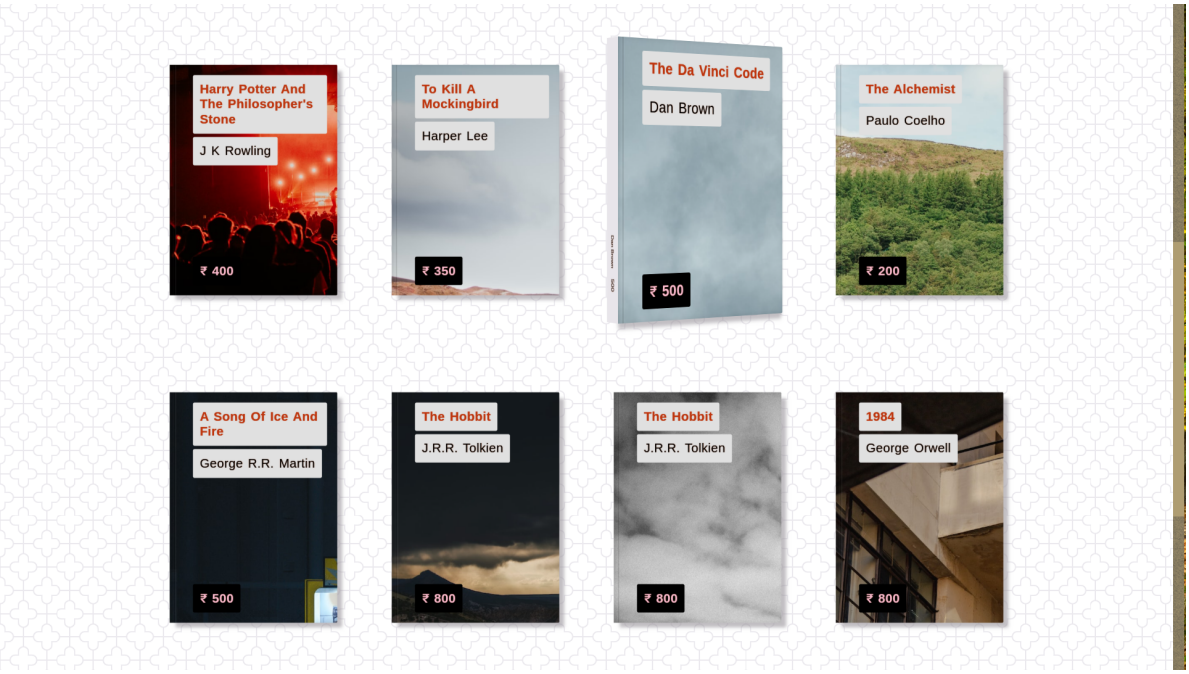

Figure 2: The Books retreived from /getbooks.php and shown on the Frontend

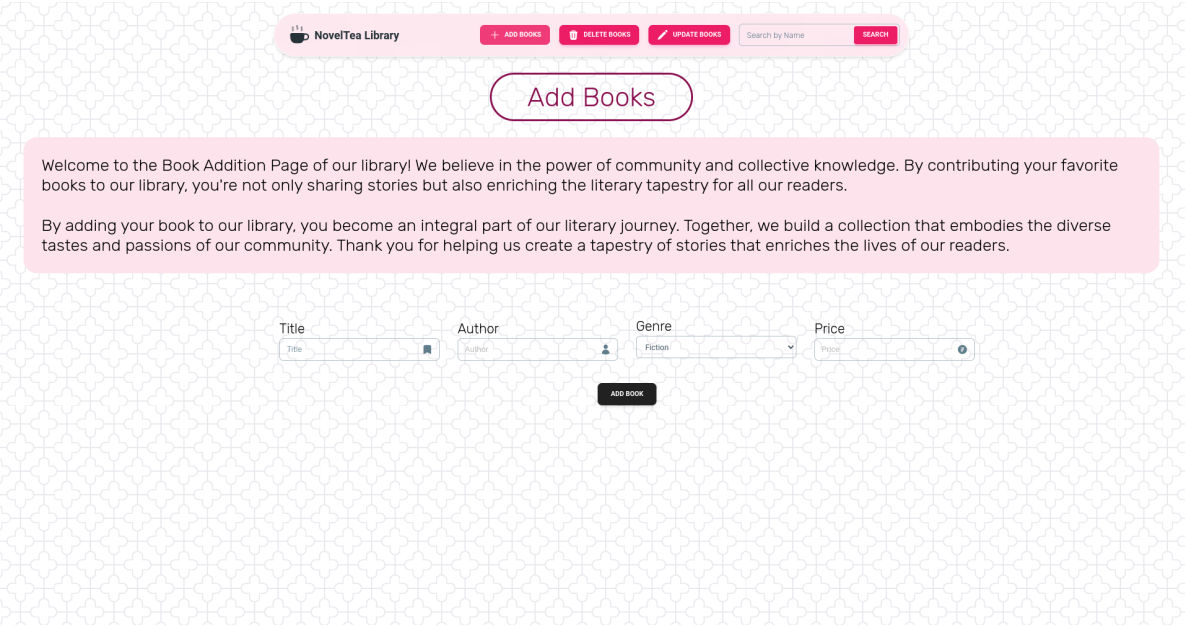

Figure 3: The Add Page

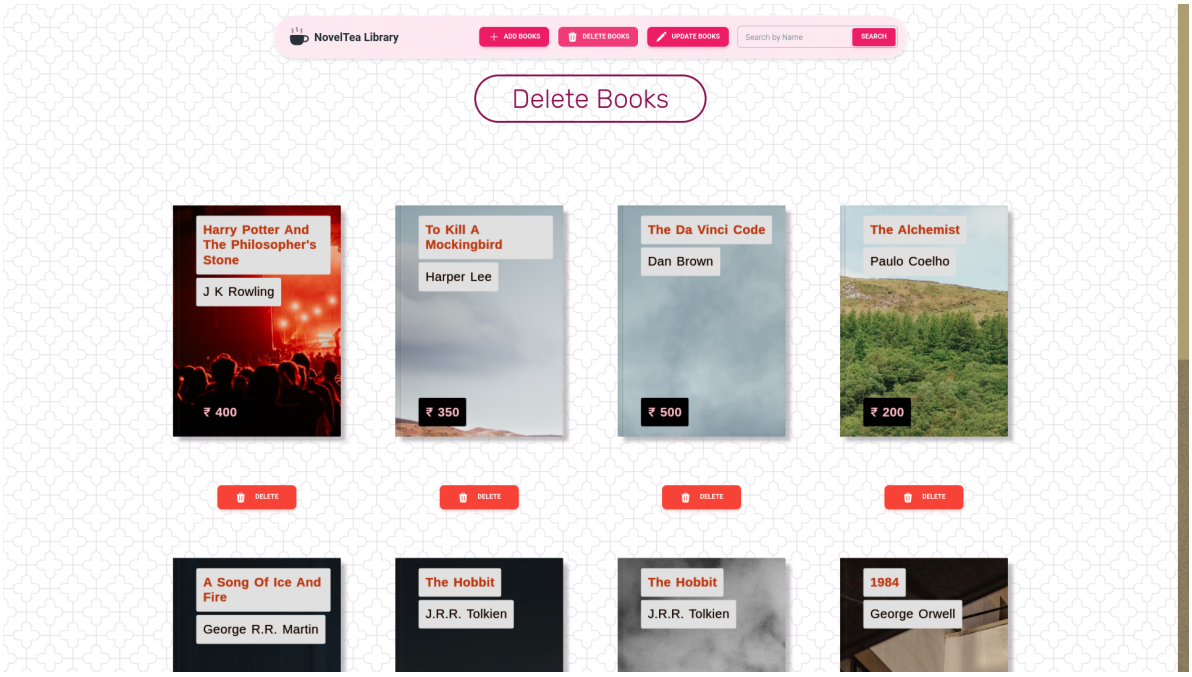

Figure 4: The Delete Page

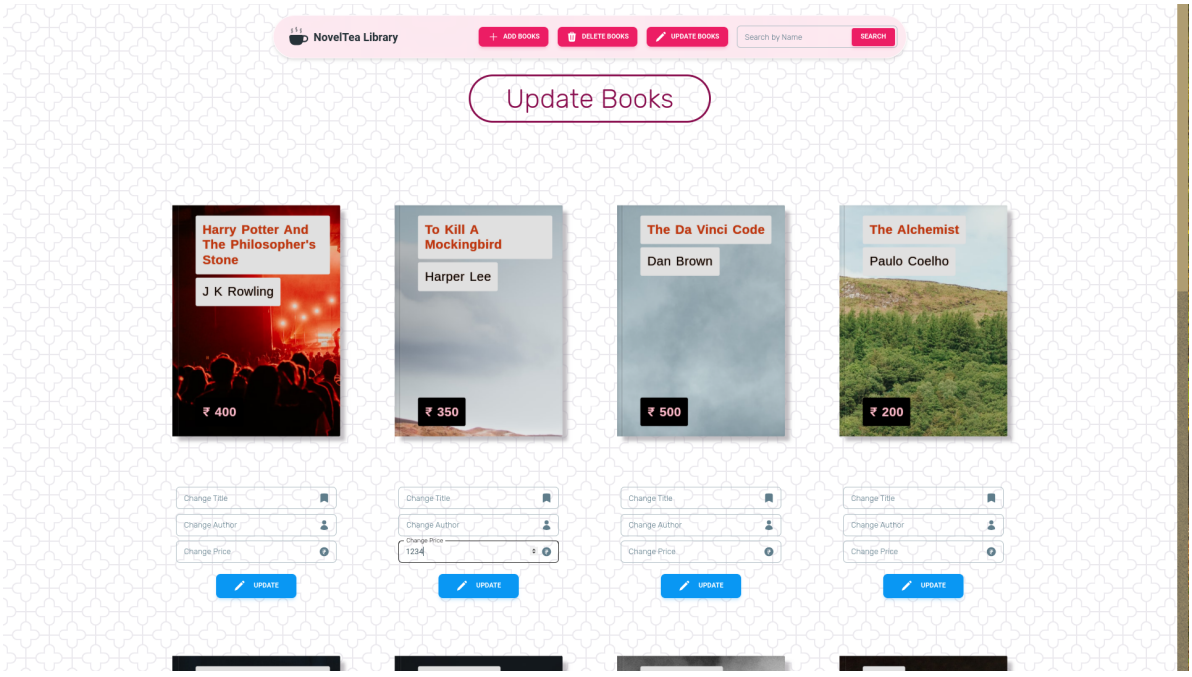

Figure 5: The Update Page, where any field can be updated

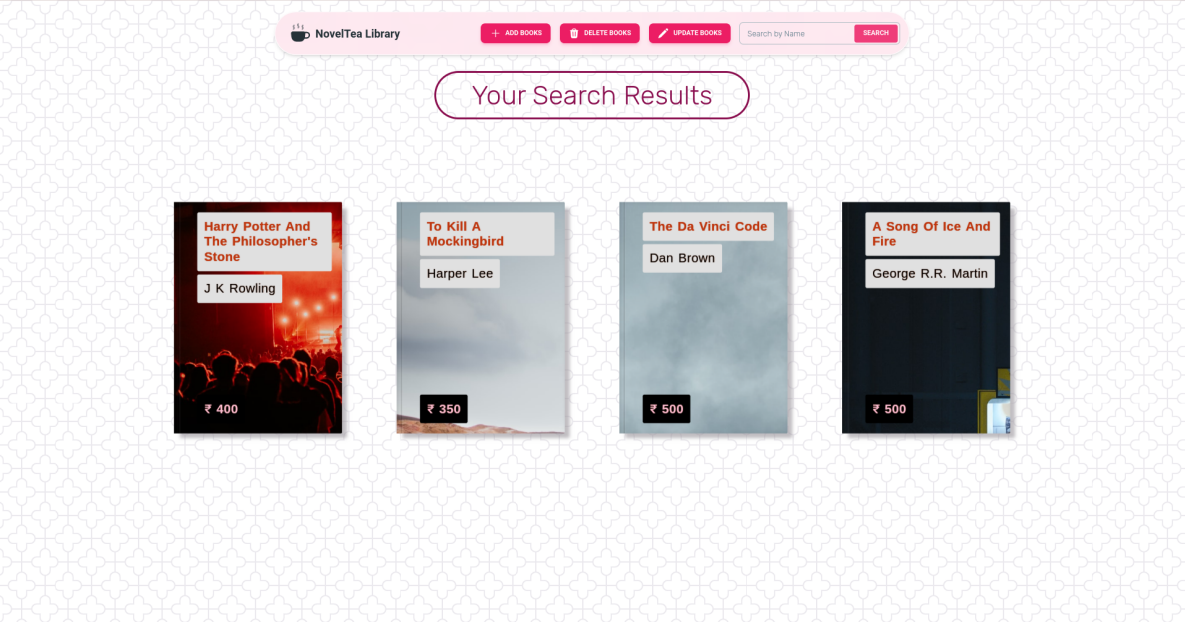

Figure 6: The Search Page showing results of search term "Harry"

8.2 Requests using Postman

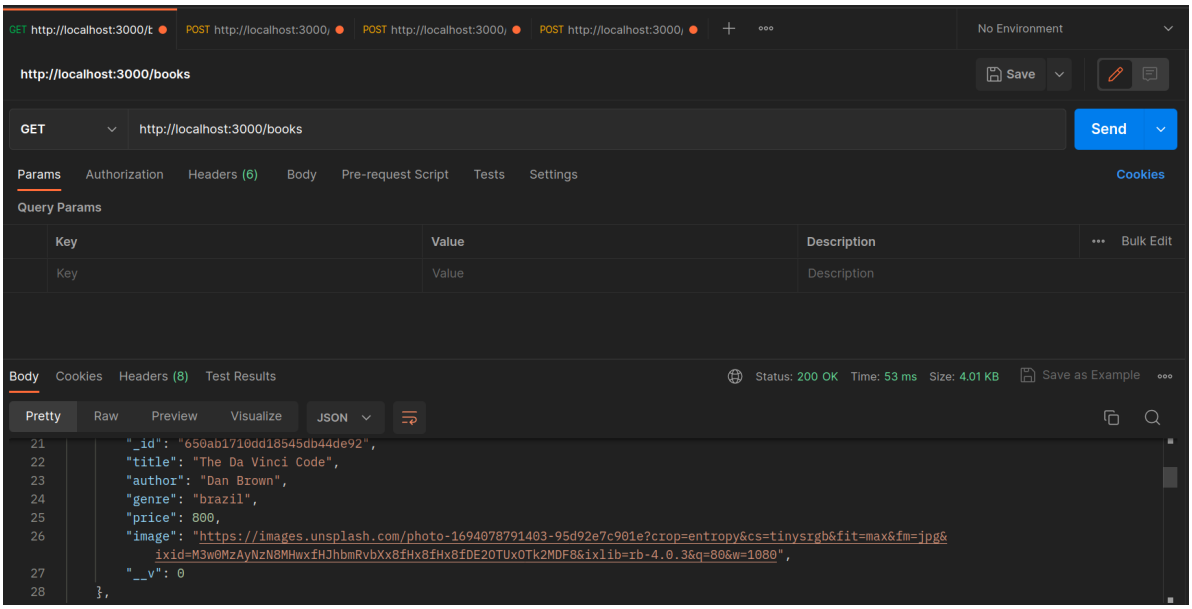

Figure 7: Requests sent using Postman

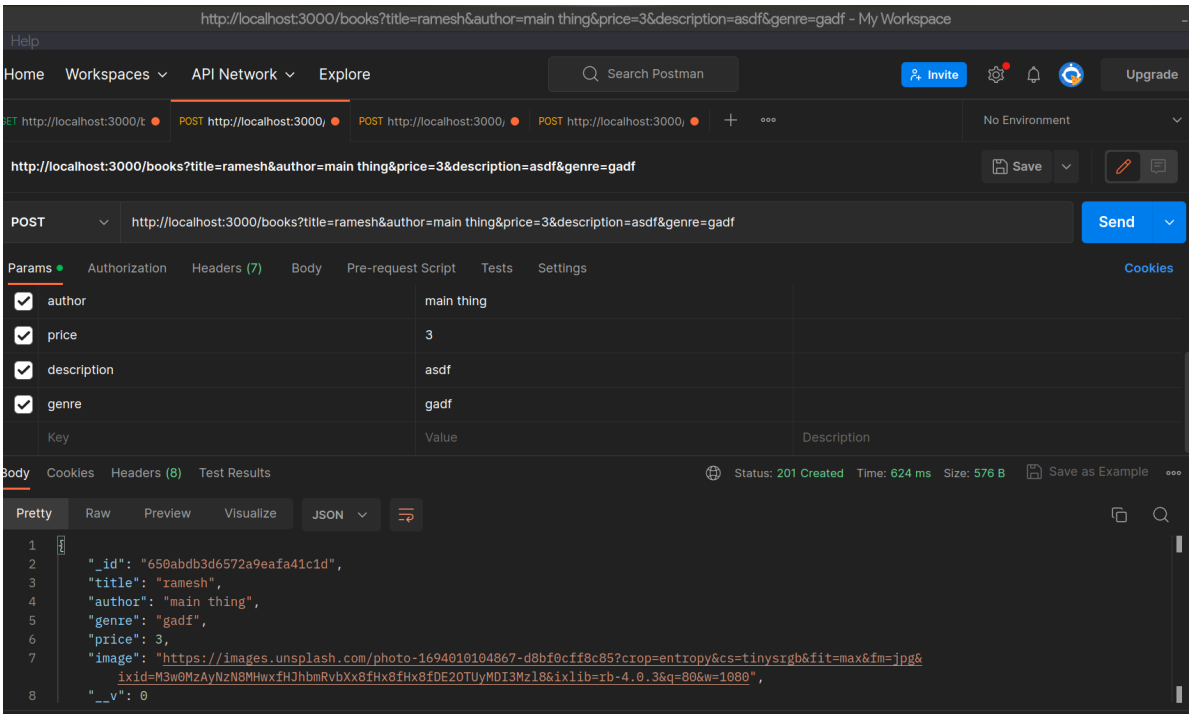

Figure 8: Requests sent using Postman

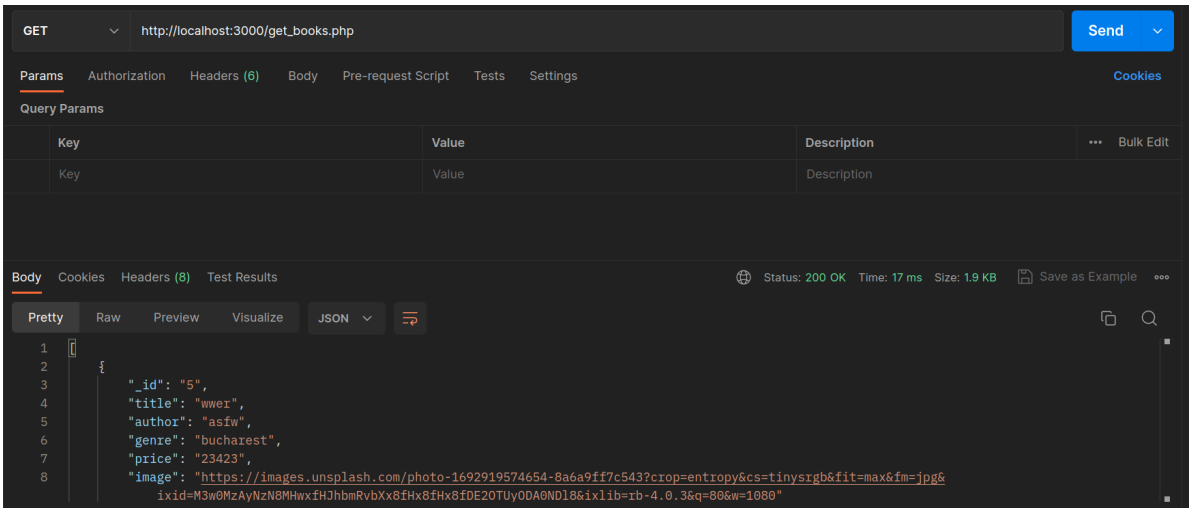

Figure 9: Requests sent using Postman, along with the output from PhP.

8.3 PHP Server

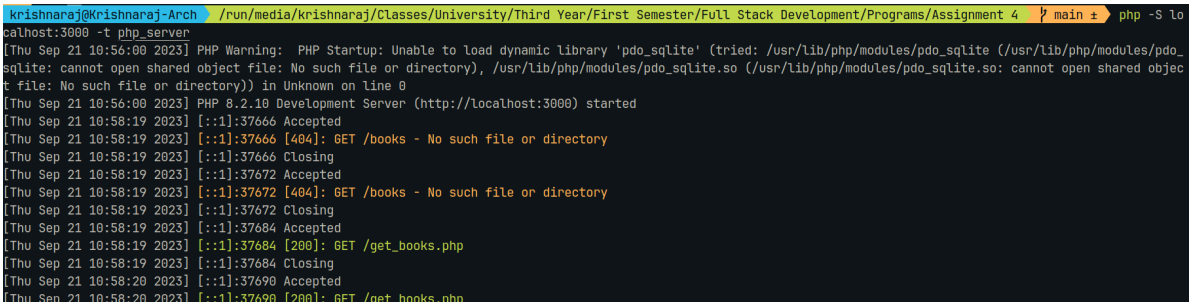

Figure 10: The terminal showing the PHP server running

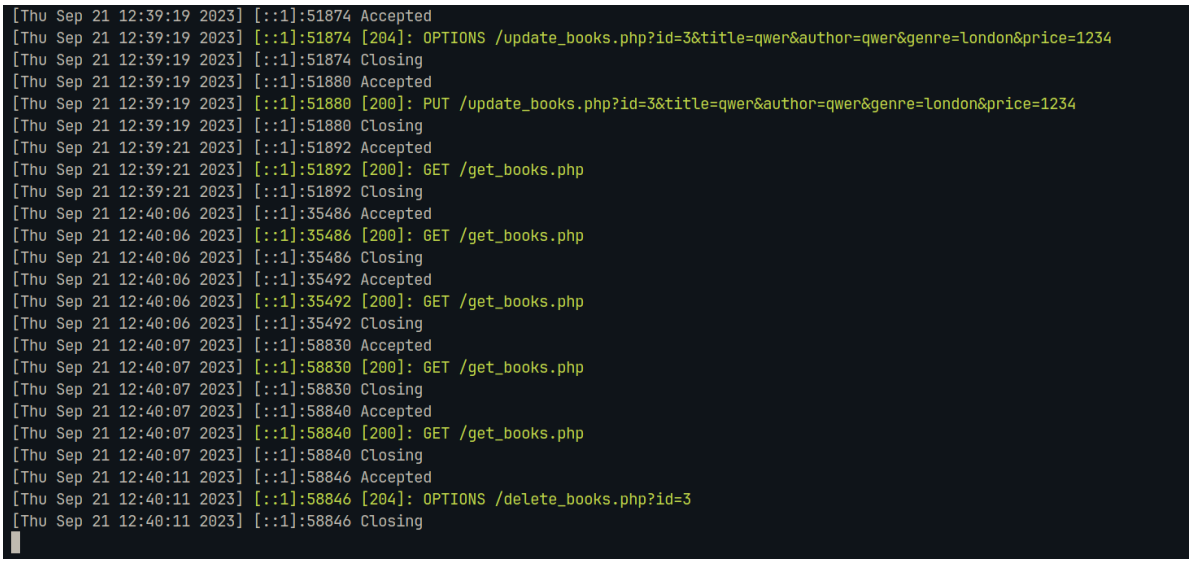

Figure 11: The various requests made to the PHP server are visible.

9 Code

```php
<?php

include("connect_db.php");
include("headers.php");


$sql = "SELECT * FROM books";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // create an array to hold the books
    $books = array();

    // loop through each row and add the book to the array
    while ($row = $result->fetch_assoc()) {
        $book = array(
```

```
            "_id"  => $row["id"],
            "title" => $row["title"],
            "author" => $row["author"],
            "genre" => $row["genre"],
            "price" => $row["price"],
            "image" => $row["image"]
        );
        array_push($books, $book);
    }

    // encode the array as a JSON object and return it
    header('Content-Type: application/json');
    echo json_encode($books);
} else {
    echo "[]";
}
```

Listing 1: "get books.php"

```
<?php
include("connect_db.php");
include("headers.php");

print_r($_REQUEST);


// Get the book data from the post request
$title = $_REQUEST["title"];
$author = $_REQUEST["author"];
$genre = $_REQUEST["genre"];
$price = $_REQUEST["price"];

$endpoint = "https://api.unsplash.com/photos/random";
$headers = array(
    "Authorization: Client-ID UBAXKPq5WgOxU9z7XjLZrt8B0hayG2dv8gh_vyGQg-0"
);

// Make a GET request to the Unsplash API
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $endpoint);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($ch);
curl_close($ch);

// Parse the JSON response and get the image URL
$data = json_decode($response, true);
$image_url = $data["urls"]["regular"];


// Insert the book into the database
$sql = "INSERT INTO books (`title`, `author`, `genre`, `price`, `image`) VALUES (
    '$title', '$author', '$genre', $price, '$image_url')";

if ($conn->query($sql) === TRUE) {
    // Return a success message
    $response = array("message" => "Book added successfully");
    header('Content-Type: application/json');
    echo json_encode($response);
```

```
} else {
    // Return an error message
    $response = array("message" => "Error adding book: " . $conn->error);
    header('Content-Type: application/json');
    echo json_encode($response);
}

$conn->close();
```

Listing 2: "add books.php"

```
<?php
include("connect_db.php");
include("headers.php");


if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
    header("Access-Control-Allow-Origin: *");
    header("Access-Control-Allow-Methods: DELETE");
    header("Access-Control-Allow-Headers: Content-Type, Authorization");
    http_response_code(204);
    exit;
}


// Get the book ID from the request parameters
$id = $_REQUEST["id"];


// Delete the book from the database
$sql = "DELETE FROM books WHERE id = $id";

if ($conn->query($sql) === TRUE) {
    // Return a success message
    $response = array("message" => "Book deleted successfully");
    header('Content-Type: application/json');
    echo json_encode($response);
} else {
    // Return an error message
    $response = array("message" => "Error deleting book: " . $conn->error);
    header('Content-Type: application/json');
    echo json_encode($response);
}

$conn->close();
```

Listing 3: "delete books.php"

```
<?php
include('connect_db.php');
include('headers.php');

if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
    header("Access-Control-Allow-Origin: *");
    header("Access-Control-Allow-Methods: PUT");
    header("Access-Control-Allow-Headers: Content-Type, Authorization");
    http_response_code(204);
    exit;
}
```

```php

// Get the book data from the put request params
print_r($_REQUEST);
print_r($_POST);
$data = array(
    "id" => $_REQUEST["id"],
    "title" => $_REQUEST["title"],
    "author" => $_REQUEST["author"],
    "genre" => $_REQUEST["genre"],
    "price" => $_REQUEST["price"],
);

// Update the book in the database
$sql = "UPDATE books SET
  title = '{$data['title']}',
  author = '{$data['author']}',
  genre = '{$data['genre']}',
  price = {$data['price']}
  WHERE id = {$data['id']}";

if ($conn->query($sql) === TRUE) {
    // Return a success message
    $response = array("message" => "Book updated successfully");
    header('Content-Type: application/json');
    echo json_encode($response);
} else {
    // Return an error message
    $response = array("message" => "Error updating book: " . $conn->error);
    header('Content-Type: application/json');
    echo json_encode($response);
}

$conn->close();
```

Listing 4: "update books.php"

```php
<?php
$servername = "localhost";
$username = "krishnaraj";
$password = "mariamaria";
$dbname = "books";

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

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

Listing 5: "connect db.php"

```php
<?php
// Set CORS headers
header("Access-Control-Allow-Origin: *");
header("Access-Control-Allow-Methods: GET");
header("Content-Type: application/json");

header("Access-Control-Allow-Origin: *");
```

```
8  header("Access-Control-Allow-Methods: DELETE");
9  header("Access-Control-Allow-Headers: Content-Type");
10
11 header("Access-Control-Allow-Origin: *");
12 header("Access-Control-Allow-Methods: PUT");
13 header("Access-Control-Allow-Headers: Content-Type, Authorization");
```

Listing 6: "headers.php"

10 Conclusion

Thus, we have learnt how to create a responsive web page using Bootstrap and Media Queries. We have also learnt to use HTML, CSS and Javascript to create a game. We have used Git and Github to create a repository and push our code to it.

11 FAQ

1. *Advantages of Server-side Scripting:*

    - **Dynamic Content Generation:** Server-side scripting allows dynamic content generation based on user input, database queries, or other variables, enhancing user interactivity.
    - **Data Security:** Sensitive data and processing logic can be kept on the server, reducing the risk of exposure to client-side manipulation or security breaches.
    - **Platform Independence:** Server-side scripts are executed on the server, making web applications compatible with various client devices and operating systems.
    - **Enhanced Performance:** Server-side processing offloads resource-intensive tasks from the client's device, improving overall application performance.

2. *XAMPP and phpMyAdmin:*

    - **XAMPP:** XAMPP is a free, open-source cross-platform software package that provides an easy way to set up a local web server environment. It includes components like Apache (web server), MySQL (database server), PHP, and Perl, allowing developers to test and develop web applications on their local machines before deploying them to a live server.
    - **phpMyAdmin:** phpMyAdmin is a web-based graphical user interface (GUI) tool for managing MySQL databases. It simplifies tasks such as creating, editing, and deleting databases and tables, running SQL queries, and managing user privileges, making database administration more accessible.

3. *Two Ways to Connect to a Database in PHP:*

    - **Using MySQLi (MySQL Improved):** MySQLi is a PHP extension specifically designed for MySQL database interaction. It provides an object-oriented and procedural interface for connecting to MySQL databases, executing queries, and fetching results. MySQLi supports features like prepared statements for enhanced security.
    - **Using PDO (PHP Data Objects):** PDO is a database abstraction layer in PHP that offers a consistent API for connecting to various database management systems, including MySQL, PostgreSQL, and SQLite. It allows developers to write database-agnostic code and offers features like prepared statements and error handling.