Name: SAYYAM SABOO

Roll No: PA-17

Batch: A-1

PRN: 1032211097

**FSD ASSIGNMENT - 4**

**Theory**

1. PHP Architecture.

PHP (Hypertext Preprocessor) is a server-side scripting language commonly used for web development. Its architecture can be divided into several key components and layers, each serving a specific purpose in the execution of PHP web applications. Here is an overview of the PHP architecture:

* **Web Server:** PHP applications run on web servers, such as Apache, Nginx, or Microsoft Internet Information Services (IIS). These web servers are responsible for receiving HTTP requests from clients (browsers) and forwarding them to the PHP interpreter for processing.
* **PHP Interpreter:** The PHP interpreter is the core component of the PHP runtime environment. It processes PHP code embedded within HTML or standalone PHP files. When a request is received by the web server and contains PHP code, the web server invokes the PHP interpreter to execute the code.
* **PHP Code:** PHP code can be embedded directly within HTML files using tags like <?php ... ?> or <?= ... ?>, or it can reside in separate .php files. PHP is a server-side language, so it runs on the server, processes the code, and generates HTML or other output to be sent back to the client's browser.
* **PHP Extensions:** PHP supports a wide range of extensions and libraries that provide additional functionality. These extensions can be bundled with PHP or added separately. Common extensions include MySQLi for database connectivity, cURL for making HTTP requests, and GD for image manipulation.
* **Database:** PHP often interacts with databases to store, retrieve, and manipulate data. Popular database management systems used with PHP include MySQL, PostgreSQL, SQLite, and Oracle. PHP provides database-specific extensions and functions for connecting to and working with databases.

2.Steps for Database connectivity in PHP

To establish database connectivity in PHP, you typically need to follow these steps:

* **Choose a Database Management System (DBMS):** Before connecting to a database in PHP, you need to decide which database management system you want to use. Common options include MySQL, PostgreSQL, SQLite, Oracle, and Microsoft SQL Server. Ensure that your chosen DBMS is installed and properly configured on your server.
* **Install PHP:** Make sure that PHP is installed on your web server. PHP is commonly used for server-side scripting and database connectivity.
* **Install Database Extension:** Depending on your chosen DBMS, you may need to install a PHP extension that allows PHP to communicate with the database. For example, if you're using MySQL, you'll need the MySQLi or PDO extension. These extensions are typically included in PHP by default, but you may need to enable them in your PHP configuration.
* **Create a Database:** Use your DBMS's administration tools or command-line interface to create a database. Define tables, fields, and relationships as needed for your application.
* **Gather Database Information:** You will need the following information to connect to the database:
  + Database server hostname or IP address.
  + Database name.
  + Database username and password with appropriate permissions.

By following these steps, you can establish a database connection in PHP and interact with your chosen database management system to store and retrieve data for your web application.

**FAQs**

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

Server-side scripting offers several advantages when developing web applications and websites. Here are some of the key benefits:

* Security: Server-side scripting allows sensitive logic and data to be processed on the server, keeping it hidden from the client-side code and potentially malicious users. This helps protect against common security threats such as Cross-Site Scripting (XSS) and SQL Injection, as sensitive data and operations are not exposed to the client.
* Data Privacy: With server-side scripting, you can control what data is shared with clients. You can restrict the amount and type of data sent to the client, ensuring that confidential information is kept secure on the server.
* Performance: Server-side scripts can perform resource-intensive operations on the server, reducing the load on the client's device. This can lead to faster page loading times, especially for complex or data-intensive applications, as the server handles tasks like database queries and data processing.
* Scalability: Server-side applications can be more easily scaled by adding more server resources or employing load balancing techniques. This scalability ensures that your application can handle increased traffic and maintain good performance as your user base grows.
* Code Reusability: Server-side scripting promotes code reusability by allowing you to separate server-side logic from the presentation layer. You can reuse server-side code for various parts of your application or even across different projects, leading to more efficient development and easier maintenance.
* Data Manipulation: Server-side scripting languages offer powerful libraries and frameworks for data manipulation, transformation, and validation. This makes it easier to work with databases, process data, and generate dynamic content.
* Cross-Browser Compatibility: Server-side code can generate HTML and other client-side code that is compatible with various web browsers. This helps ensure consistent rendering and functionality across different browsers and platforms.
* Database Interaction: Server-side scripting enables direct interaction with databases, allowing you to store, retrieve, and manipulate data efficiently. This facilitates the creation of dynamic, data-driven web applications.

While server-side scripting offers these advantages, it's essential to strike a balance between server-side and client-side processing to create responsive and user-friendly web applications. Combining the strengths of both server-side and client-side scripting can lead to a well-rounded and efficient web development approach.

1. What is XAMPP and phpMyAdmin?

**XAMPP (Cross-Platform, Apache, MySQL, PHP, and Perl):**

* **Purpose:** XAMPP is a free, open-source web server solution designed to facilitate local web development and testing. It bundles together several key components necessary for web development, including the Apache web server, MySQL (or MariaDB) database server, PHP, and Perl.
* **Features:**
  + **Apache:** XAMPP includes the Apache HTTP Server, which is one of the most widely used web servers in the world. It allows developers to host and serve web content locally.
  + **MySQL/MariaDB:** XAMPP provides a database server (either MySQL or MariaDB) for storing and managing data. This is crucial for web applications that require database functionality.
  + **PHP:** XAMPP includes PHP, a server-side scripting language widely used for web development. Developers can run PHP scripts and create dynamic web applications.
  + **Perl:** Though less common in web development than PHP, XAMPP also includes Perl, a general-purpose scripting language.

**phpMyAdmin:**

* **Purpose:** phpMyAdmin is a web-based administration tool for managing MySQL or MariaDB databases. It provides a user-friendly interface for performing tasks such as creating databases, tables, and user accounts; importing and exporting data; executing SQL queries; and more.
* **Features:**
  + **Database Management:** phpMyAdmin allows you to create, modify, and delete databases and database objects like tables, views, and stored procedures.
  + **Data Manipulation:** You can insert, update, and delete data records in database tables using the graphical interface.
  + **SQL Querying:** phpMyAdmin provides a SQL query editor, enabling you to write and execute SQL queries directly.

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

In PHP, you can connect to a database using two primary methods: the MySQLi extension (MySQL Improved) and the PDO (PHP Data Objects) extension. Both methods allow you to interact with various database management systems, including MySQL, PostgreSQL, SQLite, Oracle, and more. Here's an overview of each approach:

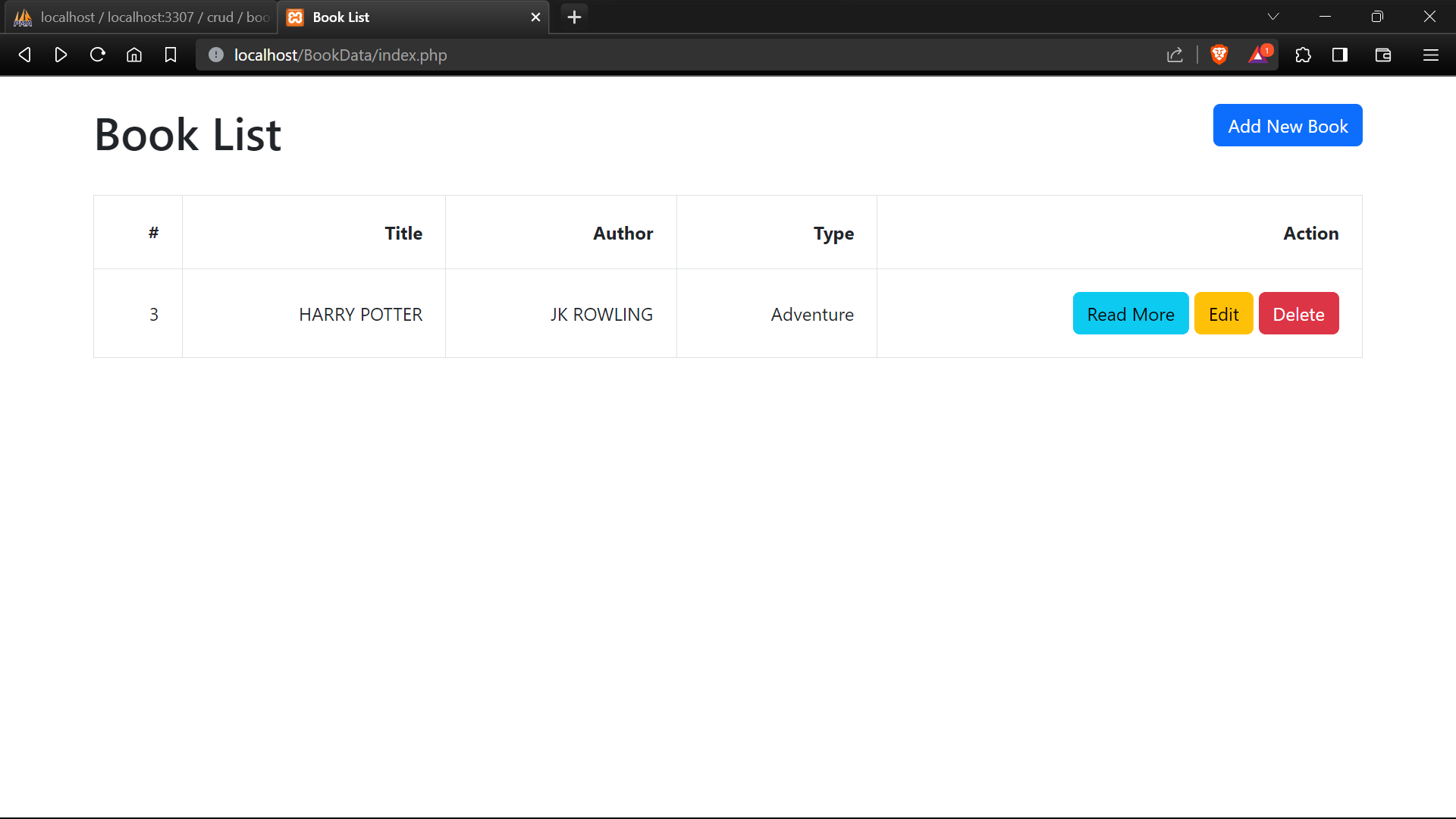
**MySQLi (MySQL Improved):**

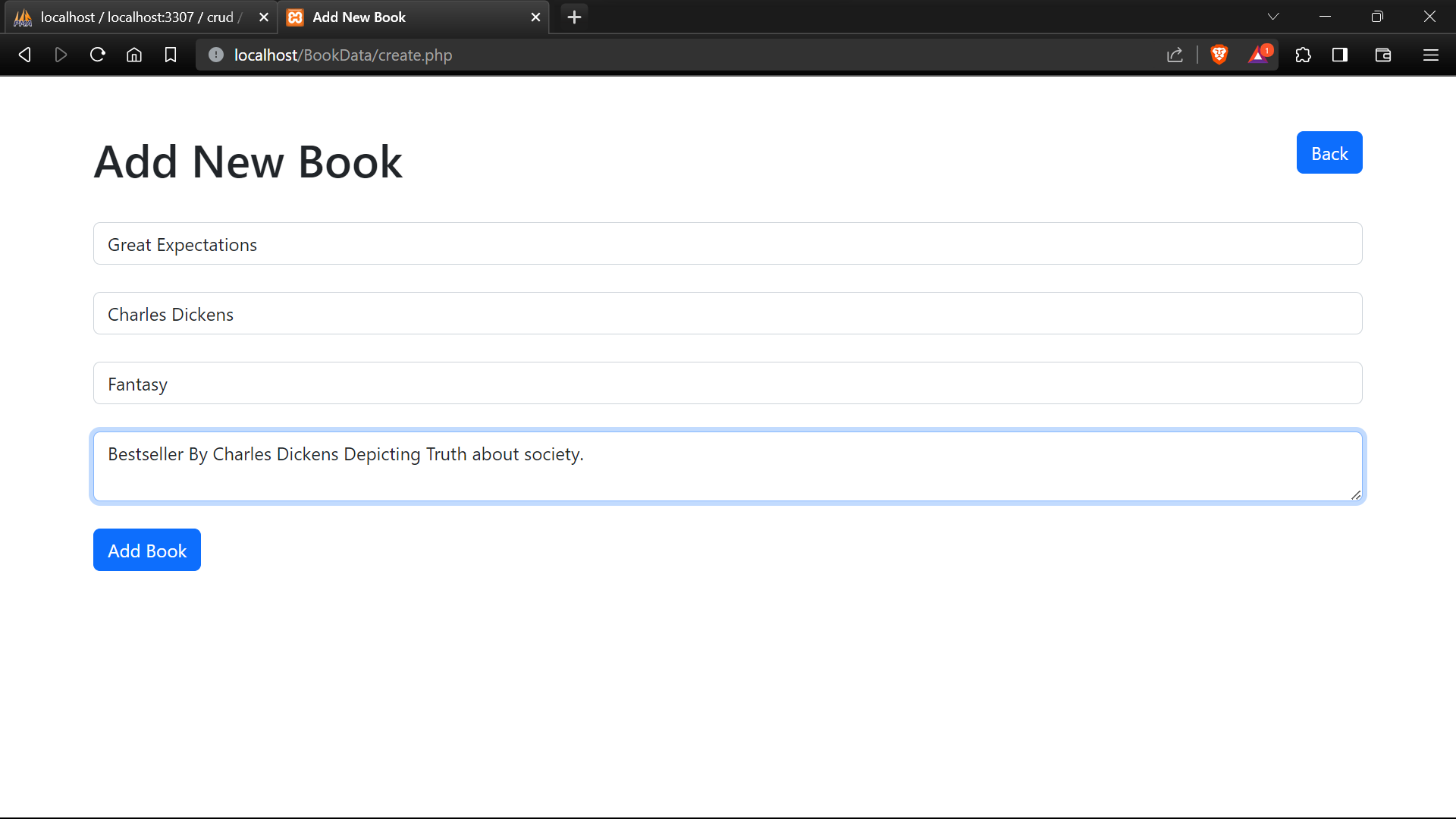
MySQLi is a PHP extension specifically designed for interacting with MySQL databases. It provides a procedural and object-oriented API for database connectivity.

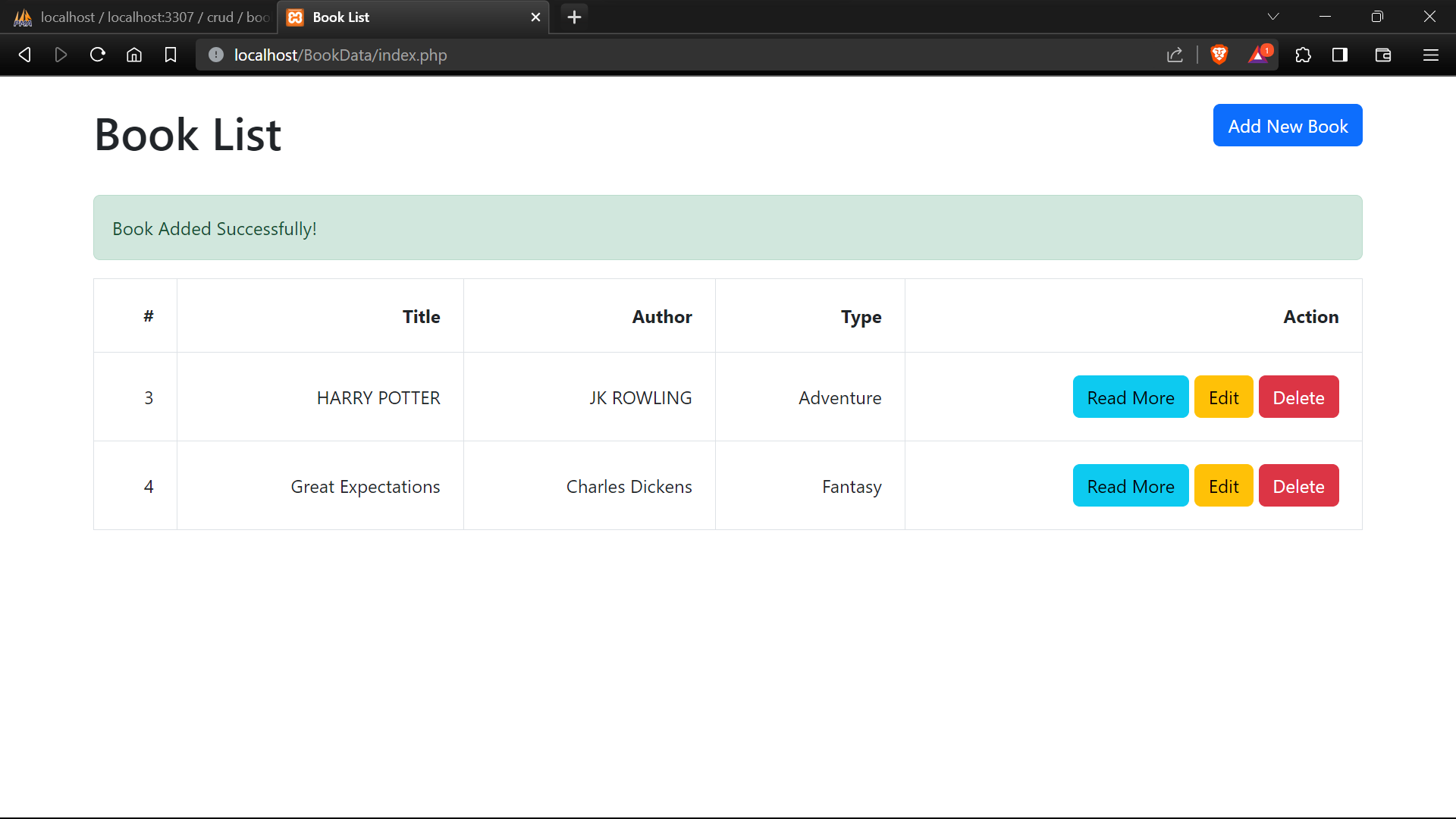
**PDO (PHP Data Objects):**

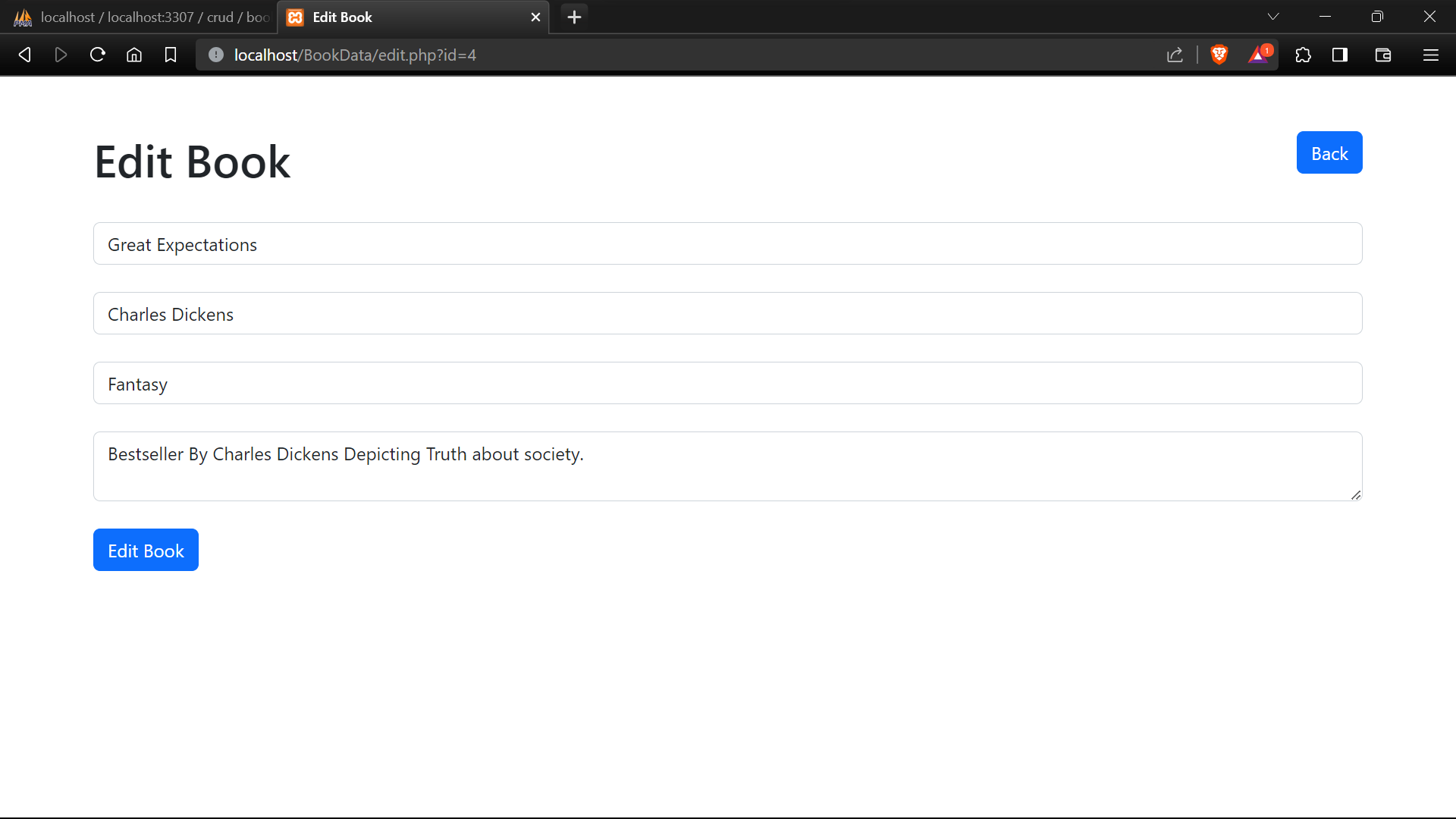
PDO is a more generic database abstraction layer for PHP that provides a consistent API for working with various database management systems.

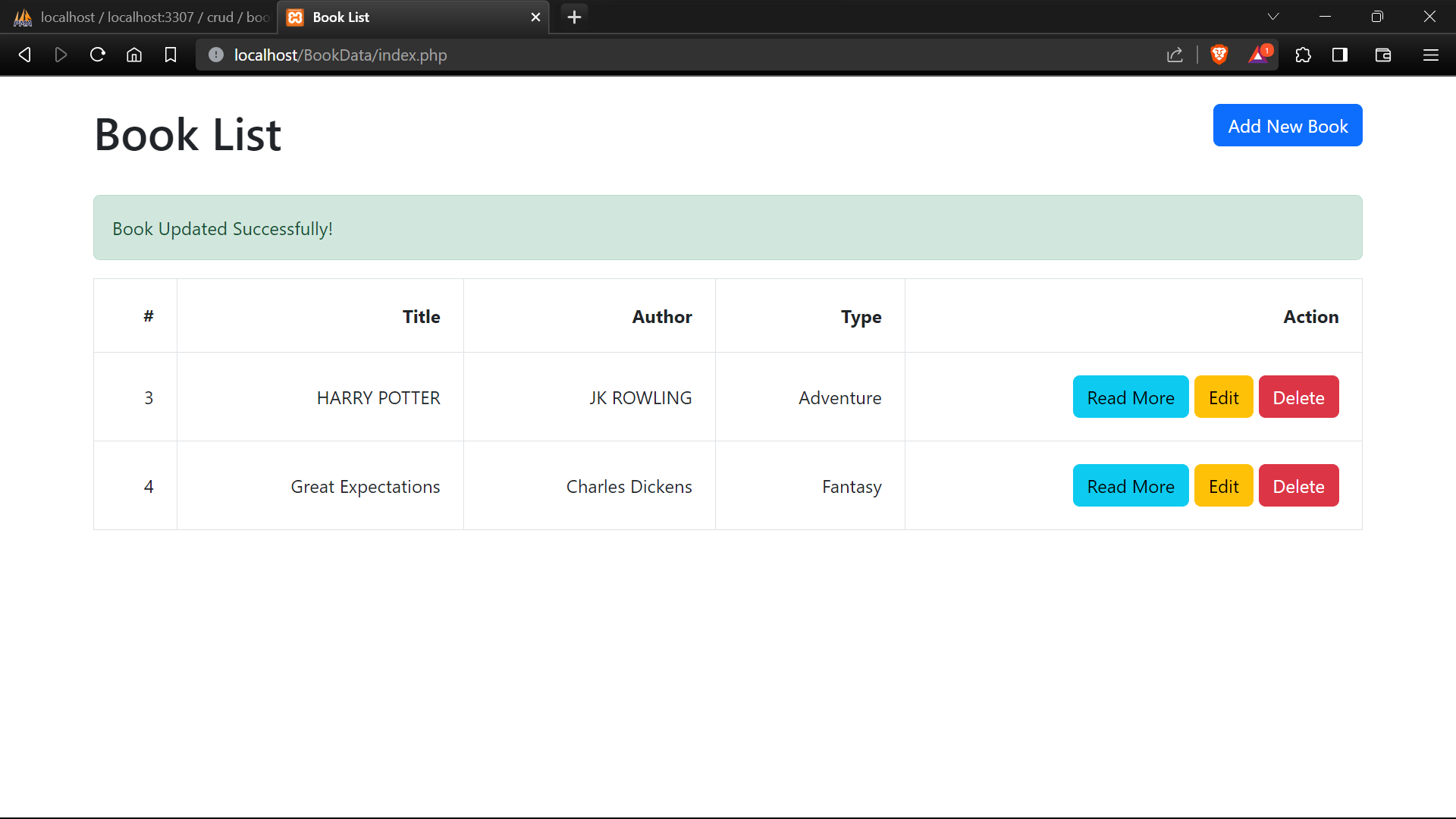
**OUTPUT-**

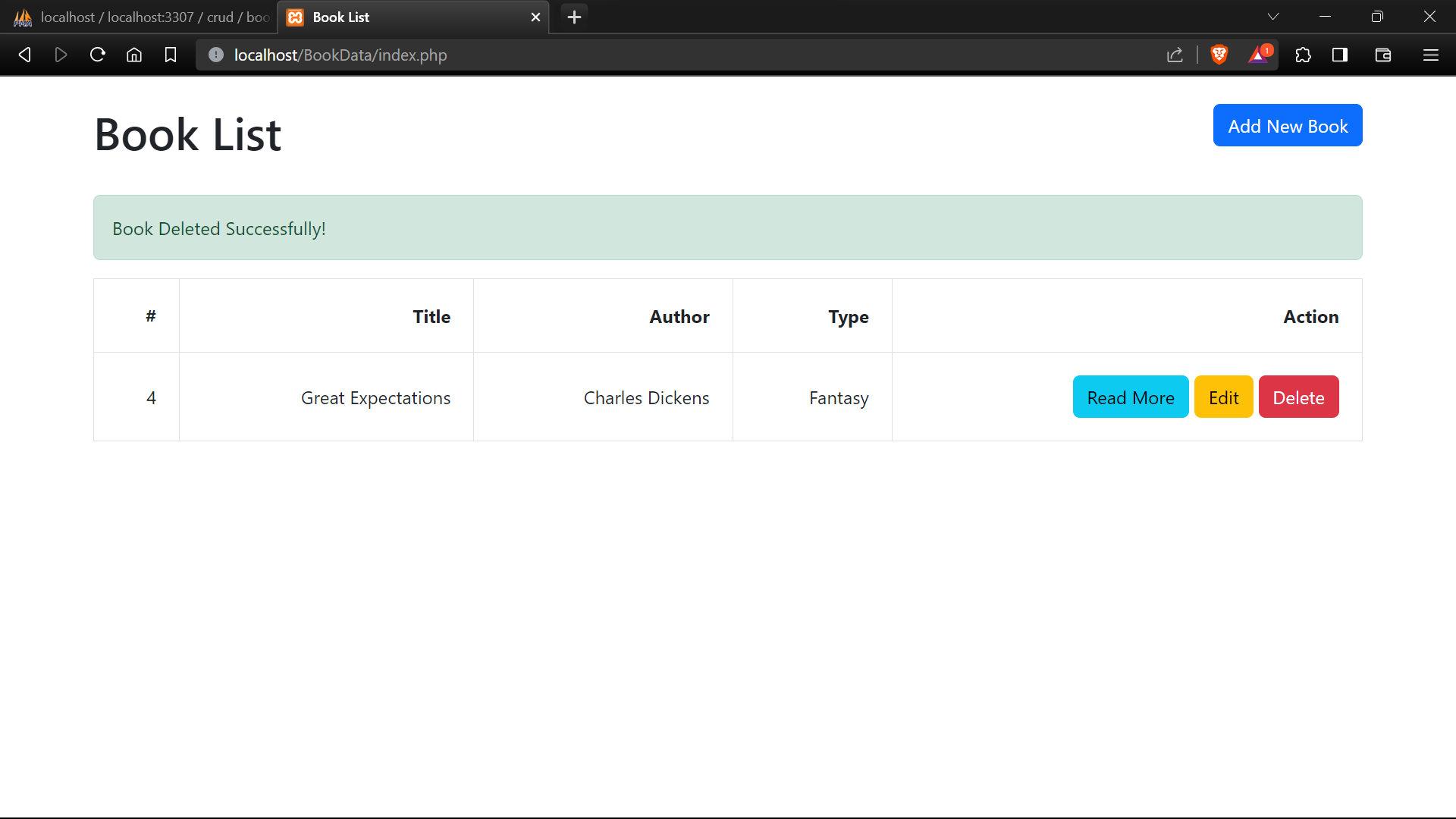












Github: <https://github.com/DarkShadows0401/FSD_LAB>