WEB DEVELOPMENT BACK-END

A Project Report

On

PHP Simple To-Do List

Submitted

By

V. Bindu Sri

(Batch 16)

Submitted

To

Submitted

On

9th August 2024

ABSTRACT

The "PHP Simple To-Do List" project is a web-based application designed to help users manage and organize their daily tasks efficiently. Built using PHP and HTML5, this application allows users to create, view, update, and delete tasks through a user-friendly interface. The primary goal of this project is to provide a simple yet effective tool for task management that can be easily integrated into daily routines.

The application leverages PHP for server-side processing, enabling dynamic interaction with a MySQL database where tasks are stored. HTML5 is used to structure the front-end interface, ensuring that the application is responsive and accessible across different devices. CSS is employed to enhance the visual appeal, providing a clean and modern look to the application.

Key features of the application include task creation with customizable details, task completion tracking, and the ability to remove or edit tasks as needed. The project emphasizes ease of use, with a straightforward navigation system and minimalistic design to ensure that users can focus on their tasks without unnecessary distractions.

This project serves as a foundational example of how PHP and HTML5 can be combined to create a functional, efficient, and user-friendly web application. It demonstrates the potential of using simple, open-source technologies to develop practical solutions for everyday needs.

OBJECTIVE

The primary objective of the "PHP Simple To-Do List" project is to develop a user-friendly web application that enables users to efficiently manage their daily tasks. This objective is achieved through the following specific goals:

1. Task Management: Provide users with the ability to create, view, update, and delete tasks in an organized manner.

2. User Interface Design: Develop a clean, intuitive, and responsive user interface using HTML5 and CSS, ensuring accessibility across various devices.

3. Server-Side Processing: Utilize PHP to handle server-side operations, including task management and interaction with a MySQL database for data storage and retrieval.

4. Data Persistence: Implement a reliable method for storing user tasks, ensuring that all data remains persistent across sessions.

5. Scalability and Extensibility: Design the application in a way that allows for easy extension of features in the future, such as adding user authentication, categorization of tasks, or integration with external services.

6. Performance Optimization: Ensure that the application runs efficiently, providing a seamless experience for users without delays or errors.

7. Security: Implement basic security measures to protect user data and prevent unauthorized access.

8. Demonstrate Web Development Best Practices: To showcase best practices in web development, including secure coding, proper documentation, and effective use of development tools, serving as a reference for future projects.

9. Facilitate Learning and Knowledge Sharing: To create a project that can serve as a learning resource for other developers, providing insights into web development with PHP, MySQL, and front-end technologies.

10. Encourage Community Contribution: To develop the application in a way that invites contributions from the open-source community, allowing other developers to enhance, extend, or customize the application.

11. Ensure Cross-Browser Compatibility: To develop the application to be compatible with all major web browsers, ensuring a consistent experience for users regardless of their choice of browser.

By achieving these objectives, the project aims to deliver a practical and effective tool for task management, demonstrating the capabilities of PHP and HTML5 in web application development. The objectives of the "PHP Simple To-Do List" project are centered around creating a functional, user-friendly, and scalable application that not only meets the immediate needs of task management but also serves as a valuable learning tool for developers. By promoting modular and reusable code, implementing robust error handling, and ensuring cross-browser compatibility, the project aims to provide a seamless user experience while adhering to best practices in web development. The use of version control and the focus on future scalability further underscore the project's commitment to maintaining a high standard of quality and facilitating collaboration. Additionally, the project is designed to be a foundation for future enhancements and an educational resource, encouraging knowledge sharing and community contribution. Through these objectives, the project aspires to be both a practical tool for users and an exemplary model for developers.

INTRODUCTION

In today's fast-paced world, effective task management is crucial for productivity and organization. The ability to keep track of daily tasks, prioritize them, and ensure their timely completion can significantly impact personal and professional success. Recognizing this need, the "PHP Simple To-Do List" project was conceived as a straightforward yet powerful tool to help users manage their tasks efficiently.

The "PHP Simple To-Do List" is a web-based application that allows users to create, update, view, and delete tasks through a simple, intuitive interface. It leverages the capabilities of PHP for server-side processing and a MySQL database for persistent data storage. HTML5 is used to structure the front-end interface, while CSS provides styling to enhance the visual appeal and usability of the application.

The development of this project not only provides a practical tool for task management but also serves as a learning experience in web development, particularly in using PHP and HTML5. It demonstrates how these technologies can be combined to create functional, user-friendly web applications. Additionally, the project offers a foundation for future enhancements, such as adding user authentication, task categorization, or integration with external services.

In summary, the "PHP Simple To-Do List" project addresses the fundamental need for task management in a simple, efficient, and user-friendly manner. It highlights the effectiveness of using PHP and HTML5 in developing web applications and provides a solid base for further development and customization.

Importance :

The "PHP Simple To-Do List" project holds significant importance in both practical use and as a learning tool. For users, it enhances productivity by providing a simple, intuitive platform for managing daily tasks, helping to organize and prioritize activities effectively. For developers, especially beginners, this project offers a foundational experience in web development, covering key concepts such as server-side scripting with PHP, front-end design using HTML5 and CSS, and database management with MySQL. It serves as a practical entry point into the world of web development, with the potential to be expanded into more complex applications. The project not only addresses a common real-world need but also exemplifies essential programming practices that can be applied to more advanced projects in the future.

Scope :

The scope of the "PHP Simple To-Do List" project encompasses the development of a basic task management application with core features such as adding, editing, and deleting tasks, all managed through a MySQL database for data persistence. The project integrates front-end technologies like HTML5 and CSS for structuring and styling, and PHP for server-side processing, creating a cohesive and responsive user interface. While the initial implementation

focuses on these core functionalities, it also includes basic security measures like input validation and error handling. However, the scope does not extend to more advanced features such as user authentication, task categorization, or notification systems, which are identified as potential future enhancements. The application is designed primarily for individual users seeking a simple task management solution, as well as beginner developers looking to gain hands-on experience in web development.

Outline :

The "PHP Simple To-Do List" project follows a structured outline, beginning with the requirement analysis to understand user needs and the feasibility study to assess the project's viability. It then progresses through system architecture and design, where the application's overall structure, database schema, and user interface are planned. The development phase involves front-end and back-end coding, integrating PHP with MySQL to manage tasks efficiently. This is followed by testing, including unit, integration, and user acceptance testing, to ensure the application functions correctly. The final phases include deployment on a local or live server, with ongoing maintenance and potential future enhancements such as user authentication and task categorization. This outline ensures a systematic approach to building a functional and extendable to-do list application.

METHODOLOGY

The development of the "PHP Simple To-Do List" application followed a structured methodology to ensure a systematic and efficient approach to building the application. The process involved several key phases, each focusing on different aspects of the project.

1. Requirement Analysis :

- **Objective**: The first step was to clearly define the requirements for the to-do list application. This involved understanding the core functionalities needed, such as task creation, viewing, updating, and deletion.
- **Outcome**: A detailed list of features was created, serving as the foundation for the project's design and development.

2. Design :

- **UI/UX Design**: The user interface was designed to be simple and intuitive, focusing on usability. Wireframes were created to visualize the layout of the application, ensuring a clean and user-friendly design.
- **Database Design**: A MySQL database schema was designed to store tasks. The schema included a single table to manage task-related information such as task ID, title, description, status, and timestamps.

3. Development :

- **Front-End Development**: HTML5 and CSS were used to create the structure and styling of the application. The design was kept minimalistic to ensure that users could easily navigate the application.
- **Back-End Development**: PHP was utilized to handle server-side operations, such as processing form submissions, interacting with the MySQL database, and rendering dynamic content on the front end.
- **Database Integration**: PHP's MySQLi functions were employed to connect to the MySQL database, perform CRUD (Create, Read, Update, Delete) operations, and ensure data persistence.

4. Implementation :

- **Task Creation**: A form was implemented to allow users to input task details, which are then stored in the database.
- **Task Viewing**: Tasks stored in the database are dynamically displayed in a table format, allowing users to view their tasks at a glance.

- **Task Updating**: An edit feature was included to enable users to modify existing tasks, with changes reflected immediately in the database.
- **Task Deletion**: A delete function was implemented, allowing users to remove tasks from the list, with the database updated accordingly.

5. Testing :

- **Unit Testing**: Individual components of the application, such as form submissions and database interactions, were tested to ensure they functioned correctly.
- **Integration Testing**: The entire system was tested to ensure that all components worked together seamlessly.
- **User Testing**: A small group of users was asked to interact with the application to provide feedback on usability and functionality. Any issues identified were addressed promptly.

6. Deployment :

- The final application was deployed on a local server for testing purposes. Instructions were prepared for deployment on a live server, making the application accessible from any web browser.

7. Documentation :

- Comprehensive documentation was prepared, detailing the design decisions, code structure, and instructions for future developers on how to extend or modify the application.

6. Maintenance and Future Enhancements :

- Post-deployment, the application was monitored for any bugs or issues that may arise. Regular updates and patches were planned to keep the application secure and up-to-date.
- Potential features for future versions were identified, including user authentication, task categorization, and integration with external services like calendars or reminders. The system's architecture was designed to be flexible enough to accommodate these enhancements without requiring major overhauls.

7. Technology Stack Selection :

- Multiple technologies were evaluated for their suitability to the project. PHP was chosen for its simplicity and widespread use in web development, MySQL for its robust and reliable database capabilities, and HTML5/CSS for structuring and styling the user interface.
- Various tools and libraries were considered for development, such as XAMPP for local server setup, phpMyAdmin for database management, and version control systems like Git for tracking changes and collaboration.

8. Security Considerations :

- Input data was validated and sanitized to prevent SQL injection and cross-site scripting (XSS) attacks. PHP's built-in functions, such as mysqli_real_escape_string() and htmlspecialchars(), were employed.
- Sessions were securely managed using PHP's native session handling. Measures such as setting session timeouts and regenerating session IDs on login were implemented to prevent session hijacking.
- Though not implemented in the initial version, a future enhancement plan was drafted for adding user authentication. This includes secure login forms, password hashing with password_hash(), and role-based access control.

9. Code Organization and Modularity :

- The code was organized into modules, separating concerns such as database access, user interface logic, and business logic. This made the codebase more maintainable and scalable.
- Functions and classes were designed to be reusable, reducing code duplication. For instance, a common function was used to handle database connections across the application.

10. Error Handling and Debugging :

- PHP's error logging mechanisms were enabled to capture and log errors during development. Custom error handlers were also implemented to ensure graceful degradation in case of failures.
- Debugging tools such as Xdebug and browser-based developer tools were used to identify and resolve issues quickly. Breakpoints, watch variables, and step execution were employed during the debugging process.

.

CODE

Style.css :

```
* {
    margin: 0;
    padding: 0;
    font-family: monospace;
}

body {
    background-color: #f0f4f8;
}

nav {
    background-color: #003366;
    margin-bottom: 50px;
}

.heading {
    text-decoration: none;
    color: #f0f4f8;
    letter-spacing: 1px;
    font-size: 30px;
    font-weight: 900;
    display: flex;
    justify-content: center;
    padding: 10px;
}

.container {
    display: flex;
    flex-direction: row;
    justify-content: space-between;
    align-items: flex-start;
    padding: 20px;
}
.input-area {
    margin-bottom: 30px;
    width: 40%;
}
input {
    width: 100%;
    height: 40px;
```

```
    border: 2px solid #0066cc;
    border-radius: 10px;
    padding: 10px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.2);
    font-size: 16px;
    background-color: #ffffff;
}

.btn {
    width: 100px;
    height: 40px;
    border-radius: 50px;
    border: 1px solid #003366;
    color: white;
    font-weight: 900;
    background-color: #003366;
    cursor: pointer;
}

table {
    border: 1px solid #cccccc;
    width: 55%;
    padding: 20px;
    border-radius: 10px;
    background-color: #ffffff;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
th {
    font-size: 20px;
    text-align: start;
    padding: 15px 10px;
    background-color: #0066cc;
    color: white;
}

th:nth-child(4) {
    text-align: center;
}

tr.border-bottom td {
    border-bottom: 1pt dashed #aaaaaa;
}
```

```
tbody {
    padding: 3px;
    font-size: 16px;
}

td {
    padding: 10px;
}

.action {
    display: flex;
    flex-direction: row;
    justify-content: center;
    align-items: center;
    gap: 20px;
}

.btn-completed, .btn-remove {
    text-decoration: none;
    color: none;
    background-color: none;
    padding: 5px 10px;
    border-radius: 5px;
    cursor: pointer;
}

.btn-remove {
    background-color: none;
}
```

Index.php :

```
<!DOCTYPE html>
<html lang="en">

<head>
<title>Todo List</title>
<link rel="stylesheet"
                type="text/css" href="style.css" />
</head>
<body>
<nav>
        <a class="heading" href="#">ToDo App</a>
</nav>
<div class="container">
        <div class="input-area">
        <form method="POST" action="add_task.php">
                <input type="text" name="task"
                                placeholder="ENTER YOUR TASK" required />
                <button class="btn" name="add">
                        Add Task
                </button>
        </form>
        </div>
        <table class="table">
        <thead>
                <tr>
                <th>Sno</th>
                <th>Tasks</th>
                <th>Status</th>
                <th>Action</th>
                </tr>
        </thead>
        <tbody>
        <?php
          require 'config.php';
                $fetchingtasks =
mysqli_query($db, "SELECT * FROM `task` ORDER BY `task_id` ASC")
or die(mysqli_error($db));
                $count = 1;
                while ($fetch = $fetchingtasks->fetch_array()) {
                ?>
                <tr class="border-bottom">
```

```
                <td>
                        <?php echo $count++? </td>
                 <td>
                      <?php echo $fetch['task'] ?>
                 </td>
                 <td>
                        <?php echo $fetch['status'] ?>
                 </td>
                 <td colspan="2" class="action">
                        <?php
                                if ($fetch['status'] != "Done")
                                {
                                        echo
  '<a href="update_task.php?task_id=' . $fetch['task_id'] . '"
  class="btn-completed">✅</a>';    }
                        ?>
<a href= "delete_task.php?task_id=<?php echo $fetch['task_id'] ?>"
                        class="btn-remove">❌</a>
                 </td>
                 </tr>
                 <?php
                                }
                        ?>
          </tbody>
          </table>
</div>
</body>
</html>
```

Config.php :

```
<?php
$db = mysqli_connect("localhost", "root", "", "todo")
   or
   die("Connection failed: " . mysqli_connect_error());
?>
```

Add_task.php :

```
<?php
require_once 'config.php';
if (isset($_POST['add'])) {
        if ($_POST['task'] != "") {
                $task = $_POST['task'];

                $addtasks = mysqli_query($db,
                        "INSERT INTO `task` VALUES('', '$task', 'Pending')")
                        or
                        die(mysqli_error($db));
                header('location:index.php');
        }
}
?>
```

Delete_task.php :

```
<?php
require_once 'config.php';

if ($_GET['task_id']) {
	$task_id = $_GET['task_id'];

	$deletingtasks = mysqli_query($db,
		"DELETE FROM `task` WHERE `task_id` = $task_id")
		or
		die(mysqli_error($db));

	header("location: index.php");
}
?>
```

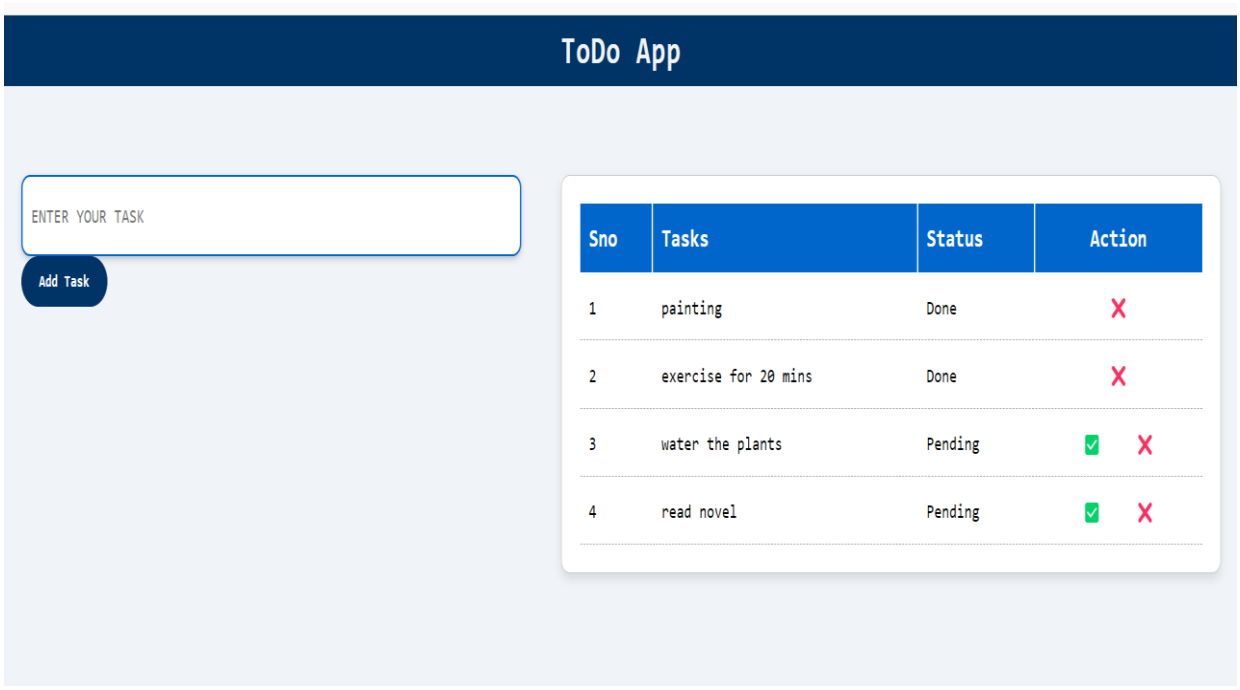
Update_task.php :

```
<?php
require_once 'config.php';

if ($_GET['task_id'] != "") {
	$task_id = $_GET['task_id'];

	$updatingtasks =
		mysqli_query($db,
			"UPDATE `task` SET `status` = 'Done' WHERE `task_id` = $task_id")
		or
		die(mysqli_error($db));
	header('location: index.php');
}
?
```

OUTPUT


ToDo App
ENTER YOUR TASK
Add Task
Sno
Tasks
Status
Action
1
painting
Done
2
exercise for 20 mins
Done
3
water the plants
Pending
4
read novel
Pending


ToDo App
complete 1stop back end project
Add Task
Sno
Tasks
Status
Action
1
painting
Done
2
exercise for 20 mins
Done
3
water the plants
Pending
4
read novel
Pending

ToDo App

ENTER YOUR TASK

Please fill out this field.

Add Task

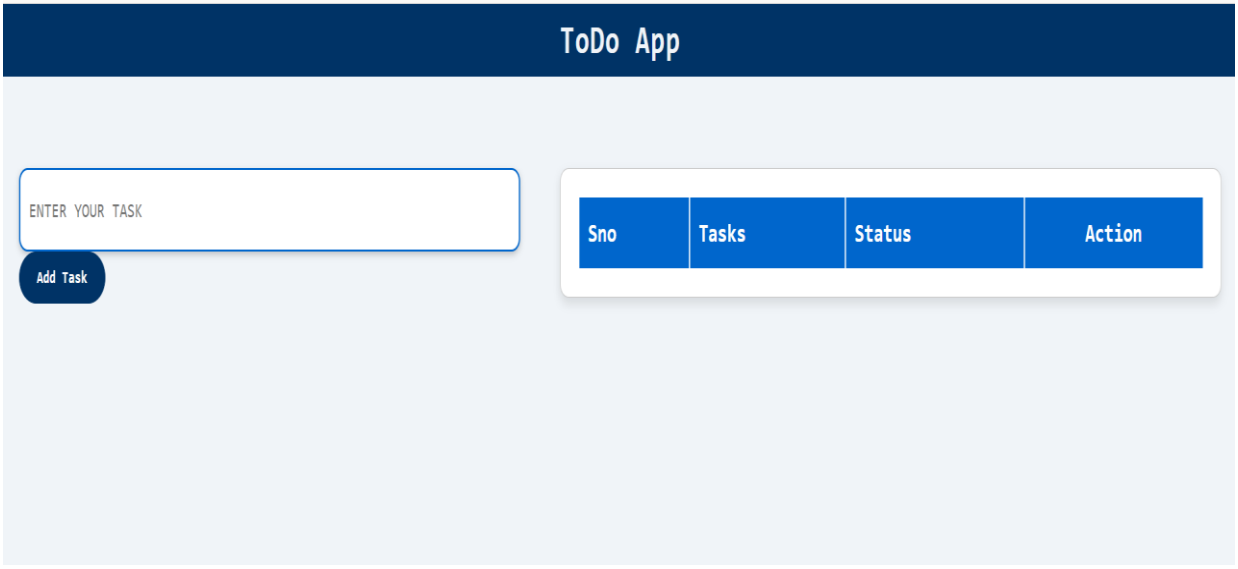
Sno	Tasks	Status	Action
1	painting	Done	X
2	exercise for 20 mins	Done	X
3	water the plants	Done	X
4	read novel	Pending	☑ X
5	complete 1stop back end project	Pending	☑ X

ToDo App

ENTER YOUR TASK

Add Task

Sno	Tasks	Status	Action
1	painting	Done	X
2	exercise for 20 mins	Done	X
3	water the plants	Done	X
4	read novel	Done	X
5	complete 1stop back end project	Done	X


ToDo App
ENTER YOUR TASK
Add Task
Sno
Tasks
Status
Action

CONCLUSION

The "PHP Simple To-Do List" project successfully achieves its goal of providing a straightforward and efficient tool for managing daily tasks. Through the use of PHP for server-side processing, HTML5 for structuring the front-end, and MySQL for data storage, the application demonstrates how these technologies can be effectively combined to create a functional and user-friendly web application.

Throughout the development process, a focus on simplicity and usability ensured that the application meets the needs of users who require a basic task management solution without unnecessary complexity. The intuitive interface allows users to quickly add, view, update, and delete tasks, making it easier to stay organized and productive.

Key Achievements:

- **Successful Integration of Technologies:** The project effectively combined PHP, HTML5, CSS, and MySQL to create a fully functional to-do list application.
- **User-Friendly Interface:** The application features a clean, intuitive design that allows users to easily add, view, update, and delete tasks.
- **Data Persistence:** The implementation of a MySQL database ensures that user tasks are stored reliably and can be retrieved across sessions.
- **Responsive Design:** The front-end was designed to be responsive, ensuring usability across different devices, including desktops, tablets, and smartphones.
- **Efficient Task Management:** The application provides core task management features that work seamlessly, contributing to better organization and productivity for users.

Future Improvements:

User Authentication: Adding a user authentication system would allow multiple users to maintain their individual task lists securely.

- **Task Categorization and Prioritization:** Implementing features such as categorization and priority levels would enhance the application's ability to manage more complex task lists.
- **Task Notifications and Reminders:** Introducing notifications and reminders for tasks could help users stay on top of important deadlines.
- **Search and Filtering Options:** Adding the ability to search and filter tasks would improve usability, especially for users managing a large number of tasks.
- **Mobile Optimization:** Further optimizing the application for mobile devices would ensure a better user experience on smaller screens.

- **Integration with External Services:** Future versions of the application could integrate with external services such as Google Calendar or email to provide more comprehensive task management solutions.

In conclusion, the "PHP Simple To-Do List" project is a robust and practical application that not only serves its intended purpose but also provides a solid foundation for further enhancements. The key achievements of the project demonstrate the effective use of web technologies in solving real-world problems, while the identified areas for future improvement offer exciting possibilities for expanding the application's functionality and usability.