National School of Sciences

Lainchaur, Kathmandu

"TO DO LIST MANAGEMENT SYSTEM"

Submitted By:

Name:- Ankit Khatri KC

Roll No:- 719

Class:- 12

Section:- M16

Group:- A

Submitted To:

Department of Computer Science

National School of Science

4 March 2025

ACKNOWLEDGMENT

It is my good fortune to get very cooperative teacher **Mr.Santosh Bhat**. During the work in progress, his support, helpfulness and constant encouragement kept me motivated in research work. He provided invaluable interest, guidance during the course of the work. I have not only learned but also got important suggestions regarding scientific writing and other related matters. I am very much grateful to him.

I would like to thank all the faculty members who have provided encouragement and suggestions during the course of work and special thanks to lab assistants.

Ankit Khatri KC (719)

CERTIFICATE OF APPROVAL

It is certified that **Mr. Ankit Khatri KC** has carried out the project work entitled **"To Do List Management System"**.

This project is the result of **his** endeavors and research. It is finalized under our guidance and supervision in the academic year 2024 – 2025.

Head of Department
Er.Sabin Raj Bagale

Department of Computer Science
National School of Science
Lainchaur, Kathmandu, Nepal

Supervisor
Mr.Santosh Bhat

Department of Computer Science
National School of Science
Lainchaur, Kathmandu, Nepal

External,NEB

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

CRUD stands for Create, Read, Update, and Delete. The four CRUD functions can be called by users to perform different types of operations on selected data within the database. This could be accomplished using code or through a graphical user interface. Let's review each of the four components in-depth to fully appreciate their collective importance in facilitating database interactions. The following are:

- Create:

  The create function allows users to create a new record in the database. In the SQL relational database application. A user can create a new row and populate it with data that corresponds to each attribute, but only an administrator might be able to add new attributes to the table itself.

- Read:

  The read function is similar to a search function. It allows users to search and retrieve specific records in the table and read their values. Users may be able to find desired records using keywords, or by filtering the data based on customized criteria.

- Update:

  The update function is used to modify existing records that exist in the database. To fully change a record, users may have to modify information in multiple fields.

- Delete:

  The delete function allows users to remove records from a database that is no longer needed. Both SQL. and Oracle HCM Cloud have a delete function that allows users to delete one or more records from the database.

1.2 Data Flow Diagram

1.3 Problem Definition

In the fast-paced digital age, managing tasks effectively is essential for productivity and time management. Many individuals and teams rely on traditional methods, such as paper or manual notes, to keep track of their tasks. However, this method can be disorganized, prone to mistakes, and difficult to manage when the task list grows. Without a digital system, tracking task progress, updating deadlines, or prioritizing tasks becomes a cumbersome process. This project aims to develop a digital To-Do List Management System (TDMS) that simplifies task management, enabling users to stay organized, prioritize tasks effectively, and track progress with ease.

1.4 Objectives

The objectives of this project are:

- To create a user-friendly To-Do List Management System that allows users to easily add new tasks, update existing ones, and delete tasks that are no longer needed.
- To implement a system that enables users to mark tasks as completed or pending, giving them better control over task prioritization and tracking.
- To provide an efficient and scalable solution that simplifies task management, helping users stay on top of deadlines, priorities, and progress.

1.5 Features

The features of the To-Do List Management System are as follows:

- **Add New Task**: Users can easily create new tasks by entering a title, description, and setting a deadline. This feature allows users to quickly add new tasks, making the system efficient and easy to use.

- **Update Task**: Users can modify any existing task, whether it's changing the deadline, updating the description, or adjusting the priority. This feature ensures that tasks can be adjusted to reflect new information or shifting priorities.
- **Delete Task**: Users have the option to remove tasks that are no longer relevant or necessary. This feature helps in keeping the task list clean and manageable.
- **Mark Task as Completed**: Once a task is finished, users can mark it as completed. This feature helps track progress and keeps the task list up-to-date with what's been accomplished.
- **Mark Task as Pending**: If a task has not yet been completed, users can mark it as pending. This helps in identifying tasks that are still in progress and need attention.
- **Task Listing and Filtering**: Users can view all tasks in a simple list format, with clear indicators of the task status (completed, pending). Tasks can be filtered by status, priority, or deadline to help users focus on what's most important.
- **User Interface**: The system will feature a clean, easy-to-navigate interface, making it simple for users to manage their tasks. Whether adding, updating, or deleting tasks, the system will offer a smooth and intuitive experience.

1.6 Feasibility

1.6.1 Technical Feasibility

The To-Do List Management System can be easily developed using widely available web technologies. The frontend will use HTML, CSS, and JavaScript to create a responsive and interactive user interface. For the backend, PHP will be used to handle CRUD (Create, Read, Update, Delete) operations, allowing users to add, update, and delete tasks. A MySQL database will store task data, such as task name, description, deadline, and status. The use of PHP sessions will ensure secure user authentication, allowing users to have personalized task lists. The system will be hosted on cloud platforms like AWS or Heroku to ensure scalability and easy access for users.[1]

1.6.2 Operational Feasibility

The system is designed to be simple and easy to use. The interface will be intuitive, allowing users to quickly add new tasks, update existing ones, or mark tasks as completed. The ability to filter tasks by status (completed, pending) or by deadline will help users prioritize their work. The system will ensure that users can efficiently manage their tasks without overwhelming them with complex features. Security will be a priority, with encrypted passwords and secure login procedures. The system will integrate seamlessly into the user's workflow, making it an essential tool for productivity.

1.6.3 Financial Feasibility

The main development costs will involve hiring developers, designers, and potential project managers to bring the system to life. Operational costs will include hosting and server maintenance, which will be scalable based on user needs. By using open-source technologies such as PHP and MySQL, licensing costs are kept minimal. Once developed, the system will save users time by providing an easy way to manage tasks, ultimately improving productivity. If desired, a premium version could be introduced with additional features, such as team collaboration or advanced reporting, generating potential revenue through subscriptions or one-time payments.

1.7 System Requirement

Listed below are the software and hardware requirements of our project.

- MySQL as DBMS for books and and user details.
- Laptop for writing codes.
- XAMPP for simulating database and server.
- Code editor like : VS code and Sublime text.

(a) Xampp

(b) VS Code

(c) Sublime Text

(d) Laptop

Figure 1.1: Requirements of the project

CHAPTER 2

DATABASE CONNECTION

2.1 About

To connect the To-Do List Management System (TDMS) to a MySQL database using XAMPP, Apache, and MySQL, we follow these steps. First, we install XAMPP, which includes Apache for hosting and MySQL for database management. Once XAMPP is installed, we start Apache and MySQL from the XAMPP control panel. We can manage our database using phpMyAdmin at `http://localhost/phpmyadmin/`.

In the application, we will use PHP to connect to the MySQL database, where the data will be stored and managed.

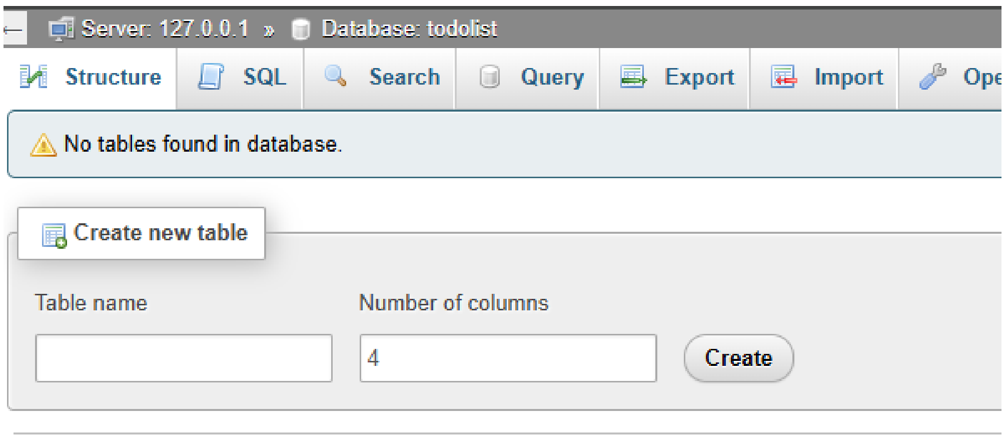
2.2 Creating Database

Follow these simple steps to create the database and table for the TDMS:

1. Open phpMyAdmin: - Go to `http://localhost/phpmyadmin/` in our web browser.

2. Create the Database: - Click on the "Databases" tab. - Enter `todolist` as the database name. - Click "Create".

3. Create the Table: - After creating the database, click on it. - Click "Create Table" and enter `tasks` as the table name. - Define the following columns: - `id` (`INT`, `AUTO_INCREMENT`, `PRIMARY KEY`) - `task` (`VARCHAR(255)`, `NOT NULL`) - `date` (`DATE`, `NOT NULL`) - `status` (`TINYINT`, `DEFAULT 0`)

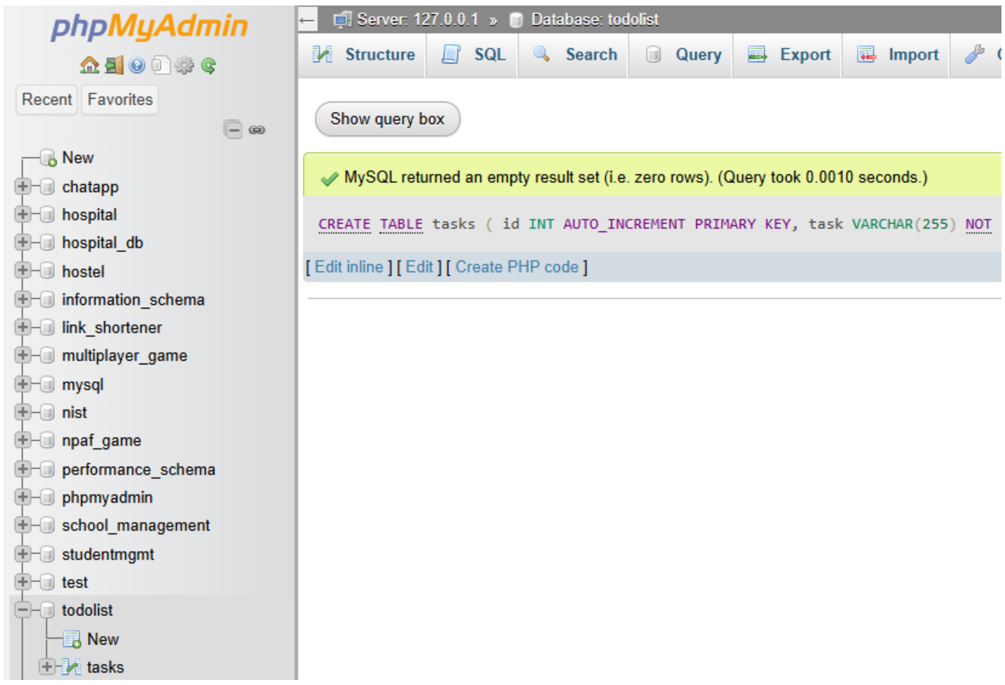


No tables found in database.

Create new table

Table name	Number of columns
	4

Figure 2.1: Database created successfully



MySQL returned an empty result set (i.e. zero rows). (Query took 0.0010 seconds.)

```
CREATE TABLE tasks ( id INT AUTO_INCREMENT PRIMARY KEY, task VARCHAR(255) NOT
```

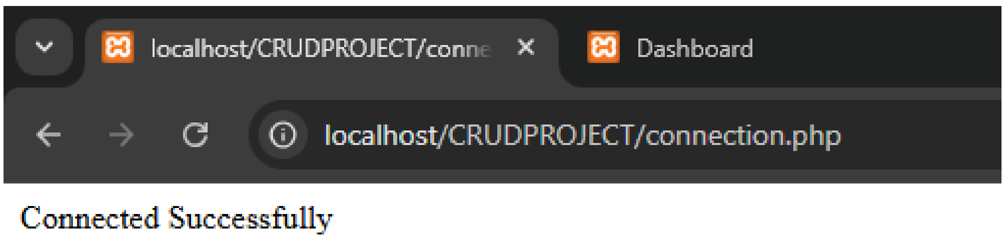
Figure 2.2: Database in phpMyAdmin

2.3 Database Connection Code

After setting up the database, we use the following PHP code to connect to it.[2]

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

```
$password = "";
$dbname = "todolist";

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

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

else{
    echo("Connected Successfully");
}
?>
```


localhost/CRUDPROJECT/conne
Dashboard
localhost/CRUDPROJECT/connection.php

```
Connected Successfully
```

Figure 2.3: Connection message

2.4 SQL for Database Creation

To create the database and table, we use the following SQL commands:

1. Create the Database:

```
CREATE DATABASE todolist;
```

2. Create the Tasks Table:

```
CREATE TABLE tasks (
    id INT AUTO_INCREMENT PRIMARY KEY,
    task VARCHAR(255) NOT NULL,
    date DATE NOT NULL,
    status TINYINT DEFAULT 0
);
```

This completes the setup for our To-Do List Management System's database and connection.

CHAPTER 3

LOGIN PAGE

3.1 About

The Login Page serves as the entry point to the To-Do List Management System, ensuring secure access for users. Built using HTML, CSS, and JavaScript, the login functionality validates user credentials on the client side before allowing access to the system. Upon entering their credentials (username and password), users are authenticated using JavaScript. If the credentials match the predefined values, the user is redirected to the dashboard. Otherwise, an error message is displayed.

3.1.1 Form Validation

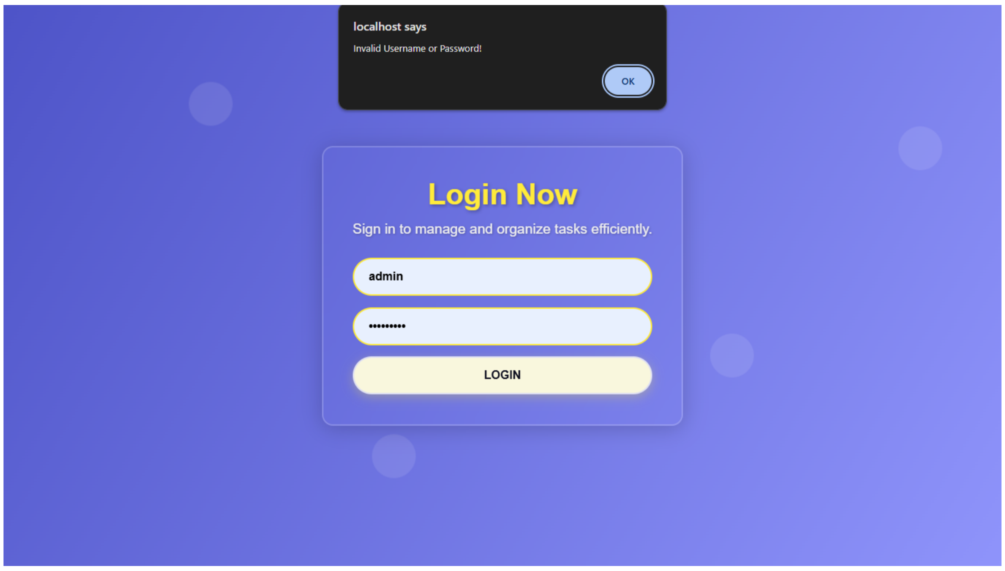
Form validation ensures that the user inputs are correct and meet the required criteria before submission. In this system, JavaScript is used to validate the login form. The validation checks if the username and password fields are filled and if they match the predefined values. If the inputs are valid, the user is redirected to the dashboard. If not, an error message is displayed.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
       scale=1.0">
    <title>Login Now</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/
       bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="style/index.css">
```

```
</head>
<body>
    <div class="login">
        <div class="login-content">
            <h1>Login Now</h1>
            <p>Sign in to manage and organize tasks efficiently.</p>

            <form class="mt-4" id="loginForm">
                <div class="mb-3">
                    <input
                        type="text"
                        class="form-control form-control-lg"
                        placeholder="Username"
                        id="username"
                        required>
                </div>
                <div class="mb-3">
                    <input
                        type="password"
                        class="form-control form-control-lg"
                        placeholder="Password"
                        id="password"
                        required>
                </div>
                <button type="submit" class="btn btn-outline-light w
                   -100">Login</button>
            </form>
        </div>
    </div>

    <div class="bg-animation">
        <span></span>
        <span></span>
        <span></span>
        <span></span>
    </div>

    <script>
        document.getElementById("loginForm").addEventListener("
            submit", function(event) {
            event.preventDefault();

            const username = document.getElementById("username").
                value;
            const password = document.getElementById("password").
                value;
            if (username === "groupa" && password === "groupa@123")
                {
                alert("Login Successful!");
                window.location.href = "dashboard.php";
            } else {
                alert("Invalid Username or Password!");
            }
        });
    </script>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
        alpha3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

3.1.2 Valid and Invalid Login

I. **Invalid Login:** When the user provides incorrect credentials (username or password), the form validation displays an alert message: "Invalid Username or Password!".

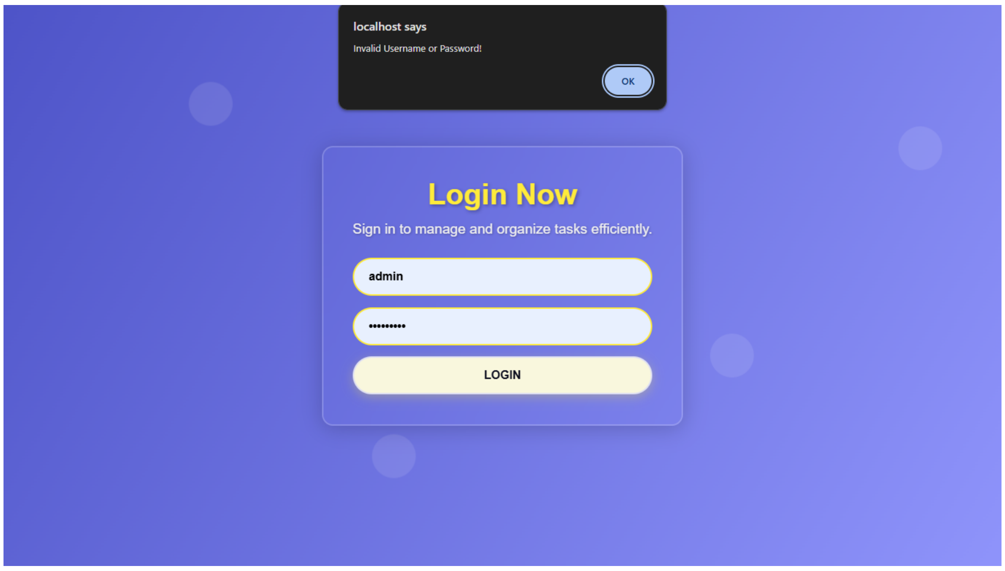


localhost says

Invalid Username or Password!

Figure 3.1: Invalid Login Case

II. **Valid Login:** When the user provides the correct credentials (username: "groupa", password: "groupa@123"), the form validation displays an alert message: "Login Successful!" and redirects the user to the dashboard.



localhost says

Invalid Username or Password!

Figure 3.2: Valid Login Case

3.2 Login Page Code

The login page is implemented using HTML, CSS, and JavaScript. The HTML structure includes a form for username and password inputs, while JavaScript handles the validation logic. Bootstrap is used for styling the form and making it responsive.

3.2.1 JavaScript Validation Code

The JavaScript code validates the login form by checking if the username and password match the predefined values. If the credentials are correct, the user is redirected to the dashboard. Otherwise, an error message is displayed.

```
document.getElementById("loginForm").addEventListener("submit",
    function(event) {
    event.preventDefault();

    const username = document.getElementById("username").value;
    const password = document.getElementById("password").value;
    if (username === "groupa" && password === "groupa@123") {
        alert("Login Successful!");
        window.location.href = "dashboard.php";
    } else {
        alert("Invalid Username or Password!");
    }
});
```

CHAPTER 4

HOME PAGE

4.1 About

The Home Page of the To-Do List Management System serves as the landing page for users, providing an introduction to the system and a gateway to access the login page. The page features a clean and modern design with a hero section that includes a welcome message and a login button. The system is designed to help users efficiently manage their tasks and stay organized.

4.1.1 Hero Section

The hero section is the main focus of the home page. It includes:

- A heading: "To-Do List Management"
- A brief description: "Welcome to our To-Do List Management System, proudly created by Group A!"
- A login button that redirects users to the login page.

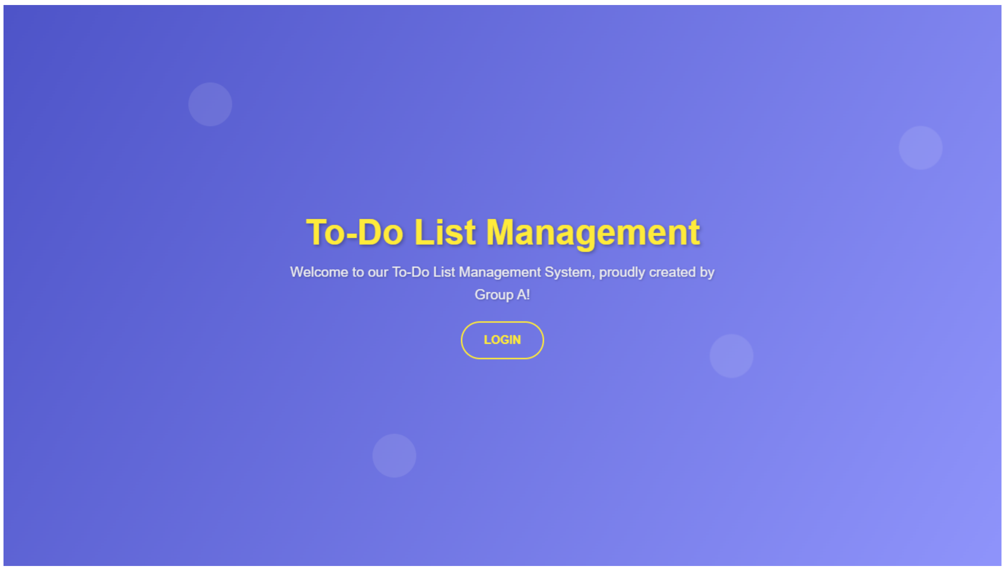

To-Do List Management
Welcome to our To-Do List Management System, proudly created by Group A!
LOGIN

Figure 4.1: Home Page of To-Do List Management System

4.2 Home Page Code

The home page is implemented using HTML, CSS, and Bootstrap for styling. Below is the necessary code for the home page.

4.2.1 HTML Code

The home page consists of a hero section with a welcome message and a login button. Bootstrap is used for styling the button and layout.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
       scale=1.0">
    <title>To-Do List Management</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/
       bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="style/index.css">
</head>
<body>
    <div class="hero">
        <div class="hero-content">
            <h1>To-Do List Management</h1>
            <p>Welcome to our To-Do List Management System, proudly
                created by Group A!</p>
            <div class="login-btn">
                <a href="login.php" class="btn btn-outline-light">
                    Login</a>
            </div>
        </div>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
        alpha3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

4.2.2 Explanation of the Code

- **Hero Section**: The hero section contains a heading, a description, and a login button styled using Bootstrap.
- **Responsive Design**: The page is designed to be responsive, ensuring it looks good on all devices.

CHAPTER 5

CRUD OPERATION

5.1 About

The To-Do List Management System implements CRUD (Create, Read, Update, Delete) operations to manage tasks efficiently. These operations allow users to add new tasks, view existing tasks, update task details, and delete tasks. Additionally, the system categorizes tasks into Pending Tasks and Completed Tasks, providing a clear overview of task progress.

5.2 Create

5.2.1 About

The Create operation allows users to add new tasks to the system. Users can input the task description and select a due date. The task is then stored in the database.[3]

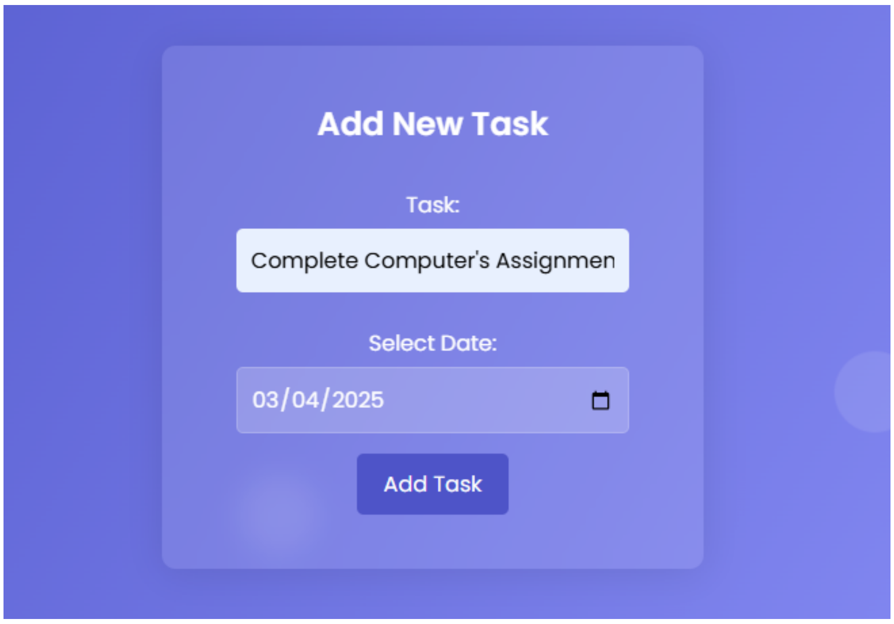
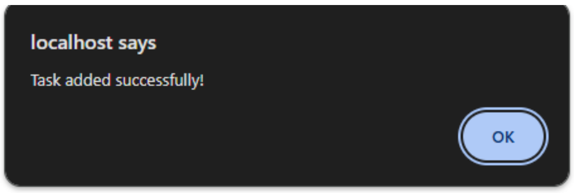
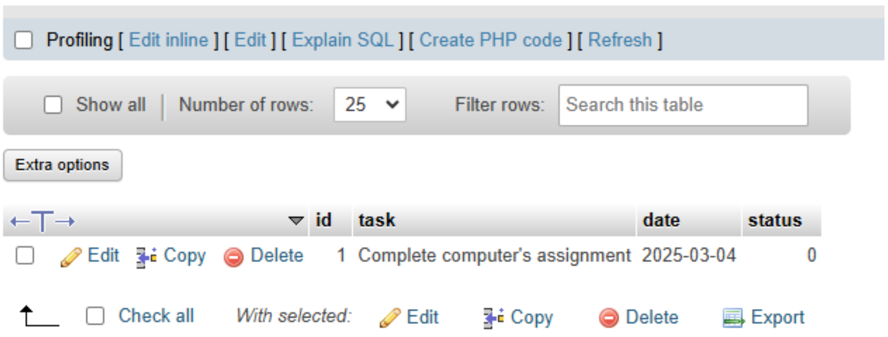


Figure 5.1: Form for adding a new task


localhost says

Task added successfully!

Figure 5.2: Success message after adding a task


Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Extra options

	id	task	date	status
Edit Copy Delete	1	Complete computer's assignment	2025-03-04	0

Check all
With selected: Edit Copy Delete Export

Figure 5.3: Database after adding a task

5.2.2 Code

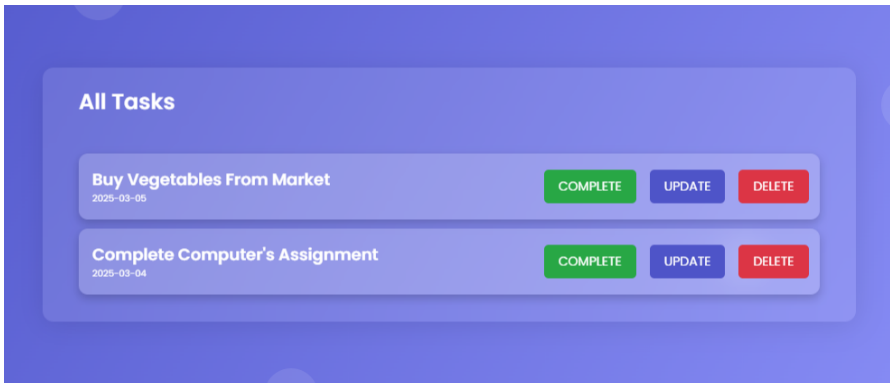
The following PHP code handles the creation of new tasks:

```
<?php
session_start();

if (isset($_GET['logout'])) {
    session_destroy();
    header("Location: login.php");
    exit();
}

include('connection.php');

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $task = ($_POST['task']);
    $date = $_POST['date'];
    $sql = "INSERT INTO tasks (task, date) VALUES ('$task', '$date')
       ";
    if ($conn->query($sql) === TRUE) {
        echo "<script>alert('Task added successfully!');</script>";
    } else {
        echo "<script>alert('Error adding task: " . $conn->error . "
           ');</script>";
    }
}
$conn->close();
?>
```

5.3 Read

5.3.1 About

The Read operation retrieves and displays all tasks from the database. Tasks are displayed in a list format, showing the task description and due date.

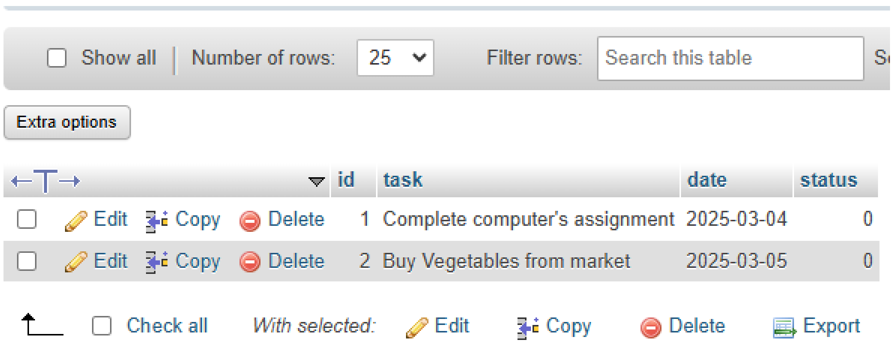


All Tasks

Buy Vegetables From Market
2025-03-05

Complete Computer's Assignment
2025-03-04

Figure 5.4: List of all tasks



	id	task	date	status
	1	Complete computer's assignment	2025-03-04	0
	2	Buy Vegetables from market	2025-03-05	0

Figure 5.5: Database of all tasks

5.3.2 Code

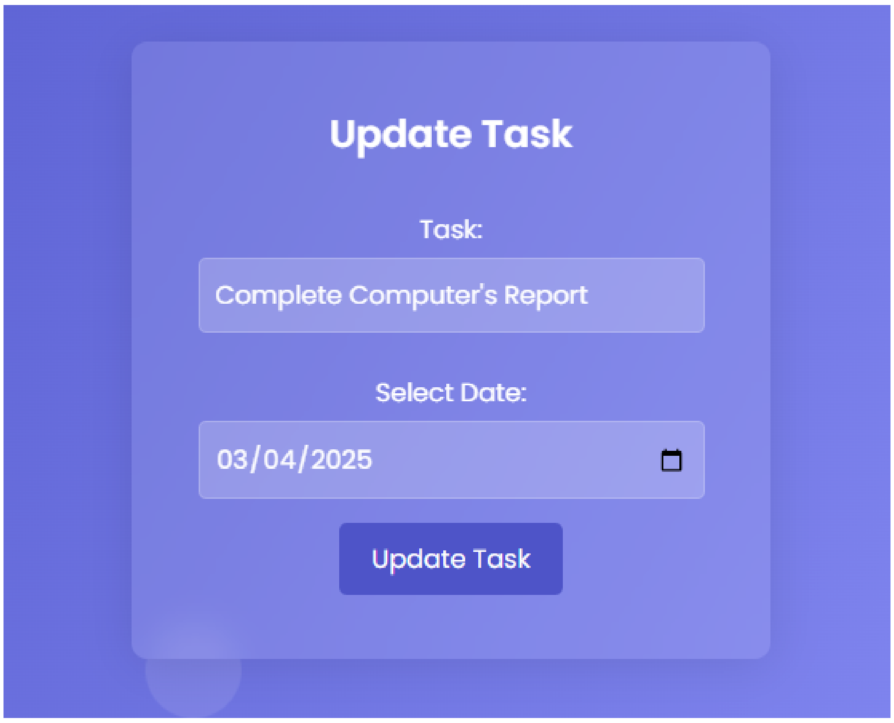
The following PHP code retrieves and displays tasks:

```
<?php
session_start();

if (isset($_GET['logout'])) {
    session_destroy();
    header("Location: login.php");
    exit();
}

include('connection.php');

$sql = "SELECT * FROM tasks ORDER BY date DESC";
$result = $conn->query($sql);

$conn->close();
?>
```

5.4 Update

5.4.1 About

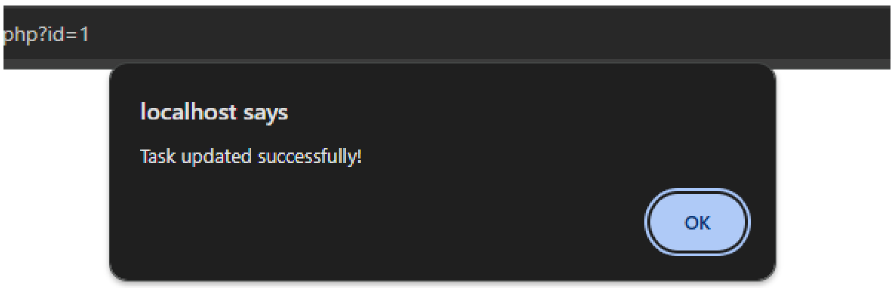
The Update operation allows users to modify the details of an existing task. Users can update the task description or due date.



Complete Computer's Report

03/04/2025

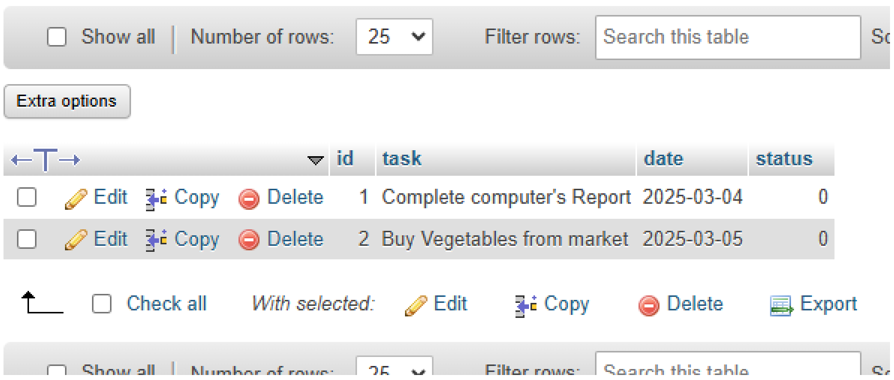
Figure 5.6: Form for updating a task


php?id=1

localhost says

Task updated successfully!

Figure 5.7: Success message after updating a task



	id	task	date	status
Edit Copy Delete	1	Complete computer's Report	2025-03-04	0
Edit Copy Delete	2	Buy Vegetables from market	2025-03-05	0

Figure 5.8: Database after updating task

5.4.2 Code

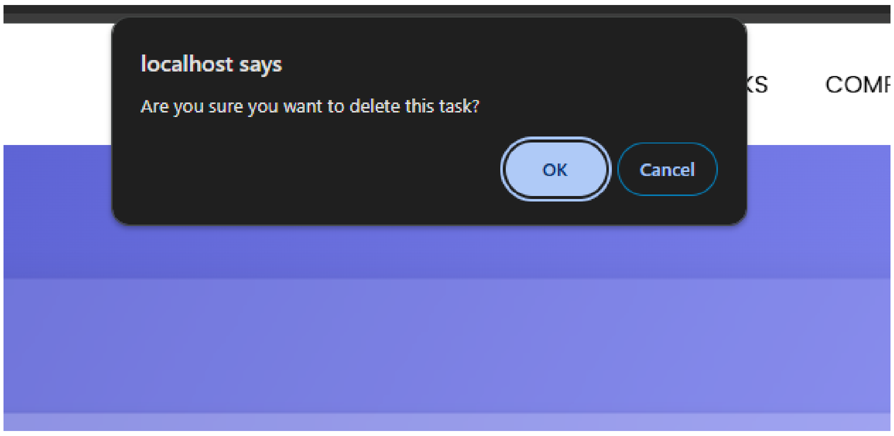
The following PHP code handles the updating of tasks:

```
<?php
session_start();

if (isset($_GET['logout'])) {
    session_destroy();
    header("Location: login.php");
    exit();
}

include('connection.php');

$taskId = $_GET['id'];
$sql = "SELECT * FROM tasks WHERE id = $taskId";
$result = $conn->query($sql);
$taskDetails = $result->fetch_assoc();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $task = ($_POST['task']);
    $date = $_POST['date'];
    $sql = "UPDATE tasks SET task = '$task', date = '$date' WHERE
       id = $taskId";
    echo $conn->query($sql) ? "<script>alert('Task updated
       successfully!'); window.location='dashboard.php';</script>" :
        "<script>alert('Error updating task.');</script>";
}
$conn->close();
?>
```

5.5 Delete

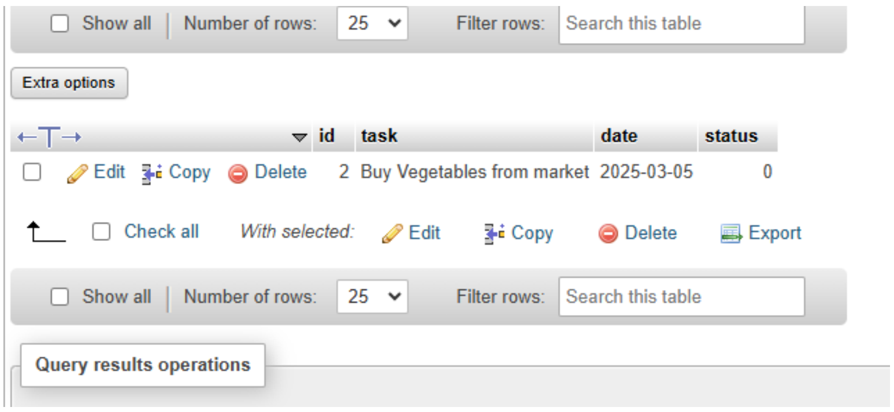
5.5.1 About

The Delete operation allows users to remove tasks from the system. Users can delete a task permanently from the database.


localhost says

Are you sure you want to delete this task?

Figure 5.9: Confirmation dialog for deleting a task



				id	task	date	status
☐	Edit	Copy	Delete	2	Buy Vegetables from market	2025-03-05	0

Figure 5.10: Database after deleting a task

5.5.2 Code

The following PHP code handles the deletion of tasks:

```
<?php
session_start();

if (isset($_GET['logout'])) {
    session_destroy();
    header("Location: login.php");
    exit();
}

include('connection.php');

if (isset($_GET['delete'])) {
    $task_id = $_GET['delete'];
    $conn->query("DELETE FROM tasks WHERE id = $task_id");
    header("Location: dashboard.php");
    exit();
}
?>
```

5.6 Additional Features

5.6.1 Pending Tasks

Pending Tasks are tasks that have not been marked as completed. Users can view, update, or mark these tasks as completed.[4]

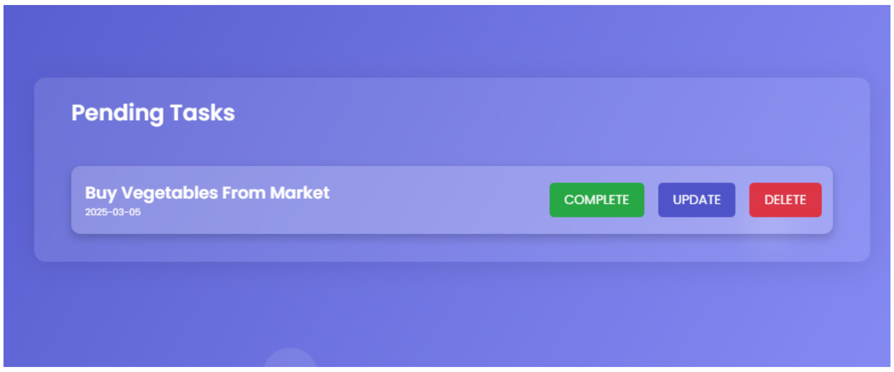
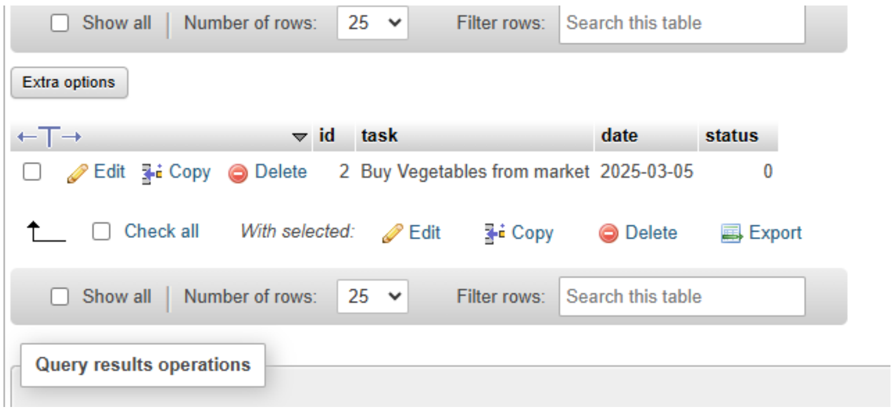


Figure 5.11: List of pending tasks



	id	task	date	status
Edit Copy Delete	2	Buy Vegetables from market	2025-03-05	0

Figure 5.12: Database after updating task

5.6.2 Code for Pending Tasks

The following PHP code retrieves and displays pending tasks:

```
<?php
session_start();

if (isset($_GET['logout'])) {
    session_destroy();
    header("Location: login.php");
    exit();
}

include('connection.php');

$sql = "SELECT * FROM tasks WHERE status = 0 ORDER BY date DESC";
$result = $conn->query($sql);

$conn->close();
?>
```

5.6.3 Completed Tasks

Completed Tasks are tasks that have been marked as completed. Users can view or delete these tasks.

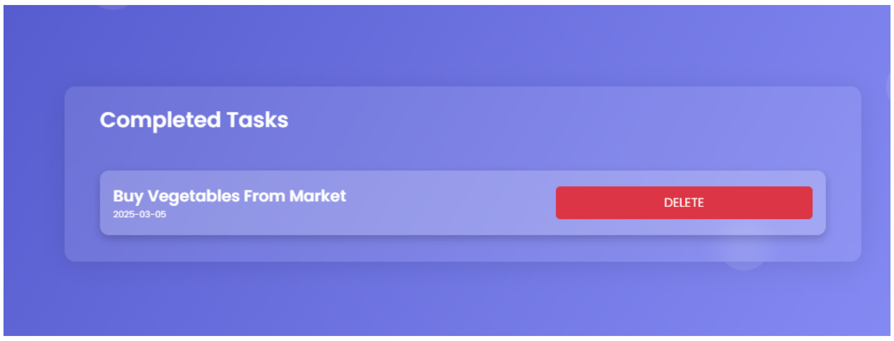
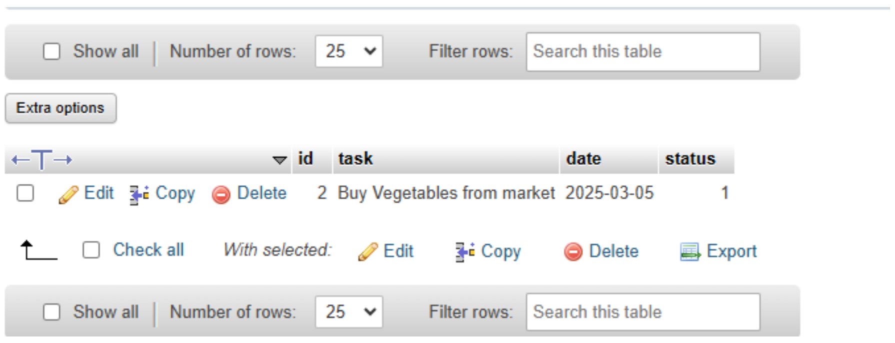


Figure 5.13: List of completed tasks


Show all | Number of rows: 25 Filter rows: Search this table
Extra options
Edit Copy Delete
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table

id	task	date	status
2	Buy Vegetables from market	2025-03-05	1

Figure 5.14: Database after completing task

5.6.4 Code for Completed Tasks

The following PHP code retrieves and displays completed tasks:

```php
<?php
session_start();

if (isset($_GET['logout'])) {
    session_destroy();
    header("Location: login.php");
    exit();
}

include('connection.php');

$sql = "SELECT * FROM tasks WHERE status = 1 ORDER BY date DESC";
$result = $conn->query($sql);

if (isset($_GET['delete'])) {
    $id = $_GET['delete'];
    $conn->query("DELETE FROM tasks WHERE id = $id");
    header("Location: completedtasks.php");
    exit();
}

$conn->close();
?>
```

CHAPTER 6

EPILOGUE

6.1 CONCLUSION

The To-Do List Management System is a practical and efficient tool designed to help users organize and manage their tasks effectively. By implementing CRUD operations (Create, Read, Update, Delete), the system allows users to add, view, modify, and delete tasks seamlessly. The integration of PHP and MySQL ensures a robust backend, while the use of HTML, CSS, and JavaScript provides a responsive and user-friendly interface. The system also categorizes tasks into **Pending** and **Completed**, offering users a clear overview of their progress. This project demonstrates how simple yet powerful technologies can be combined to create a functional and scalable task management solution.

6.2 LIMITATIONS

Despite its functionality, the system has certain limitations:

I. **Basic User Authentication**: The system lacks advanced authentication mechanisms like Two-Factor Authentication (2FA) or password recovery options.

II. **No Multi-User Support**: The system is designed for individual use and does not support multiple users or collaborative task management.

III. **Limited Notifications**: There are no automated reminders or notifications for upcoming or overdue tasks.

IV. **No Cloud Integration**: The system is locally hosted, limiting access to a single device or network.

V. **Basic UI/UX**: While functional, the user interface could be enhanced with modern design elements and animations for a better user experience.

6.3 FUTURE ENHANCEMENT

To make the system more versatile and user-friendly, the following enhancements can be implemented:

I. **Multi-User Support**: Introduce user roles (e.g., Admin, User) and allow multiple users to collaborate on shared tasks.

II. **Automated Notifications**: Add email or SMS reminders for task deadlines and overdue tasks.

III. **Cloud Integration**: Host the system on the cloud to enable remote access and synchronization across devices.

IV. **Advanced Authentication**: Implement features like Two-Factor Authentication (2FA), password recovery, and CAPTCHA for enhanced security.

V. **Task Prioritization**: Allow users to prioritize tasks (e.g., High, Medium, Low) and sort them accordingly.

VI. **Mobile App Development**: Develop a mobile application for Android and iOS platforms to provide users with on-the-go access.

VII. **AI-Powered Suggestions**: Use machine learning algorithms to suggest task deadlines or categorize tasks based on user behavior.

VIII. **Integration with Calendars**: Sync tasks with popular calendar applications like Google Calendar or Outlook.

REFERENCES

[1] Prachandra Shrestha. *Essentials of Computer Science*. Asmita Books Publisher and Distributors, 2078.

[2] Dipak Pudasaini. *Computer Science II*. Buddha Publication, 2024.

[3] Freedial. How to create crud operation in php and mysql, 2024. Accessed: [2021].

[4] Step by Step (YouTube). How to create crud operations in php and mysql, 2024. Accessed: [2021].