# Assignment No.6

## Program-:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Contact Form</title>

</head>

<body>

<h1>Contact Us</h1>

<form action="contact.php" method="POST">

<label for="name">Name:</label><br>

<input type="text" id="name" name="name" required><br><br>

<label for="email">Email:</label><br>

<input type="email" id="email" name="email" required><br><br>

<label for="message">Message:</label><br>

<textarea id="message" name="message" rows="5" required></textarea><br><br>

<input type="submit" value="Submit">

</form>

</body>

</html>

<?php

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// Get form data

$name = htmlspecialchars(trim($\_POST['name']));

$email = htmlspecialchars(trim($\_POST['email']));

$message = htmlspecialchars(trim($\_POST['message']));

// Validate form data

if (!empty($name) && !empty($email) && !empty($message)) {

// Email configuration

$to = ["prathameshjadhav362@gmail.com](mailto:prathameshjadhav362@gmail.com)";

$subject = "New contact form submission from $name";

$body = "Name: $name\nEmail: $email\n\nMessage:\n$message";

$headers = "From: $email\r\n";

// Send email

if (mail($to, $subject, $body, $headers)) { echo "Message sent successfully!";

} else {

echo "Failed to send message. Please try again later.";

}

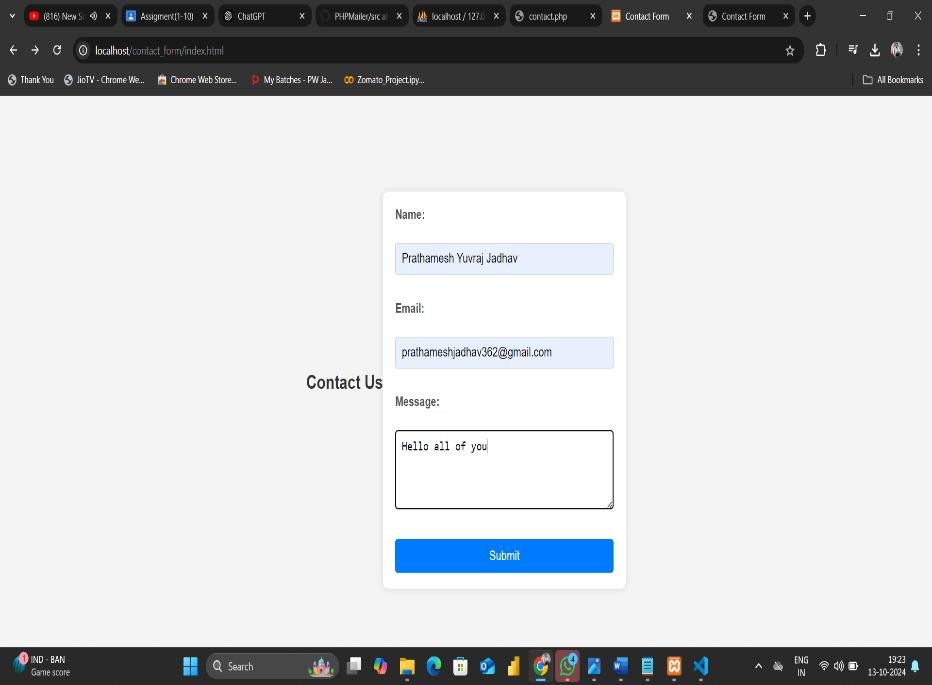
} else {

echo "All fields are required!";

}

}

?>

//Output-:

# Assignment No.7

## Program-:

<?php

// MySQL database credentials

$servername = "localhost"; // Usually 'localhost'

$username = "root"; // Database username

$password = ""; // Database password

$database = "Sales"; // Database name

// Create connection to the MySQL database

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

// Check if connection was successful if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// SQL query to select all products from the table

$sql = "SELECT \* FROM Products";

$result = $conn->query($sql);

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Product List</title>

<style> table {

width: 50%;

border-collapse: collapse; margin: 50px auto;

}

th, td { padding: 10px;

border: 1px solid #ddd; text-align: center;

}

th {

background-color: #f4f4f4;

}

</style>

</head>

<body>

<h2 style="text-align: center;">Products List</h2>

<table>

<tr>

<th>ID</th>

<th>Name</th>

<th>Price</th>

<th>Description</th>

</tr><?php

if ($result->num\_rows > 0) {

// Loop through and display each row of results while ($row = $result->fetch\_assoc()) {

echo "<tr>";

echo "<td>" . $row['P\_ID'] . "</td>";

echo "<td>" . $row['Name'] . "</td>";

echo "<td>" . $row['Price'] . "</td>";

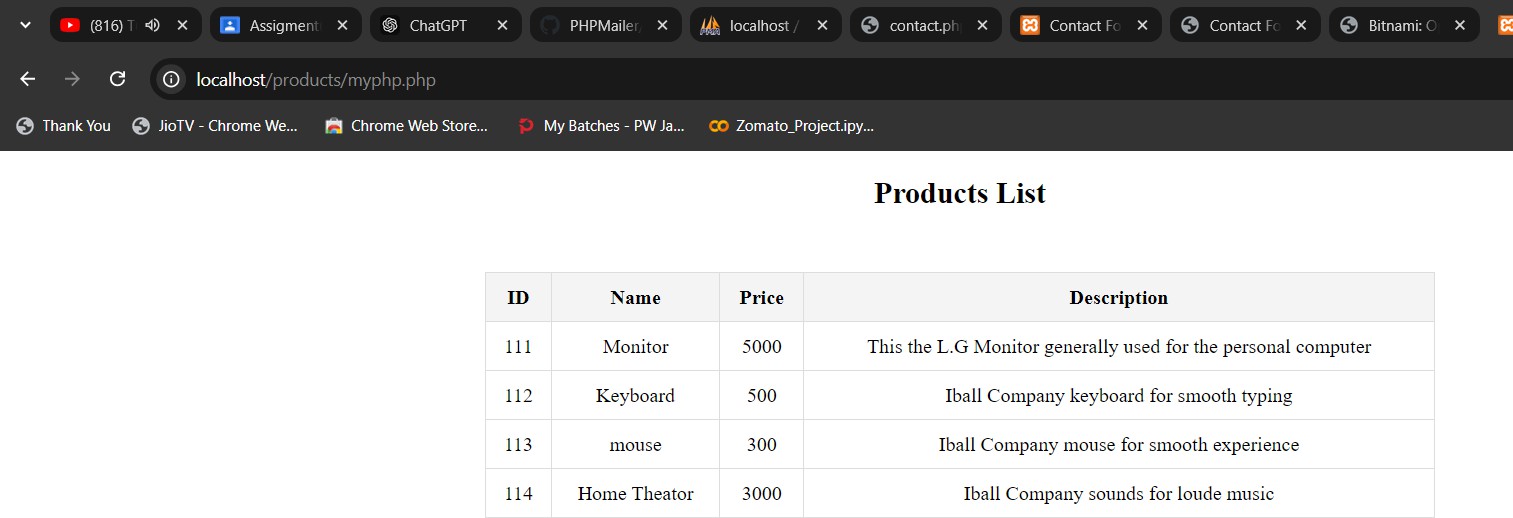
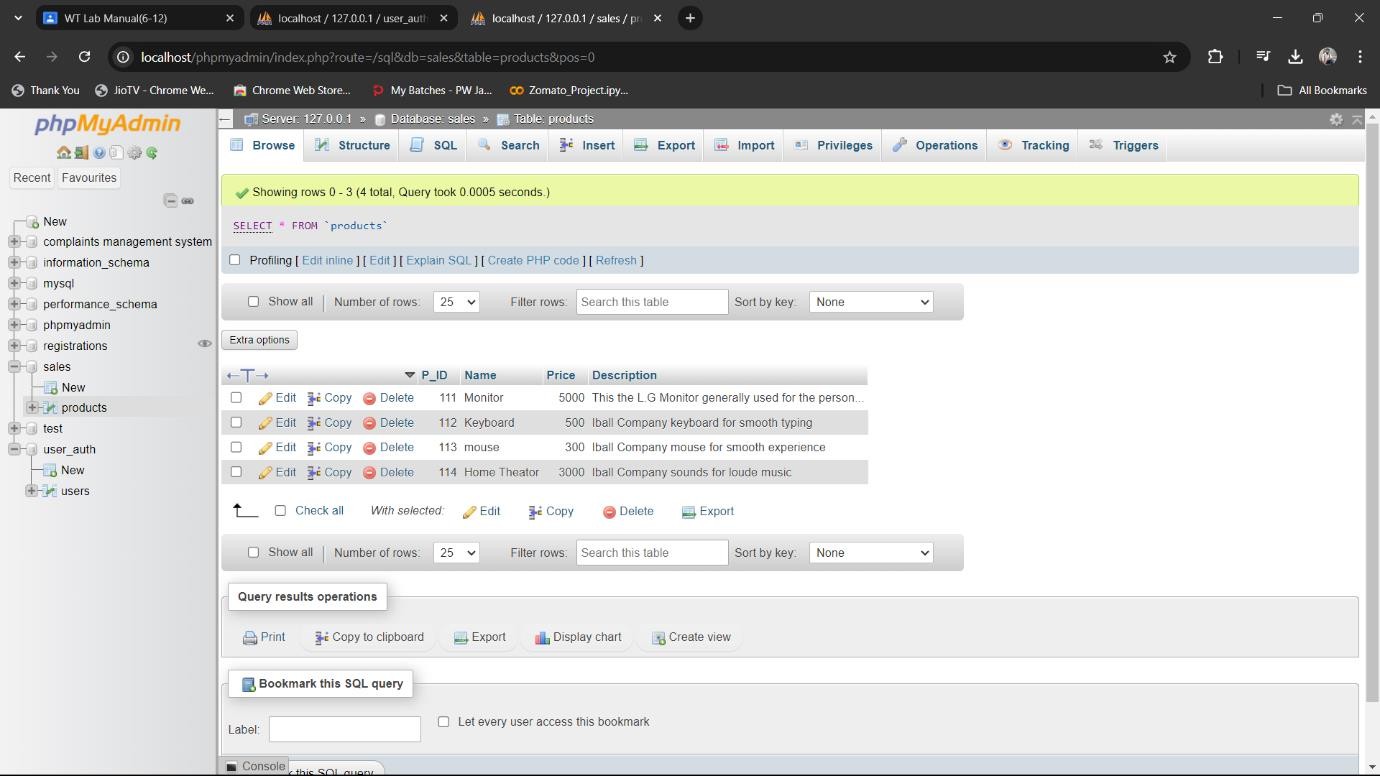
echo "<td>" . $row['Description'] . "</td>"; echo "</tr>";}

} else {

echo "<tr><td colspan='4'>No products found.</td></tr>";}

$conn->close();?>

</table>



</body>

</html>

Output-:

# Assignment No.8

## Program-:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Contact Form with Validation</title>

<link rel="stylesheet" href="style.css"> <!-- Link to CSS for styling -->

</head>

<body>

<form id="contactForm">

<h1>Contact Us</h1>

<label for="name">Name:</label>

<input type="text" id="name" name="name" placeholder="Your name" required>

<small class="error-message" id="nameError"></small>

<label for="email">Email:</label>

<input type="email" id="email" name="email" placeholder="Your email" required>

<small class="error-message" id="emailError"></small>

<label for="message">Message:</label>

<textarea id="message" name="message" placeholder="Your message" rows="5" required></textarea>

<small class="error-message" id="messageError"></small>

<input type="submit" value="Submit">

</form>

<script src="script.js"></script> <!-- Link to JavaScript for form validation -->

</body>

</html>

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif; background-color: #f4f4f4; display: flex;

justify-content: center; align-items: center;

height: 100vh;

}

form {

background-color: #fff; padding: 20px;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

width: 100%;

max-width: 400px;

}

h1 {

text-align: center;

margin-bottom: 20px; font-size: 24px;

}

label {

display: block;

margin-bottom: 8px; font-weight: bold;

}

input[type="text"], input[type="email"], textarea {

width: 100%; padding: 10px;

margin-bottom: 10px; border: 1px solid #ccc; border-radius: 4px;

}

input[type="submit"] { width: 100%; padding: 12px;

background-color: #007bff; color: white;

border: none; border-radius: 4px; font-size: 16px; cursor: pointer;

}

input[type="submit"]:hover { background-color: #0056b3;

}

.error-message { color: red;

font-size: 12px; display: none;

}

// Form and input elements

const form = document.getElementById('contactForm'); const nameInput = document.getElementById('name');

const emailInput = document.getElementById('email');

const messageInput = document.getElementById('message');

// Error message elements

const nameError = document.getElementById('nameError'); const emailError = document.getElementById('emailError');

const messageError = document.getElementById('messageError');

// Helper function to show error message

function showError(input, errorElement, message) { errorElement.innerText = message;

errorElement.style.display = 'block'; input.style.borderColor = 'red';

}

// Helper function to clear error message function clearError(input, errorElement) {

errorElement.innerText = '';

errorElement.style.display = 'none'; input.style.borderColor = 'green';

}

// Name validation (min 3 characters)

nameInput.addEventListener('input', () => { const nameValue = nameInput.value.trim(); if (nameValue.length < 3) {

showError(nameInput, nameError, 'Name must be at least 3 characters');

} else {

clearError(nameInput, nameError);

}

});

// Email validation (basic pattern check)

emailInput.addEventListener('input', () => { const emailValue = emailInput.value.trim();

const emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]+$/; if (!emailPattern.test(emailValue)) {

showError(emailInput, emailError, 'Please enter a valid email address');

} else {

clearError(emailInput, emailError);

}

});

// Message validation (min 10 characters)

messageInput.addEventListener('input', () => { const messageValue = messageInput.value.trim(); if (messageValue.length < 10) {

showError(messageInput, messageError, 'Message must be at least 10 characters');

} else {

clearError(messageInput, messageError);

}

});

// Final form submission validation

form.addEventListener('submit', (e) => { const nameValue = nameInput.value.trim(); const emailValue = emailInput.value.trim();

const messageValue = messageInput.value.trim(); if (nameValue.length < 3) {

showError(nameInput, nameError, 'Name must be at least 3 characters'); e.preventDefault();

}

if (!/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(emailValue)) {

showError(emailInput, emailError, 'Please enter a valid email address'); e.preventDefault();

}

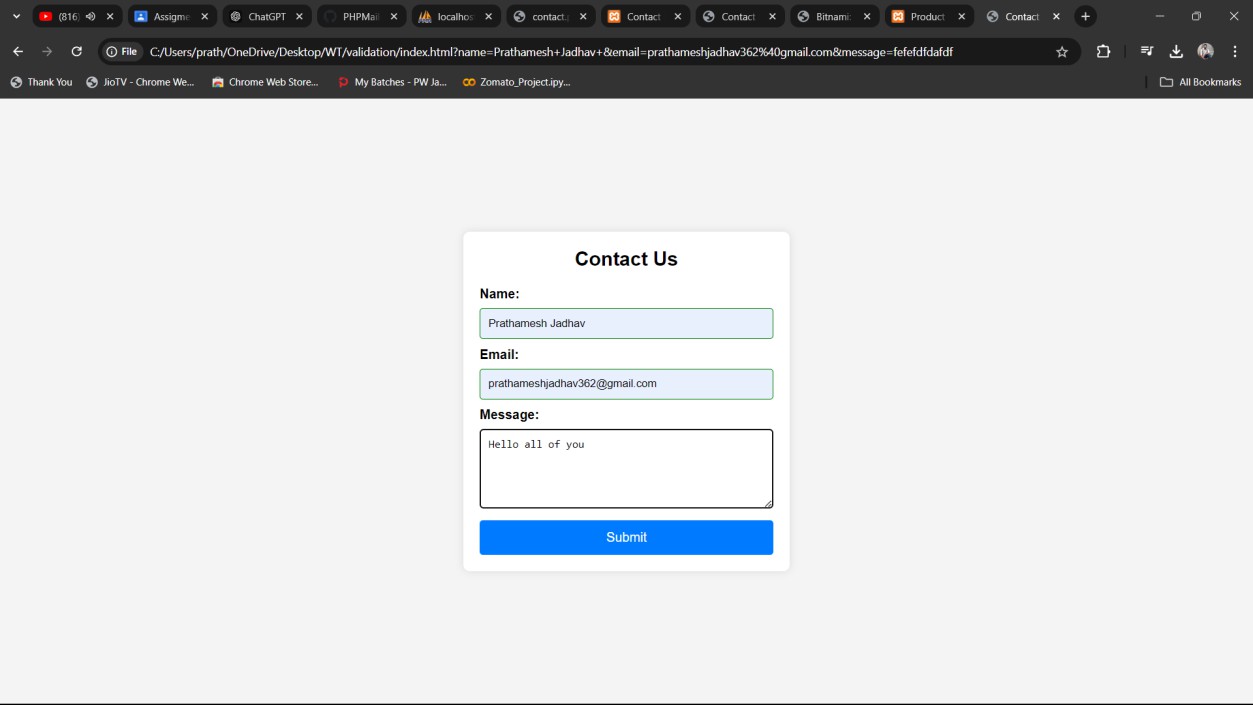
if (messageValue.length < 10) {

showError(messageInput, messageError, 'Message must be at least 10 characters'); e.preventDefault();

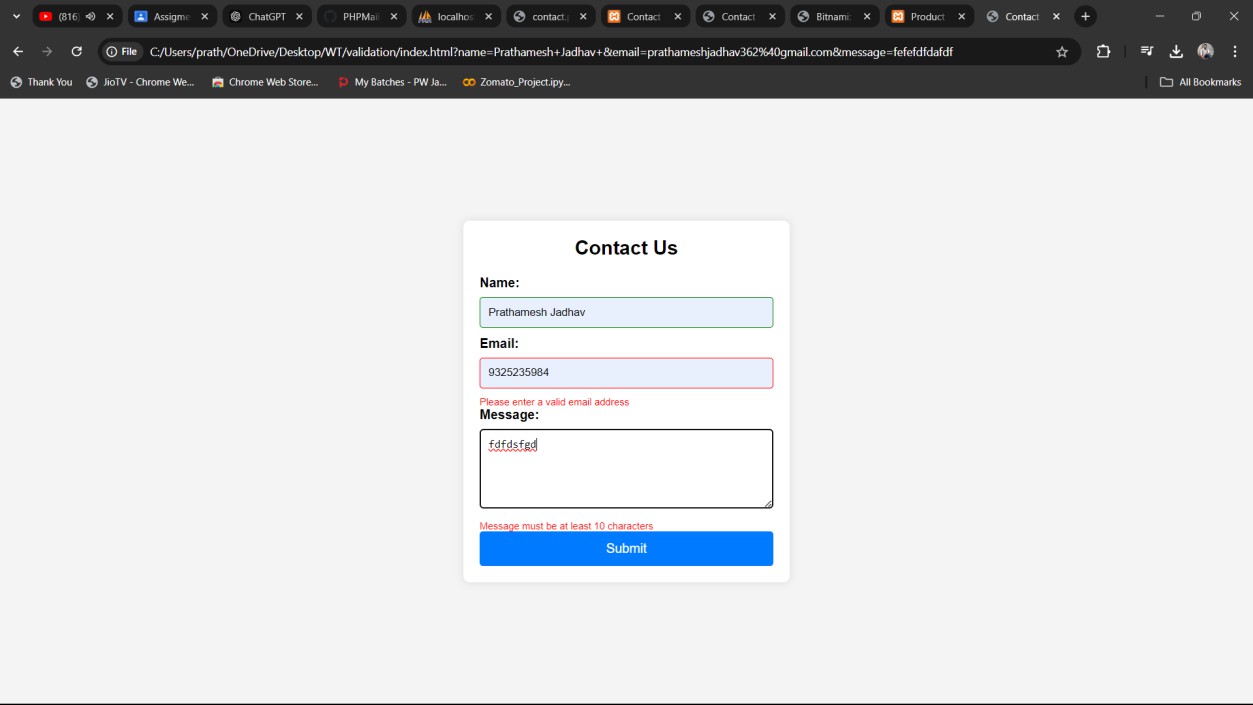
}

});

Correct Format (Validate)



Wrong Format(Invalidate)



# Assignment No.9

## Program-:

<?php session\_start();

// Database connection

$conn = new mysqli("localhost", "root", "", "user\_auth");

// Check for connection errors if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Handle form submission

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$email = $conn->real\_escape\_string($\_POST['email']);

$password = $\_POST['password'];

// Fetch user from the database

$result = $conn->query("SELECT \* FROM users WHERE email='$email'");

if ($result->num\_rows > 0) {

$user = $result->fetch\_assoc();

// Verify the password

if (password\_verify($password, $user['password'])) {

$\_SESSION['username'] = $user['username']; header("Location: dashboard.php");

exit();

} else {

echo "Incorrect password!";

}

} else {

echo "No user found with this email!";

}

}

$conn->close();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login</title>

</head>

<body>

<h2>Login</h2>

<form action="login.php" method="POST">

Email: <input type="email" name="email" required><br><br>

Password: <input type="password" name="password" required><br><br>

<input type="submit" value="Login">

</form>

</body>

</html>

<?php session\_start();

if (!isset($\_SESSION['username'])) {

header("Location: login.php"); exit();

}

echo "Welcome, " . $\_SESSION['username'] . "!"; echo "<br><a href='logout.php'>Logout</a>";

?>

<?php session\_start(); session\_destroy();

header("Location: login.php"); exit();

?>

<?php

// Database connection

$conn = new mysqli("localhost", "root", "", "user\_auth");

// Check for connection errors if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Handle form submission

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$username = $conn->real\_escape\_string($\_POST['username']);

$email = $conn->real\_escape\_string($\_POST['email']);

$password = password\_hash($\_POST['password'], PASSWORD\_DEFAULT);

// Insert user into the database

$sql = "INSERT INTO users (username, email, password) VALUES ('$username', '$email', '$password')";

if ($conn->query($sql) === TRUE) {

echo "Registration successful. <a href='login.php'>Login here</a>";

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

}

$conn->close();

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Register</title>

</head>

<body>

<h2>Register</h2>

<form action="register.php" method="POST">

Username: <input type="text" name="username" required><br><br> Email: <input type="email" name="email" required><br><br>

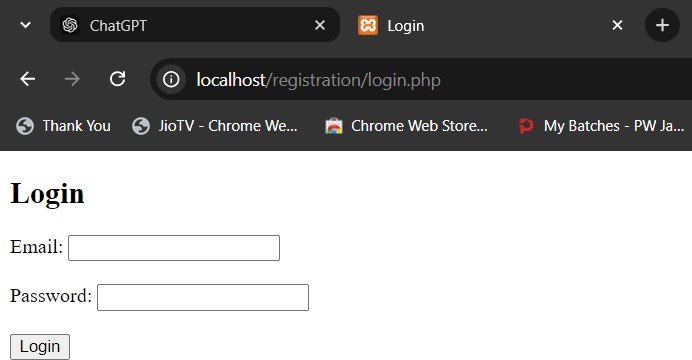
Password: <input type="password" name="password" required><br><br>

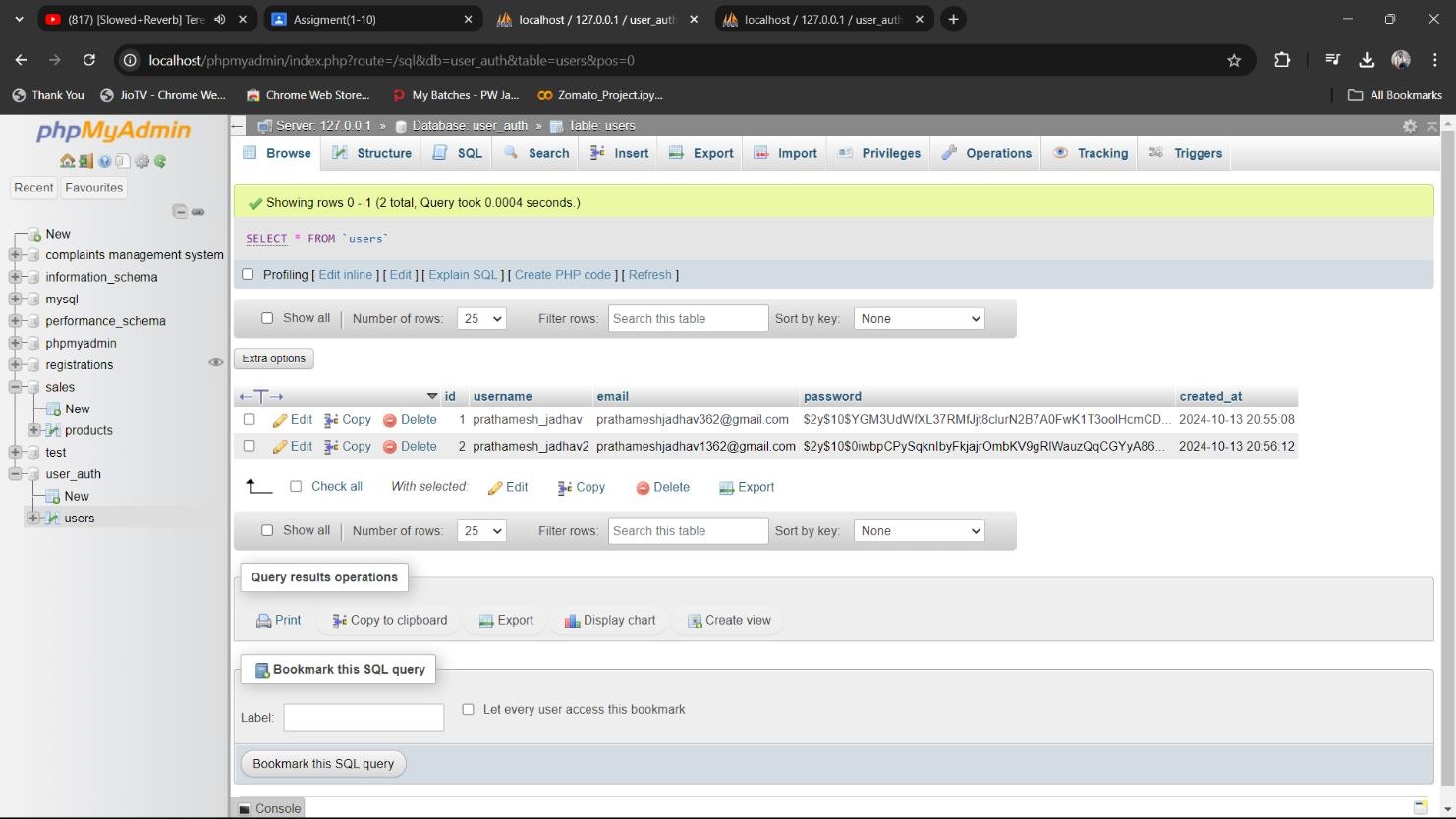
<input type="submit" value="Register">

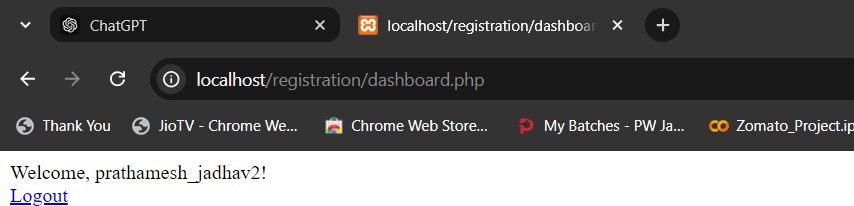
</form>

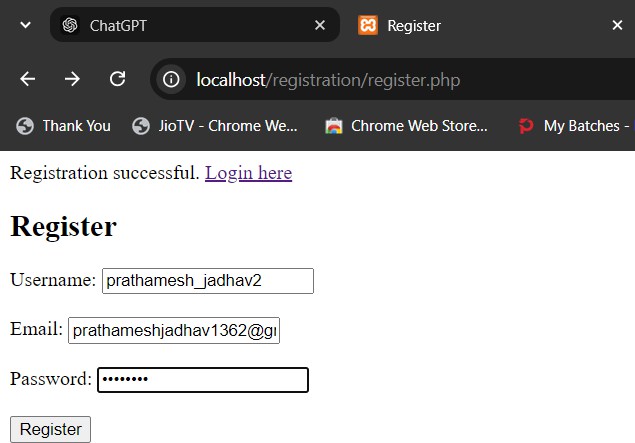
</body>

</html>









# Assignment No.10

## Program-:

from flask import Flask app = Flask( name )

@app.route('/greet/<name>') def greet(name):

# Return a personalized greeting message

return f"Hello, {name}! Welcome to the Flask app." if name == ' main ':

## app.run(debug=True)

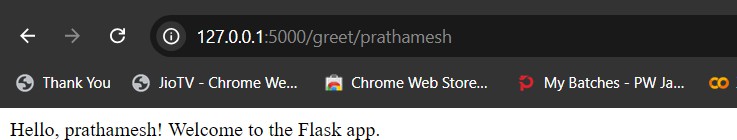
//Output-:

"c:\Users\prath\OneDrive\Documents\My documents\T.Y.B.Tech\_Acadmics\Project\flask\_practi.py"

* Serving Flask app 'flask\_practi'
* Debug mode: on

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

* Running on [http://127.0.0.1:5000](http://127.0.0.1:5000/) Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 110-974-918



# Assignment No.11

## Program-:

class Person

# Attribute accessors for name, age, and email attr\_accessor :name, :age, :email

# Initialize method to set the attributes when creating a new Person object def initialize(name, age, email)

@name = name @age = age

@email = email end

# Method to display person details def display\_info

puts "Name: #{@name}" puts "Age: #{@age}" puts "Email: #{@email}"

end end

# Create a new person object

person = Person.new("John Doe", 30, "[johndoe@example.com](mailto:johndoe@example.com)")

person.name = "Jane Smith" person.age = 25

person.email = ["janesmith@example.com](mailto:janesmith@example.com)" person.display\_info

## //Output-:

Name: Jane Smith Age: 25

Email: janesmith@exam

Program-:

# Assignment No.12

#### Navigate to the Project Directory

cd hello\_rails\_app **Start the Rails Server** rails server

#### Create a New Controller

rails generate controller Welcome index

#### Edit the Controller Action

class WelcomeController < ApplicationController def index

end end

#### Modify the View

<h1>Hello, Rails!</h1>

#### Set the Root Route

Rails.application.routes.draw do get 'welcome/index'

# Set the root path root 'welcome#index' end

#### Restart the Rails Server

rails server

#### View the Application

[http://localhost:3000](http://localhost:3000/)

## //Output-:

Hello, Rails!

# MINI-PROJECT-: TASK MANAGER

1. Aim:

To develop a task manager web application that allows users to create, manage, and categorize tasks, with user authentication and project-specific categorization.

1. Objectives:
   * Implement user authentication (signup, login, logout).
   * Enable users to add, edit, and delete tasks.
   * Allow categorization of tasks into different projects.
   * Provide the functionality to mark tasks as complete or incomplete.
   * Display upcoming tasks on the dashboard.
2. Theory:

### Web Application Overview:

A task manager is a web application designed to help users keep track of their tasks. It offers various features like creating, updating, deleting, and organizing tasks into projects. To enhance the user experience, functionalities such as user authentication, project-based task grouping, and a dashboard displaying task status are essential.

Key Technologies:

* + **HTML**: Used for structuring the web pages.
  + **CSS**: Used for styling the pages.
  + **PHP**: Handles server-side logic, including user authentication and task management.
  + **MySQL**: Stores user and task data.
  + **JavaScript (optional)**: Adds interactivity, such as handling form submissions without page refresh.

### Tools and Technologies:

* + **Languages**: HTML, CSS, PHP, SQL
  + **Database**: MySQL
  + **Framework (optional)**: Laravel or CodeIgniter for PHP (or pure PHP)
  + **Local Server**: XAMPP or WAMP for running PHP and MySQL
  + **Version Control**: Git (optional)

### 3.1 Web Application Overview:

* + A web application is a client-server software application that runs in a web browser. Unlike traditional desktop applications, web apps don't require installation and can be accessed from any device with a browser and internet connection. The task manager app you are developing is a dynamic web application designed to help users manage tasks

effectively. It involves user interaction, persistent data storage, and real- time data manipulation.

* + The core features of this web app include:
  + **User Authentication:** To ensure that only authorized users can access the tasks.
  + **Task Management:** Adding, editing, and deleting tasks for better organization.
  + **Task Categorization:** Tasks can be grouped under specific projects, improving clarity.
  + **Task Status Tracking:** Tasks can be marked as complete or incomplete for better task management.
  + **Dashboard View:** A centralized interface to display upcoming tasks and their status, aiding users in keeping track of deadlines and priorities.

### 3.2 Client-Server Architecture:

* + The Task Manager is built on the client-server model:
  + **Client-side:** The client side (frontend) consists of HTML for structure, CSS for styling, and optional JavaScript for dynamic behavior.
  + **Server-side:** The server side (backend) is powered by PHP, which handles the business logic, processes data from the user, and communicates with the database.
  + **Database:** MySQL is used to store user credentials, tasks, and project data, ensuring persistence across sessions.

### 3.3 HTML (HyperText Markup Language):

* + HTML is the standard language for creating web pages. It defines the structure of a webpage and organizes the content into elements like headings, paragraphs, forms, and buttons. In this project, HTML is used to create forms for user input (e.g., login forms, task creation forms), tables for displaying task lists, and buttons for user interaction (e.g., task completion or deletion).
  + Key elements of HTML in the project:
  + **Forms:** For user input (e.g., login, signup, task creation).
  + **Tables:** For displaying tasks in an organized manner.
  + **Buttons:** For handling user actions like adding, deleting, and marking tasks as complete.

### 3.4 CSS (Cascading Style Sheets):

* + CSS is used to style the HTML elements and improve the visual appeal of the web application. It controls the layout, colors, fonts, and positioning of elements on the web page.
  + In this project, CSS is responsible for:
  + **Layout design:** Ensuring the web pages (login, task dashboard) have a user-friendly and responsive design.
  + **Styling forms and tables:** Making the input fields, buttons, and task lists visually appealing.
  + **Media queries:** (optional) To make the web app responsive across different devices (e.g., mobile, tablet, desktop).

### 3.5 PHP (Hypertext Preprocessor):

* + PHP is a server-side scripting language that is embedded in HTML to manage dynamic content, databases, session management, and more. In this project, PHP is responsible for:
  + **Handling User Authentication:** Checking user credentials during login, registering new users, and logging out.
  + **CRUD Operations (Create, Read, Update, Delete):** PHP is used to create new tasks, read tasks from the database, update task details, and delete tasks.
  + **Session Management:** Maintaining user sessions so that each logged-in user can manage their tasks independently.
  + **Database Interactions:** PHP interacts with MySQL to store and retrieve data for tasks and users.
  + Key PHP functions and concepts used:
  + **Sessions:** $\_SESSION is used to track user logins and store session information like user\_id to differentiate between users.
  + **Forms and POST method:** PHP processes form data sent via POST (e.g., login, task creation) and takes appropriate actions like querying the database or updating records.
  + **SQL Queries:** PHP constructs and executes SQL queries to perform database operations.

### 3.6 MySQL (Relational Database Management System):

* + MySQL is an open-source RDBMS that uses Structured Query Language (SQL) to interact with databases. For this task manager application,

MySQL is used to store user information, tasks, and project data. Each entity (user, task, project) is stored in a table with relationships between them (e.g., each task belongs to a project and each project belongs to a user).

* + Important concepts used in MySQL for the project:
  + **Tables and Relationships:** The application uses three tables (users, projects, tasks) with foreign key relationships to associate tasks with projects and projects with users.
  + **CRUD Operations:** SQL commands like INSERT, SELECT, UPDATE, and DELETE are used to manage tasks and projects.
  + **Data Integrity:** Constraints like NOT NULL, UNIQUE, and FOREIGN KEY ensure that data is entered correctly and relationships between tables are maintained.

### 3.7 User Authentication:

* + Authentication is the process of verifying the identity of a user. In the task manager, authentication ensures that only authorized users can access their tasks and data.
  + The authentication process includes:
  + **Signup:** A user provides a username and password during registration.

The password is hashed using PHP’s password\_hash() function for security, and the username is stored in the users table.

* + **Login:** When a user logs in, their username and password are checked against the hashed values in the database. If verified, a session is created, and the user is redirected to their task dashboard.
  + **Logout:** When a user logs out, their session is destroyed, preventing unauthorized access to their tasks.

### 3.8 Task Management:

* + Task management is the core feature of the application. Each task is associated with a project and a user. Tasks can have attributes like title, description, due date, and status (complete or incomplete).
  + The task manager allows users to:
  + **Add Tasks:** Create new tasks with details like title, description, and due date.
  + **Edit Tasks:** Modify task details if required.
  + **Delete Tasks:** Remove tasks that are no longer needed.
  + **Mark Tasks as Complete/Incomplete:** Users can toggle the status of tasks to manage progress.
  + Each task is displayed on the user's dashboard, with its status and due date prominently shown, helping users prioritize their tasks.

### 3.9 Dashboard and User Interface:

* + The dashboard provides a user-friendly interface for managing tasks. It lists all tasks under their respective projects and provides options to:
  + Mark tasks as complete or incomplete.
  + Edit task details.
  + Delete tasks.
  + A well-designed dashboard improves the user experience by offering

clear visibility of upcoming tasks and allowing easy interaction with task controls.

Conclusion:

This mini-project provided hands-on experience in building a simple task manager web application. By using HTML, CSS, PHP, and MySQL, we successfully implemented key features like user authentication, task management, and project-based categorization. It demonstrates the power of web technologies to create a functional and user-friendly task management system.

## Program-:

<?php

require 'db.php'; // Include database connection

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$username = $\_POST['username'];

$password = password\_hash($\_POST['password'], PASSWORD\_DEFAULT); // Hashing the password

// Inserting user into the database

$query = "INSERT INTO users (username, password) VALUES ('$username', '$password')";

if ($conn->query($query)) {

echo "User registered successfully! <a href='login.php'>Login</a>";

} else {

echo "Error: " . $conn->error;

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Signup</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h2>Signup</h2>

<form method="POST" action="">

<label for="username">Username:</label>

<input type="text" name="username" id="username" required><br><br>

<label for="password">Password:</label>

<input type="password" name="password" id="password" required><br><br>

<button type="submit">Signup</button>

</form>

</body>

</html>

<?php session\_start(); session\_destroy();

header("Location: login.php"); // Redirect to login page after logout

?>

<?php session\_start(); require 'db.php';

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$username = $\_POST['username'];

$password = $\_POST['password'];

// Fetch user from the database

$query = "SELECT \* FROM users WHERE username='$username'";

$result = $conn->query($query);

if ($result && $result->num\_rows > 0) {

$user = $result->fetch\_assoc();

// Verify password

if (password\_verify($password, $user['password'])) {

$\_SESSION['user\_id'] = $user['id'];

header("Location: dashboard.php"); // Redirect to dashboard after login exit();

} else {

echo "Invalid password.";

}

} else {

echo "No user found.";

}

}

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Login</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h2>Login</h2>

<form method="POST" action="">

<label for="username">Username:</label>

<input type="text" name="username" id="username" required><br><br>

<label for="password">Password:</label>

<input type="password" name="password" id="password" required><br><br>

<button type="submit">Login</button>

</form>

</body>

</html>

<?php

$servername = "localhost";

$username = "root"; // Default XAMPP user is root

$password = ""; // Default XAMPP password is empty

$dbname = "task\_manager";

// Create connection

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

// Check connection

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

?>

<?php session\_start(); require 'db.php';

// Redirect to login if the user is not logged in if (!isset($\_SESSION['user\_id'])) {

header("Location: login.php"); exit();

}

$user\_id = $\_SESSION['user\_id'];

// Fetch projects and tasks for the logged-in user

$query = "SELECT \* FROM projects WHERE user\_id='$user\_id'";

$projects = $conn->query($query);

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Dashboard</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h2>Welcome to your Task Manager Dashboard</h2>

<a href="logout.php">Logout</a>

<h3>Your Projects</h3>

<?php while ($project = $projects->fetch\_assoc()): ?>

<h4><?php echo $project['name']; ?></h4>

<?php

$project\_id = $project['id'];

$task\_query = "SELECT \* FROM tasks WHERE project\_id='$project\_id'";

$tasks = $conn->query($task\_query);

?>

<ul>

<?php while ($task = $tasks->fetch\_assoc()): ?>

<li><?php echo $task['title']; ?> - <?php echo $task['status']; ?></li>

<?php endwhile; ?>

</ul>

<?php endwhile; ?>

<a href="add\_task.php">Add New Task</a>

</body>

</html>

<?php session\_start(); require 'db.php';

// Redirect to login if the user is not logged in

if (!isset($\_SESSION['user\_id'])) { header("Location: login.php"); exit();

}

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$project\_id = $\_POST['project\_id'];

$title = $\_POST['title'];

$description = $\_POST['description'];

$due\_date = $\_POST['due\_date'];

// Inserting new task into the database

$query = "INSERT INTO tasks (project\_id, title, description, due\_date) VALUES ('$project\_id', '$title', '$description', '$due\_date')";

if ($conn->query($query)) {

echo "Task added successfully! <a href='dashboard.php'>Back to Dashboard</a>";

} else {

echo "Error: " . $conn->error;

}

}

// Fetch user projects

$user\_id = $\_SESSION['user\_id'];

$query = "SELECT \* FROM projects WHERE user\_id='$user\_id'";

$projects = $conn->query($query);

?>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Add Task</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<h2>Add New Task</h2>

<form method="POST" action="">

<label for="project">Project:</label>

<select name="project\_id" id="project" required>

<?php while ($project = $projects->fetch\_assoc()): ?>

<option value="<?php echo $project['id']; ?>"><?php echo $project['name']; ?></option>

<?php endwhile; ?>

</select><br><br>

<label for="title">Task Title:</label>

<input type="text" name="title" id="title" required><br><br>

<label for="description">Description:</label>

<textarea name="description" id="description" required></textarea><br><br>

<label for="due\_date">Due Date:</label>

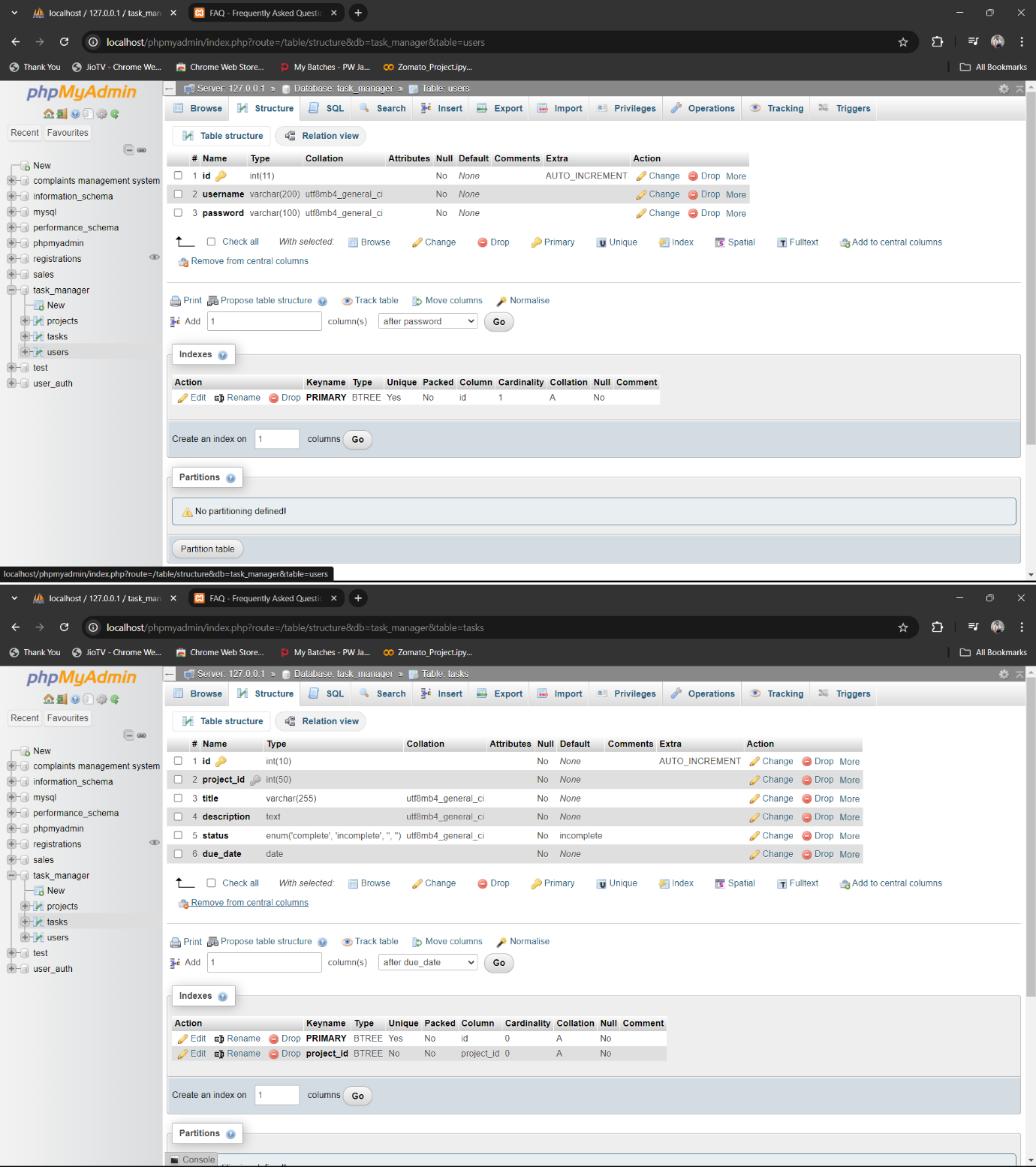
<input type="date" name="due\_date" id="due\_date" required><br><br>

<button type="submit">Add Task</button>

</form>

</body>

</html>

Outputs-:

