MINI – PROJECT
Web page creation

User and Task List /Order List/ * List

Project Objective

The objective of this project is to create a dynamic and responsive web page that enables users to manage their tasks and view personal information. The page will utilize HTML for structure, CSS (with Bootstrap) for styling, and JavaScript for interactivity. Key features include displaying user information, managing a task list with priority levels, and providing a real-time date and time display.

Key Features

1. **Responsive Layout**
    - **Sidebar:** Contains navigation links or user options.
    - **Navigation Bar:** Provides access to different sections of the page.
    - **Content Area:** Displays user information, task list, and date/time.
2. **User Information Display**
    - **User Details:** Shows user's name and city.
    - **Form Inputs:** Allows users to enter or update their name and city, with real-time validation.
3. **Task Management**
    - **Task List:** Displays a list of tasks in a table, grid, or flex layout.
    - **Priority Levels:** Tasks can be categorized with priority levels (P1, P2, P3, P4) where P1 is high priority and P2/P3 are medium priorities, and P4 is low priority.
    - **Task Addition:** A form to add new tasks with validation and a Bootstrap modal for confirmation.
    - **Priority Indicators:** Tasks are visually distinguished based on priority, and high-priority tasks have interactive effects like mouse-over color changes.
4. **Date and Time Display**
    - **Current Date/Time:** Shows the current date and time prominently on the page.

- **Automatic Updates:** The time is updated every second using `setInterval`.

5. **Form Validation**
    - **Required Fields:** Ensures all necessary fields are filled out before submission.
    - **User Feedback:** Displays alerts or custom Bootstrap modals to confirm or notify users of form submission status.
6. **Interactive Features**
    - **Task Priority Switch:** Allows users to filter tasks based on priority.
    - **Mouse-over Effects:** Provides visual feedback on high-priority tasks.

Technologies Used

- **HTML:** For structuring the content of the web page.
- **CSS:** Styled using Bootstrap for a responsive and modern design.
- **JavaScript:** For interactive elements such as dynamic date/time updates, form validation, and priority switching.

Expected User Interaction

Landing on the Page: Users will see a clean, responsive layout with sections for their information and task management.
Entering User Information: Users can input their name and city through a form. Validation ensures accuracy before submission.
Managing Tasks: Users can add tasks, set their priorities, and view the updated list. Tasks are categorized and displayed with color-coded priorities.
Viewing Date and Time: Users see the current date and time, which updates automatically.

Code:
Index.html file:

```
<!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="Styles.css">
```

```
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/boxicons@latest/css/boxicons.min.css">
</head>
<body id="body-pd">
    <header class="header" id="header">
        <div class="header_toggle"> <i class='bx bx-menu' id="header-toggle"></i> </div>
        <div class="header_img"> <img src="https://i.imgur.com/hczKIze.jpg" alt=""> </div>
    </header>
    <div class="l-navbar" id="nav-bar">
        <nav class="nav">
            <div>
                <a href="#" class="nav_logo"> <i class='bx bx-layer nav_logo-icon'></i> <span class="nav_logo-name">Mini - Project - 2</span> </a>
                <div class="nav_list">
                    <a href="#" class="nav_link active">
                        <i class='bx bx-grid-alt nav_icon'></i>
                        <span class="nav_name">Dashboard</span>
                    </a>
                    <a href="#" class="nav_link">
                        <i class='bx bx-user nav_icon'></i>
                        <span class="nav_name">Users Entry</span>
                    </a>
                    <a href="#task" class="nav_link">
                        <i class='bx bx-task nav_icon'></i>
                        <span class="nav_name">Task</span>
                    </a>
                    <a href="#date-time" class="nav_link">
                        <i class='bx bx-timer nav_icon'></i>
                        <span class="nav_name">Date & Time</span>
                    </a>
                </div>
            </div>
            <a href="#" class="nav_link">
                <i class='bx bx-log-out nav_icon'></i>
                <span class="nav_name">SignOut</span>
            </a>
        </nav>
    </div>
    <!--Container Main start-->
    <div class="container-fluid">
```

```
        <button class="btn btn-primary d-md-none" type="button"
id="sidebarToggle">Toggle Sidebar</button>
        <div class="d-flex justify-content-between align-items-center my-4 py-5">
            <h2>User Information</h2>
        </div>
        <form id="userForm">
            <div class="mb-2">
                <label for="name" class="form-label">Name:</label>
                <input type="text" class="form-control" id="name" required>
                <div class="invalid-feedback">Please enter your name.</div>
            </div>
            <div class="mb-2">
                <label for="city" class="form-label">City:</label>
                <input type="text" class="form-control" id="city" required>
                <div class="invalid-feedback">Please enter your city.</div>
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>

        <h2 id="task" class="mt-4">Task List</h2>
        <hr>
        <form id="taskForm">
            <div class="mb-2">
                <label for="taskName" class="form-label">Task Name:</label>
                <input type="text" class="form-control" id="taskName" required>
                <div class="invalid-feedback">Please enter your task...</div>
            </div>
            <div class="mb-2">
                <label for="userName" class="form-label">User:</label>
                <input type="text" list="userNames" class="form-control"
id="userName" required>
                <datalist id="userNames">
                    <option value="None">None</option>
                    <!-- Users will be added here dynamically -->
                  </datalist>
                <div class="invalid-feedback">Please enter your user...</div>
            </div>
            <div class="mb-4">
                <label for="priority" class="form-label">Priority:</label>
                <select class="form-select" id="priority" required>
                    <option value="">Select priority</option>
                    <option value="P1">P1 (High-priority)</option>
                    <option value="P2">P2 (Medium-priority)</option>
                    <option value="P3">P3 (Medium-priority)</option>
                    <option value="P4">P4 (Low-priority)</option>
```

```
                </select>
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
        </form>
        <br>
        <input type="text" id="filterInput" class="filter-input"
placeholder="Filter tasks...">
        <button class="btn btn-secondary" id="sortTask">Sort by Task</button>
        <button class="btn btn-secondary" id="sortPriority">Sort by
Priority</button>
        <table class="table task-table">
            <thead>
                <tr>
                    <th>Task</th>
                    <th>Priority</th>
                </tr>
            </thead>
            <tbody id="taskList">
                <!-- Tasks will be added here dynamically -->
            </tbody>
        </table>

        <hr>
        <h3 id="date-time">Date & time</h3>
        <div class="date-time-display" id="dateTimeDisplay">
            <!-- Date and time will be displayed here -->
        </div>
    </div>
    <!--Container Main end-->
    <script src="JavaScript.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/js/bootstrap.bundle.min.js"></script>
</html>
```

Styles.css file:

```
@import
url("https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&display=swa
p");
:root {
  --header-height: 3rem;
```

```
  --nav-width: 68px;
  --first-color: #4723d9;
  --first-color-light: #afa5d9;
  --white-color: #f7f6fb;
  --body-font: "Nunito", sans-serif;
  --normal-font-size: 1rem;
  --z-fixed: 100;
}
*,
::before,
::after {
  box-sizing: border-box;
}
body {
  position: relative;
  margin: var(--header-height) 0 0 0;
  padding: 0 1rem;
  font-family: var(--body-font);
  font-size: var(--normal-font-size);
  transition: 0.5s;
}
a {
  text-decoration: none;
}
.header {
  width: 100%;
  height: var(--header-height);
  position: fixed;
  top: 0;
  left: 0;
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 0 1rem;
  background-color: var(--white-color);
  z-index: var(--z-fixed);
  transition: 0.5s;
}
.header_toggle {
  color: var(--first-color);
  font-size: 1.5rem;
  cursor: pointer;
}
.header_img {
  width: 35px;
```

```
  height: 35px;
  display: flex;
  justify-content: center;
  border-radius: 50%;
  overflow: hidden;
}
.header_img img {
  width: 40px;
}
.l-navbar {
  position: fixed;
  top: 0;
  left: -30%;
  width: var(--nav-width);
  height: 100vh;
  background-color: var(--first-color);
  padding: 0.5rem 1rem 0 0;
  transition: 0.5s;
  z-index: var(--z-fixed);
}
.nav {
  height: 100%;
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  overflow: hidden;
}
.nav_logo,
.nav_link {
  display: grid;
  grid-template-columns: max-content max-content;
  align-items: center;
  column-gap: 1rem;
  padding: 0.5rem 0 0.5rem 1.5rem;
}
.nav_logo {
  margin-bottom: 2rem;
}
.nav_logo-icon {
  font-size: 1.25rem;
  color: var(--white-color);
}
.nav_logo-name {
  color: var(--white-color);
  font-weight: 700;
```

```
}
.nav_link {
  position: relative;
  color: var(--first-color-light);
  margin-bottom: 1.5rem;
  transition: 0.3s;
}
.nav_link:hover {
  color: var(--white-color);
}
.nav_icon {
  font-size: 1.25rem;
}
.show {
  left: 0;
}
.body-pd {
  padding-left: calc(var(--nav-width) + 1rem);
}
.active {
  color: var(--white-color);
}
.active::before {
  content: "";
  position: absolute;
  left: 0;
  width: 2px;
  height: 32px;
  background-color: var(--white-color);
}
.height-100 {
  height: 100vh;
}
.content {
  padding: 20px;
}
.priority-high {
  background-color: #ffcccc;
}
.priority-medium {
  background-color: #ffffcc;
}
.priority-low {
  background-color: #ccffcc;
}
```

```css
.task-table tbody tr:hover {
  background-color: #f1f1f1;
}
.date-time-display {
  font-size: 1.2rem;
  margin-top: 20px;
}
.dark-mode {
  background-color: #343a40;
  color: #f8f9fa;
}
.dark-mode .sidebar {
  background-color: #495057;
}
.dark-mode .sidebar a {
  color: #020202;
}
.dark-mode .sidebar a:hover {
  background: #6c757d;
}

@media screen and (min-width: 768px) {
  body {
    margin: calc(var(--header-height) + 1rem) 0 0 0;
    padding-left: calc(var(--nav-width) + 2rem);
  }
  .header {
    height: calc(var(--header-height) + 1rem);
    padding: 0 2rem 0 calc(var(--nav-width) + 2rem);
  }
  .header_img {
    width: 40px;
    height: 40px;
  }
  .header_img img {
    width: 45px;
  }
  .l-navbar {
    left: 0;
    padding: 1rem 1rem 0 0;
  }
  .show {
    width: calc(var(--nav-width) + 156px);
  }
  .body-pd {
```

```
    padding-left: calc(var(--nav-width) + 188px);
  }
}
```

JavaScript.js file:

```
// Form validation and task handling
document.addEventListener('DOMContentLoaded', function() {



    // Function to display the current date and time
function updateDateTime() {
    const now = new Date();
    const options = { weekday: 'long', year: 'numeric', month: 'long', day:
'numeric', hour: '2-digit', minute: '2-digit', second: '2-digit' };
    document.getElementById('dateTimeDisplay').textContent =
now.toLocaleDateString('en-US', options);
}
setInterval(updateDateTime, 1000);
updateDateTime();



const showNavbar = (toggleId, navId, bodyId, headerId) =>{
    const toggle = document.getElementById(toggleId),
    nav = document.getElementById(navId),
    bodypd = document.getElementById(bodyId),
    headerpd = document.getElementById(headerId)

        // Validate that all variables exist
    if(toggle && nav && bodypd && headerpd){
        toggle.addEventListener('click', ()=>{
        nav.classList.toggle('show')
        toggle.classList.toggle('bx-x')
        bodypd.classList.toggle('body-pd')
        headerpd.classList.toggle('body-pd')
        })
    }
}

    showNavbar('header-toggle','nav-bar','body-pd','header')
```

```
    /*===== LINK ACTIVE =====*/
    const linkColor = document.querySelectorAll('.nav_link')

    function colorLink(){
    if(linkColor){
        linkColor.forEach(l=> l.classList.remove('active'))
        this.classList.add('active')
    }}

    linkColor.forEach(l=> l.addEventListener('click', colorLink))

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

    const sortTaskButton = document.getElementById('sortTask');
    const sortPriorityButton = document.getElementById('sortPriority');
    const filterInput = document.getElementById('filterInput');

    const userForm = document.getElementById('userForm');
    const nameInput = document.getElementById('name');
    const cityInput = document.getElementById('city');



    const taskForm = document.getElementById('taskForm');
    const taskNameInput = document.getElementById('taskName');
    const userNameInput = document.getElementById('userName');
    const prioritySelect = document.getElementById('priority');

    const userNamesDatalist = document.getElementById('userNames');




    // Initialize tasks array
    const tasks = [];
    const users = [];

// Handle userFrom submit event
    userForm.addEventListener('submit', function(event) {
        event.preventDefault();

        // Get user input values
        const name = nameInput.value.trim();
        const city = cityInput.value.trim();
```

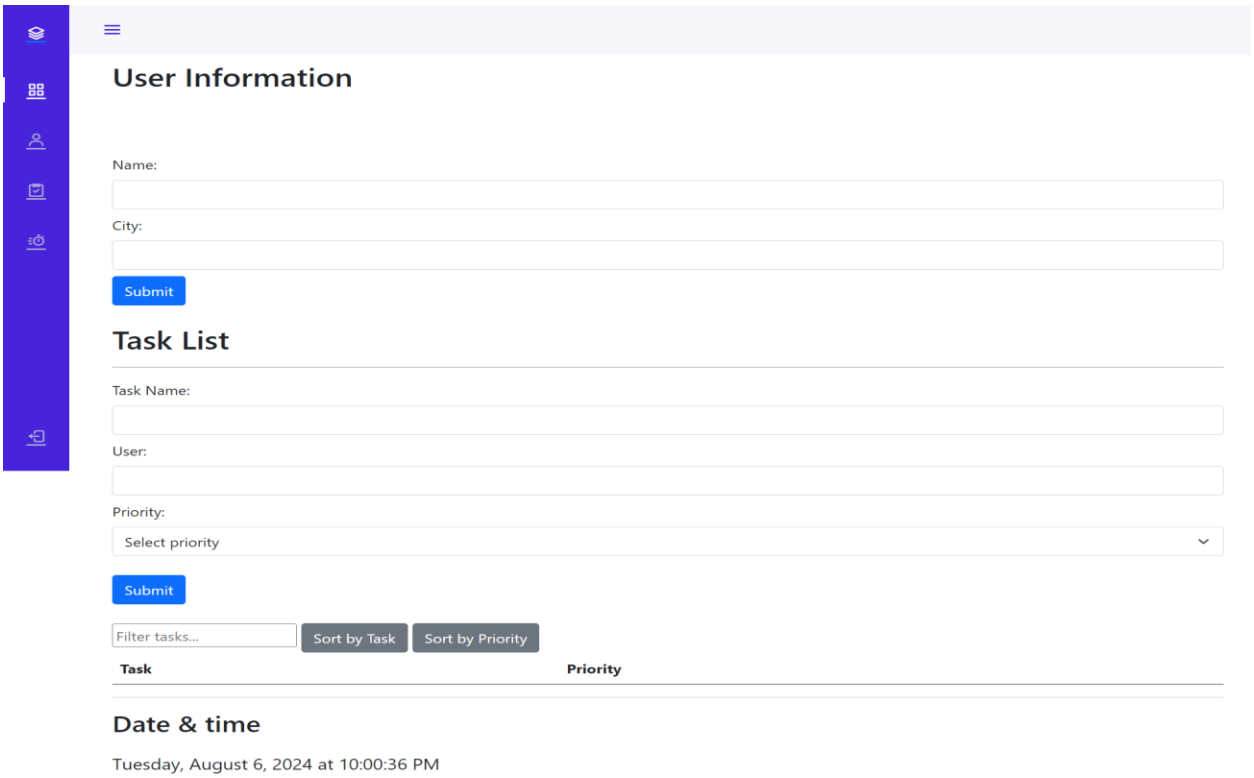
```
        // Validate input fields
        if (name && city) {
            users.push({ name, city });
            nameInput.value = '';
            cityInput.value = '';
            console.log(`Added new user: ${name}, ${city}`);

            confirm(`Added new user: ${name}`) === true ? renderUsers(users) :
console.log("Not added");
        } else {
            // Handle invalid inputs
            console.error('Please enter valid name and city.');
        }
    });

// Handle taskFrom submit event
    taskForm.addEventListener('submit', function(event) {
        event.preventDefault();

        const taskName = taskNameInput.value.trim();
        const userName = userNameInput.value.trim();
        const priority = prioritySelect.value;

        console.log(`Task: ${taskName}, User: ${userName}, Priority:
${priority}`);

        if (taskName && userName && priority) {
            tasks.push({ name: taskName, user: userName, priority: priority });
            taskNameInput.value = '';
            userNameInput.value = '';
            prioritySelect.value = '';

            confirm(`Added new task ${taskName}`) === true ? renderTasks(tasks) :
console.log('Not added');
        } else {
            // Handle invalid inputs
            console.error('Please enter valid task details.');
        }
    });
```

```
    function renderTasks(tasksToRender) {
        taskList.innerHTML = ''; // Clear existing tasks

        tasksToRender.forEach(task => {
            // Determine class based on task priority
            const priorityClass = task.priority === 'P1' ? 'priority-high' :
                                  (task.priority === 'P2' || task.priority ===
'P3') ? 'priority-medium' :
                                  'priority-low';

            const row = document.createElement('tr');
            row.classList.add(priorityClass);
            row.innerHTML = `
                <td>${task.name} (${task.user})</td>
                <td>${task.priority}</td>
            `;
            taskList.appendChild(row);
        });
    }



    function renderUsers(userToRender) {
        userNamesDatalist.innerHTML = '';
        userToRender.forEach(user => {
            const opt = document.createElement('option');
            opt.value = `${user.name} from ${user.city}`;
            opt.innerHTML = `${user.name} from ${user.city}`;
            userNamesDatalist.appendChild(opt);
            console.log(opt);
        });
    }




    function sortTasksByPriority() {
        const priorityOrder = { 'P1': 1, 'P2': 2, 'P3': 3, 'P4': 4 };
        tasks.sort((a, b) => priorityOrder[a.priority] -
priorityOrder[b.priority]);
        renderTasks(tasks);
    }

    function sortTasksByTask() {
        tasks.sort((a, b) => a.name.localeCompare(b.name));
```

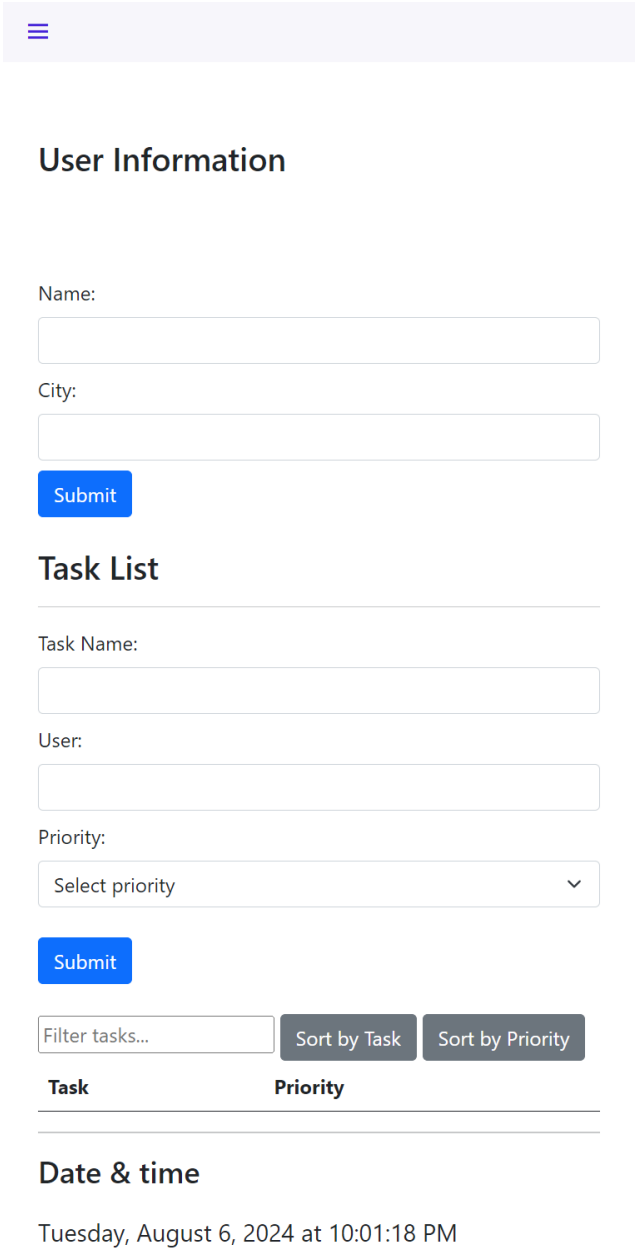
```
        renderTasks(tasks);
    }


    function filterTasks() {
        const filterText = filterInput.value.toLowerCase();
        const filteredTasks = tasks.filter(task =>
            task.name.toLowerCase().includes(filterText) ||
            task.user.toLowerCase().includes(filterText) ||
            task.priority.toLowerCase().includes(filterText)
        );
        renderTasks(filteredTasks);
    }

    sortTaskButton.addEventListener('click', sortTasksByTask);
    sortPriorityButton.addEventListener('click', sortTasksByPriority);
    filterInput.addEventListener('input', filterTasks);

    // Initial rendering
    renderTasks(tasks);
    renderUsers(users);
});
```

Output:

Desktop view:


User Information
Name:
City:
Submit
Task List
Task Name:
User:
Priority:
Select priority
Submit
Filter tasks...
Sort by Task
Sort by Priority
Task
Priority
Date & time
Tuesday, August 6, 2024 at 10:00:36 PM

Mobile view:



Use Case Specifications:

Use Case 1: Viewing User Information

Actor: User

Precondition: User has accessed the web page.

Main Flow:

The user lands on the web page and sees a sidebar, navigation bar, and content area.
The content area displays user information, which includes their name and city.
The user information is dynamically updated based on input from the form.

Postcondition: The user information is correctly displayed in the content area.

Use Case 2: Adding a Task

Actor: User

Precondition: User is on the web page with the task list section visible.

Main Flow:

The user fills out the form to enter details of a new task including name, description, and priority.
The user selects the priority from a set of options (P1, P2, P3, P4).
Upon form submission, a validation check ensures all required fields are filled.
If validation passes, a Bootstrap modal confirms task addition and displays a success message.
The new task is added to the task list, displayed in a table/grid/flex layout with the specified priority.

Postcondition: The new task is visible in the task list with the selected priority.

Use Case 3: Viewing Task List

Actor: User

Precondition: User has accessed the web page.

Main Flow:

The user navigates to the task list section of the page.
The task list is displayed in a table/grid/flex layout.
Tasks are sorted or filtered based on their priority levels.

Postcondition: The user can view and interact with the task list.

Use Case 4: Updating Task Priority

Actor: User

Precondition: User is viewing the task list.

Main Flow:

The user selects a task from the list.
The user changes the priority of the selected task using a dropdown or priority buttons (P1, P2, P3, P4).
The task priority is updated, and the task is repositioned or re-colored based on its new priority.
A message or alert confirms that the priority change was successful.

Postcondition: The task's priority is updated and reflected in the task list.

Use Case 5: Displaying Current Date and Time

Actor: User

Precondition: User has accessed the web page.

Main Flow:

The current date and time are displayed prominently on the page.
The time updates automatically every second using `setInterval`.

Postcondition: The user sees an accurate and continuously updated date and time.

Use Case 6: Handling Required Field Validation

Actor: User

Precondition: User is filling out a form on the web page.

Main Flow:

The user submits the form with one or more required fields left blank.
Validation messages are shown indicating which fields are missing.
The form is not submitted until all required fields are filled out.

Postcondition: The user is prompted to correct the form before submission.

Use Case 7: Confirming Task Addition with Modal

Actor: User

Precondition: User has filled out the form to add a new task.

Main Flow:

After filling out the form and clicking "Submit," a Bootstrap modal appears.
The modal displays a confirmation message about adding the task.
The user confirms the addition, and the task is added to the task list.

Postcondition: The task is successfully added to the task list and confirmed by the user.

Use Case 8: Applying Mouse-over Effects for High Priority Tasks

Actor: User

Precondition: The user is viewing the task list.

Main Flow:

The user hovers the mouse over a task with high priority (P1).

The task item's background color or style changes to indicate high priority.
The effect provides a visual cue of the task's importance.

Postcondition: High priority tasks are visually distinct when hovered over.

Use Case 9: Switching between Task Priorities

Actor: User

Precondition: User is on the web page with tasks listed.

Main Flow:

The user interacts with a priority switch control to filter tasks by priority.
The task list updates to show only tasks of the selected priority level.

Postcondition: The task list displays tasks filtered by the selected priority.