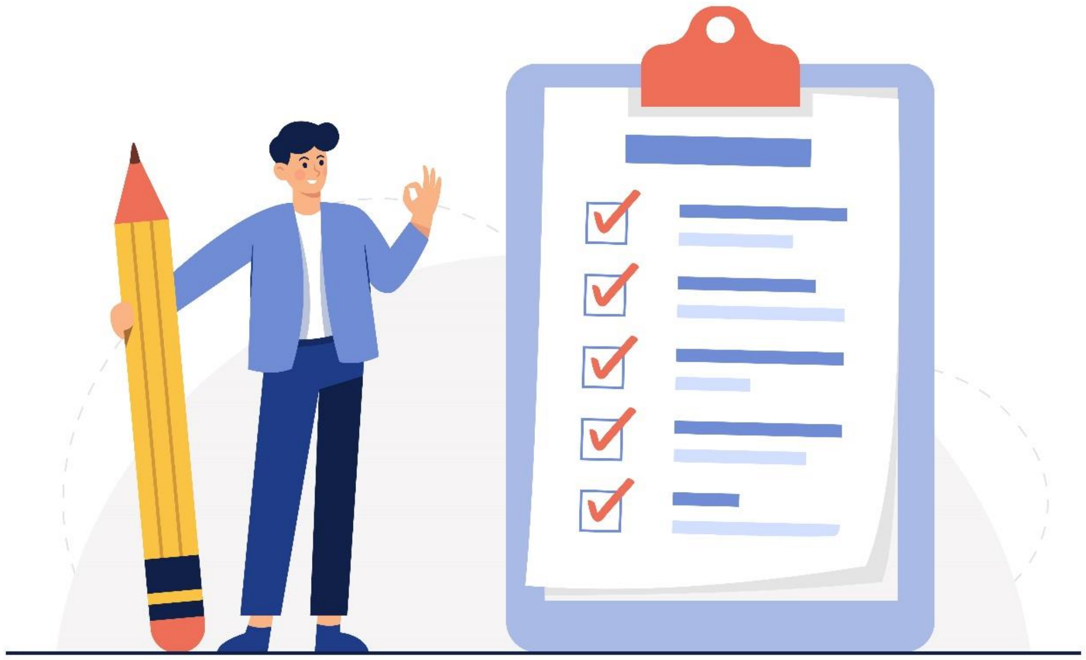
Smart ToDo

System Information



Developer: Nishani Dissanayake

Date: 17. 03. 2025

Table of Contents

1. Introduction

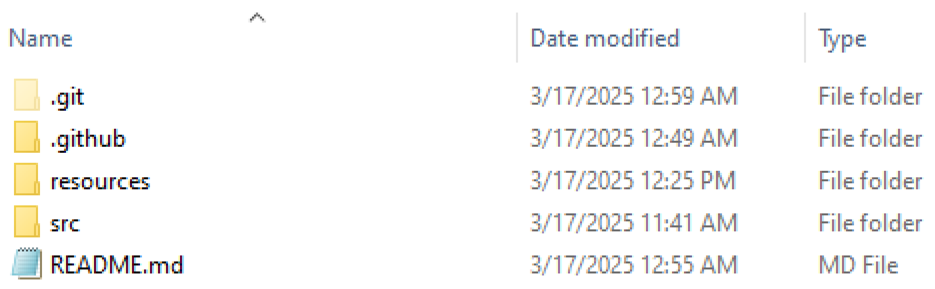
This document comprises a comprehensive overview of the “Smart ToDo” Blazor web application developed for the Blazor developer coding challenge. It covers the project structure, authentication, database design, UI/UX choices, features implemented, tools & technologies, and several test cases.

Application goal: Allow users to login to the system and create, update, and manage day-to-day tasks efficiently.

2. Project Structure

The project is mainly separated into two components,

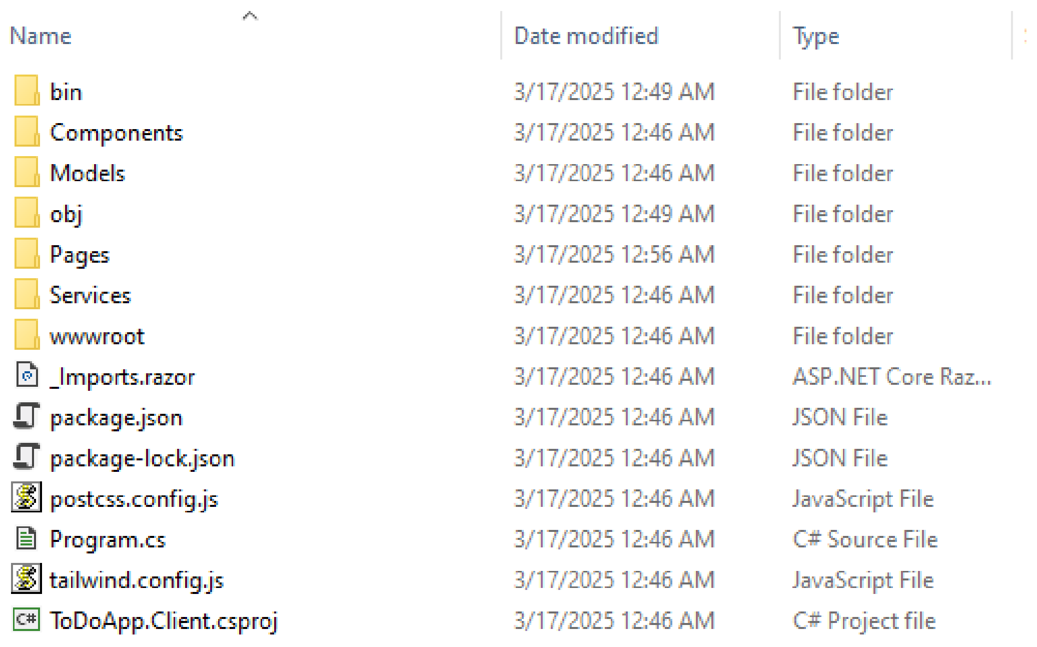
- **src:** code
- **resources:** documentation and diagrams



Name	Date modified	Type
.git	3/17/2025 12:59 AM	File folder
.github	3/17/2025 12:49 AM	File folder
resources	3/17/2025 12:25 PM	File folder
src	3/17/2025 11:41 AM	File folder
README.md	3/17/2025 12:55 AM	MD File

When considering the **src** folder, the project folders are further broken down into several modules.

- **Pages**: Contains all the Razor pages for different functionalities (Login, To-Do List)
- **Components**: Contains reusable UI elements such as modals for adding and editing tasks.
- **Services**: Implements authentication, local storage operations, and task management logic.
- **Models**: Defines the data structures for tasks and users.



Name	Date modified	Type
bin	3/17/2025 12:49 AM	File folder
Components	3/17/2025 12:46 AM	File folder
Models	3/17/2025 12:46 AM	File folder
obj	3/17/2025 12:49 AM	File folder
Pages	3/17/2025 12:56 AM	File folder
Services	3/17/2025 12:46 AM	File folder
wwwroot	3/17/2025 12:46 AM	File folder
_Imports.razor	3/17/2025 12:46 AM	ASP.NET Core Raz...
package.json	3/17/2025 12:46 AM	JSON File
package-lock.json	3/17/2025 12:46 AM	JSON File
postcss.config.js	3/17/2025 12:46 AM	JavaScript File
Program.cs	3/17/2025 12:46 AM	C# Source File
tailwind.config.js	3/17/2025 12:46 AM	JavaScript File
ToDoApp.Client.csproj	3/17/2025 12:46 AM	C# Project file

This modular structure is followed to ensure that there is a clear separation of concerns. Also, the ease of reusability and maintainability of the components are also taken into consideration.

3. Tools and Technologies

Technologies	
Main technologies	Blazor with .NET 9
Styling	Tailwind CSS
State management	Local Storage (Blazored.LocalStorage)
Logging	ILogger (.NET logging)
Authentication	LocalStorage-based authentication
Git flow	Version control strategy

Tools	
IDE	Visual Studio
Version control	GitHub
ER diagram design	LucidChart

4. Features Implemented

4.1 Core Features

Primary requirement fulfillment.

1. Task Management

- Add new tasks with a title, description, and due date
- Mark tasks as complete/incomplete with a checkbox
- Edit existing tasks (update title, description, due date)
- Delete tasks from the list
- Display the count of incomplete tasks

2. Component Architecture & State Management

- Component-based architecture
- Implemented Blazor's two-way binding `(@bind)` for real-time updates
- Used Blazor `EditForm` with `DataAnnotationsValidator` for input validation
- Ensured tasks are correctly managed and state is preserved

3. Input Validation

- Title is a required field before adding a task
- Proper validation messages for missing fields
- Users cannot submit an empty task

4. User Interface & UX

- Simple and user-friendly interface
- Organized layout for task lists and task inputs
- Buttons and actions are intuitive and easy to use

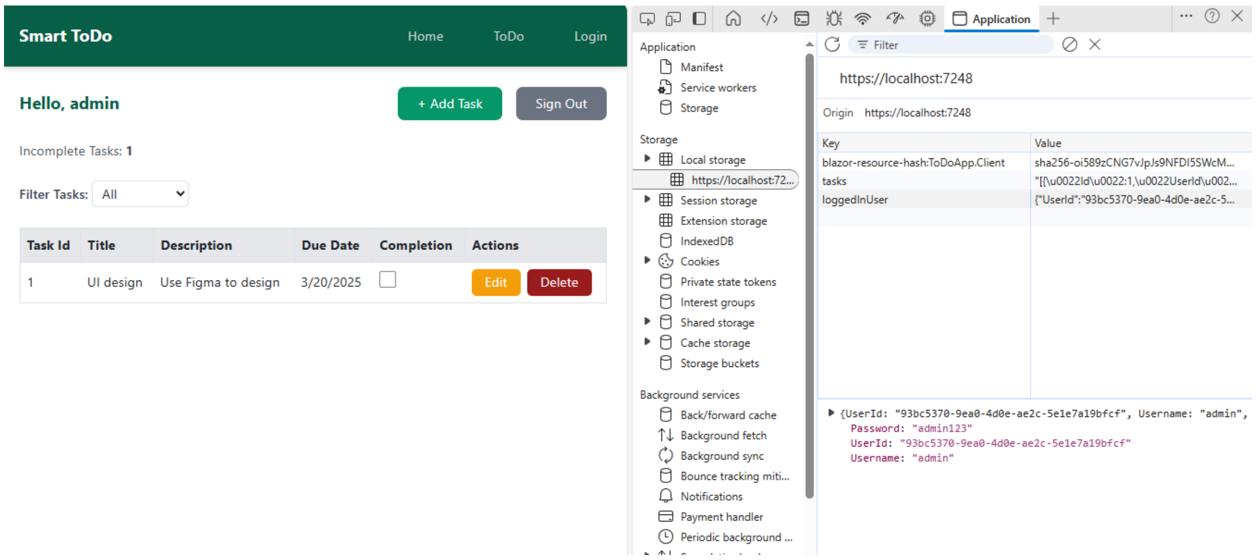
4.2 Additional Features

1. Task Filtering

- Filter tasks based on their completion status: All Tasks, Active Tasks (Incomplete), Completed Tasks

2. Persistent Storage

- Save & load tasks to/from browser Local Storage
- Users' login and tasks are saved



Hello, admin

Incomplete Tasks: **1**

Filter Tasks: All

Task Id	Title	Description	Due Date	Completion	Actions
1	UI design	Use Figma to design	3/20/2025	☐	Edit Delete

https://localhost:7248

Origin https://localhost:7248

Key	Value
blazor-resource-hash:ToDoApp.Client	sha256-oi589zCNG7vJpJs9NFDI5SWcM...
tasks	"[{\u0022Id\u0022:1,\u0022UserId\u002...
loggedInUser	{"UserId":"93bc5370-9ea0-4d0e-ae2c-5...

```
▶ {UserId: "93bc5370-9ea0-4d0e-ae2c-5e1e7a19bfcf", Username: "admin",
   Password: "admin123"
   UserId: "93bc5370-9ea0-4d0e-ae2c-5e1e7a19bfcf"
   Username: "admin"
```

3. Authentication System

- Implemented dummy user authentication
- Login functionality with username & password
- Only logged-in users can access the ToDo list
- User's session is saved in Local Storage
- Logout functionality to clear session

4. Responsive & Modern UI

- Used Tailwind CSS for styling
- Applied rounded buttons, proper spacing, and shadows
- Dropdown menus, forms, and task lists are mobile-friendly
- Dark green & white theme with natural black fonts

5. Modal-Based Task Creation & Editing

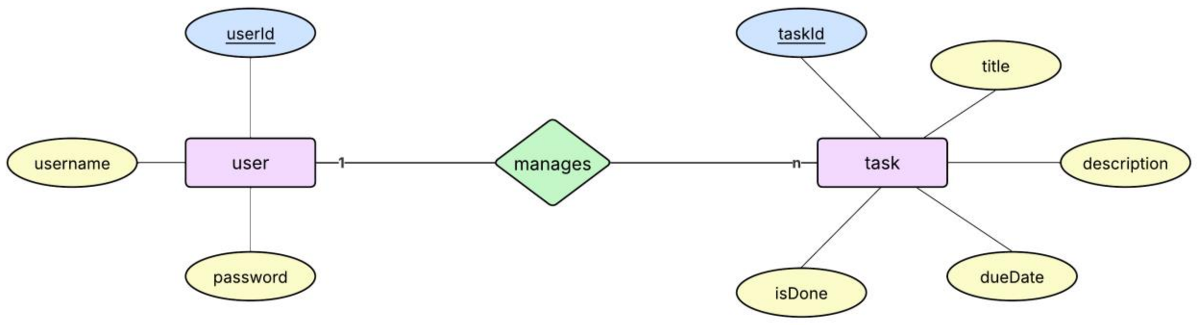
- Used modals for adding and editing tasks instead of separate pages
- Prevents clutter and improves user experience
- Forms are reset (cleared) after saving or canceling

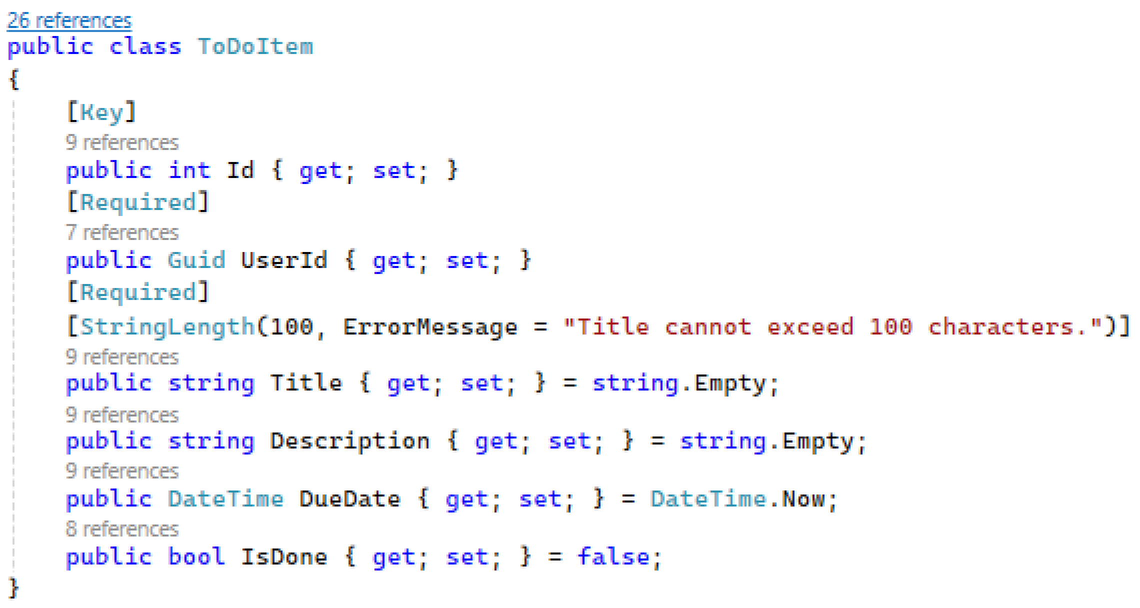
6. Logging & Debugging

- Implemented console logging for debugging authentication and task operations
- Used `ILogger` for tracking login attempts, task modifications, and errors

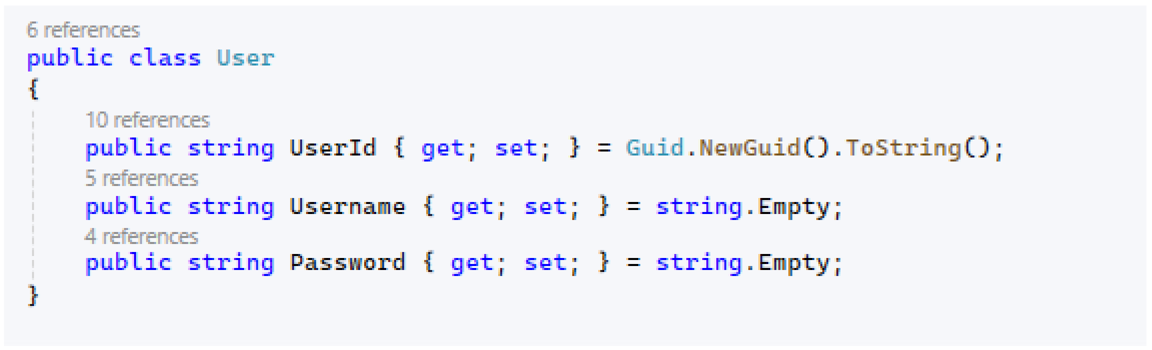
5. Database Design

The application currently does not have a database. It uses local storage for now, but still there are data models and an ideal database design to fit the current implementation would be:





```
26 references
public class ToDoItem
{
    [Key]
    9 references
    public int Id { get; set; }
    [Required]
    7 references
    public Guid UserId { get; set; }
    [Required]
    [StringLength(100, ErrorMessage = "Title cannot exceed 100 characters.")]
    9 references
    public string Title { get; set; } = string.Empty;
    9 references
    public string Description { get; set; } = string.Empty;
    9 references
    public DateTime DueDate { get; set; } = DateTime.Now;
    8 references
    public bool IsDone { get; set; } = false;
}
```



```
6 references
public class User
{
    10 references
    public string UserId { get; set; } = Guid.NewGuid().ToString();
    5 references
    public string Username { get; set; } = string.Empty;
    4 references
    public string Password { get; set; } = string.Empty;
}
```

7. Test Cases

1. User Login - Valid Credentials
Test ID: TC001
Description: Verify that a user can successfully log in with valid credentials.
Prerequisites: The application must be running, and the user must exist in the dummy user list.
Steps:

1. Navigate to the /login page.
2. Enter a valid username (admin) and password (admin123).
3. Click the "Login" button.

Expected Result:

- The user is redirected to the /todo page.
- The username is displayed on the page.

2. User Login - Invalid Credentials
Test ID: TC002
Description: Verify that an error message is shown for incorrect login credentials.
Prerequisites: The application must be running.
Steps:

1. Navigate to the /login page.
2. Enter an incorrect username (wrongUser) and password (wrongPass).
3. Click the "Login" button.

Expected Result:

- An error message "Invalid credentials!" is displayed.
- The user is not redirected to the /todo page.

3. Add a New Task
Test ID: TC003
Description: Verify that a user can add a new task successfully.
Prerequisites: User must be logged in.
Steps:

1. Click the + Add Task button.
2. Enter a title, description, and due date in the modal.
3. Click the Add Task button.

Expected Result:

- The task appears in the task list.
- The task is stored in Local Storage.

4. Task Filter - Completed Tasks
Test ID: TC004
Description: Verify that only completed tasks are displayed when the filter is set to "Completed".
Prerequisites: At least one completed task must exist.
Steps:

1. Mark a task as done (checkbox checked).
2. Change the filter dropdown to "Completed".

Expected Result:

- Only completed tasks are shown in the table.

5. Task Filter - Active Tasks
Test ID: TC005
Description: Verify that only active (incomplete) tasks are displayed when the filter is set to "Active".
Prerequisites: At least one incomplete task must exist.
Steps:

1. Ensure that at least one task is unchecked (not completed).
2. Change the filter dropdown to "Active".

Expected Result:

- Only incomplete tasks are displayed.

6. Edit an Existing Task
Test ID: TC006
Description: Verify that a user can successfully edit a task.
Prerequisites: A task must exist in the list.
Steps:

1. Click the Edit button on a task.
2. Modify the title, description, and due date.
3. Click Update.

Expected Result:

- The updated task details are saved and displayed.

7. Delete a Task
Test ID: TC007
Description: Verify that a task can be deleted successfully.
Prerequisites: At least one task must exist in the list.
Steps:

1. Click the Delete button for a task.
2. Confirm that the task is removed from the list.

Expected Result:

- The task is removed from the task list.

8. Logout Functionality
Test ID: TC008
Description: Verify that the user can log out.
Prerequisites: User must be logged in.
Steps:

1. Click the **Sign Out** button.

Expected Result:

- The user is redirected to the /login page.
- The session is cleared from Local Storage.

9. User Data Persists
Test ID: TC009
Description: Verify that the user’s data remain through the session.
Prerequisites: User must be logged in.
Steps:

1. Log in as a user.
2. Add several tasks.
3. Sign out.
4. Log in again.

Expected Result:

- The user's tasks are still displayed.

10. Input Validations - Empty Task Fields
Test ID: TC010
Description: Verify that adding a task without a title shows an error.
Prerequisites: User must be logged in.
Steps:

1. Click + Add Task.
2. Leave the title empty.
3. Click Add Task.

Expected Result:

- An error message is displayed: "Title is required".
- The task is not added.

8. User Guide

Step 1: Clone the repository

```
git clone https://github.com/NishuDissanayake/Smart-ToDo.git
cd Smart-ToDo/src/ToDoApp/ToDoApp/ToDoApp
```

Step 2: Install dependencies

```
dotnet restore
```

Step 3: Run the application

```
dotnet run
```

Step 4: Open in browser

http://localhost:5203/

You should see the Smart ToDo Homepage.

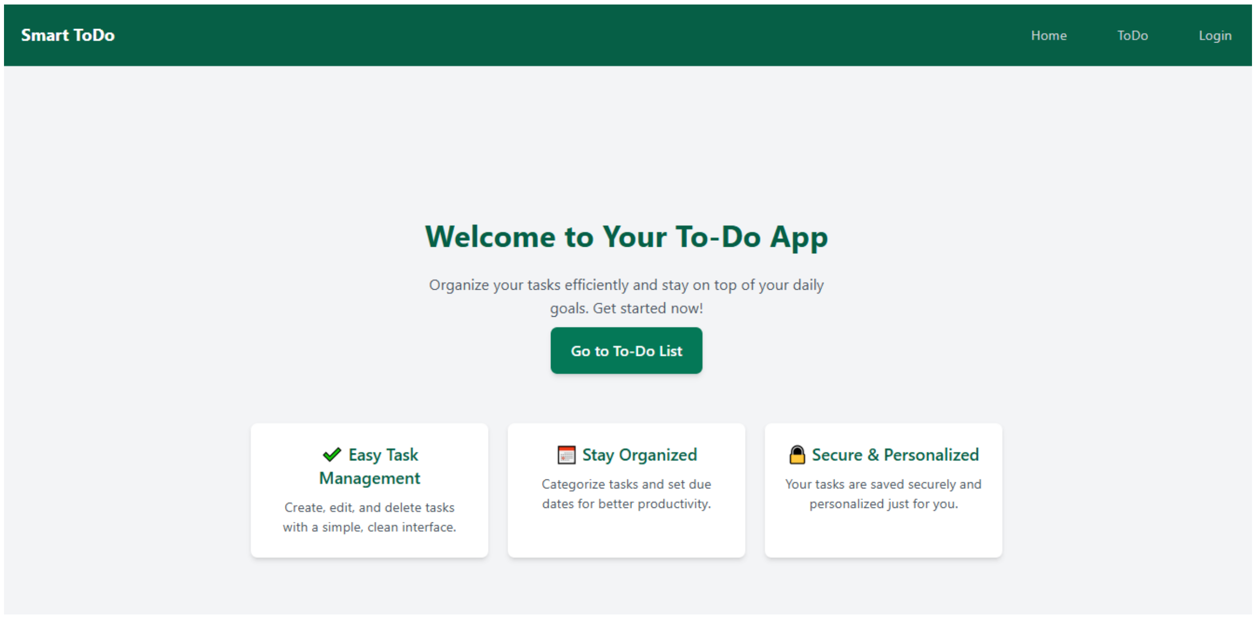
Step 5: Use dummy credentials to log in

admin	admin123
user1	password1
user2	password2

6. UI/UX

The theme mainly follows green colored elements on a white background, alternating between white and black texts as necessary.

1. Home



Welcome to Your To-Do App

Organize your tasks efficiently and stay on top of your daily goals. Get started now!

✓ Easy Task Management

Create, edit, and delete tasks with a simple, clean interface.

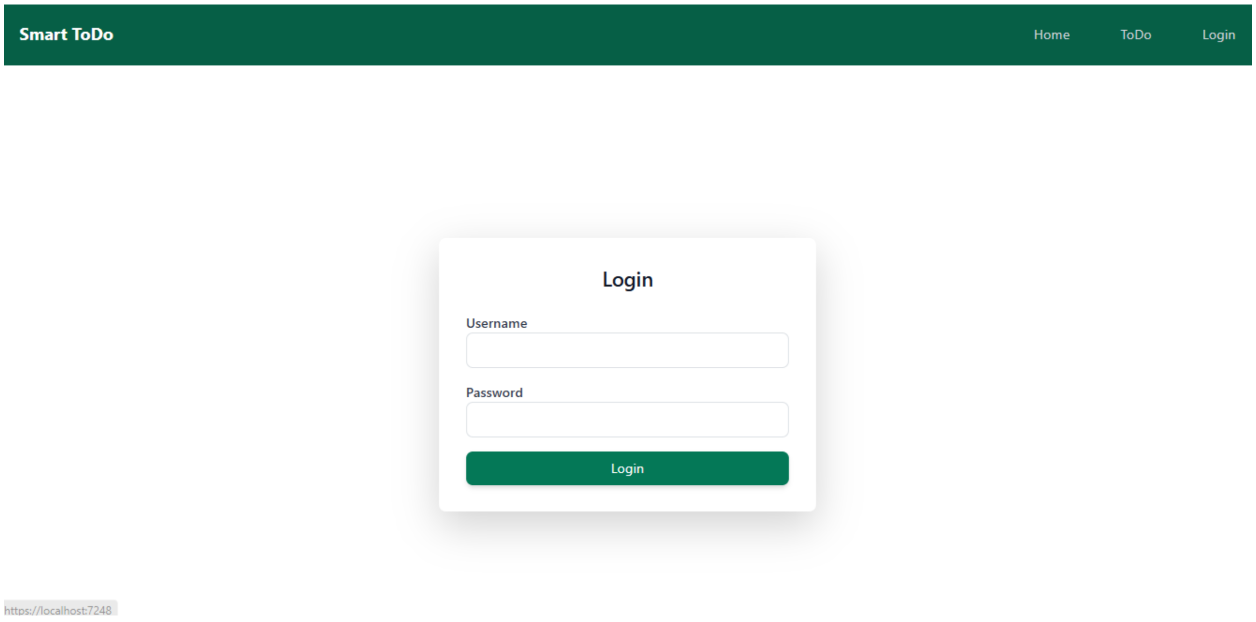
Stay Organized

Categorize tasks and set due dates for better productivity.

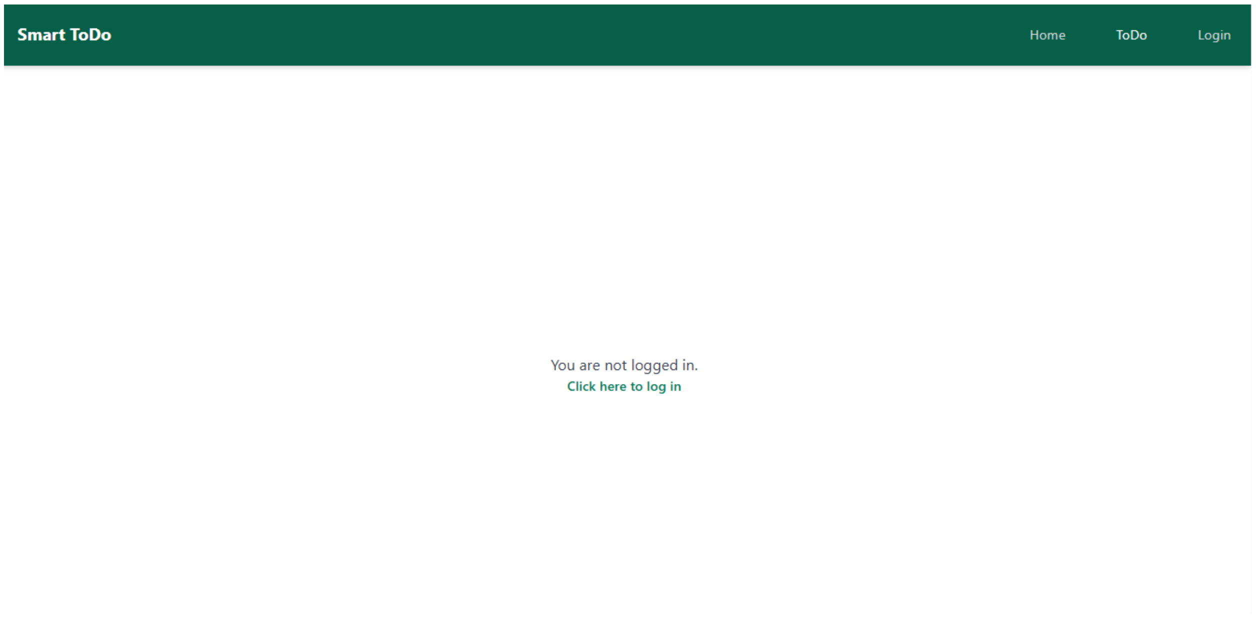
Secure & Personalized

Your tasks are saved securely and personalized just for you.

2. User login

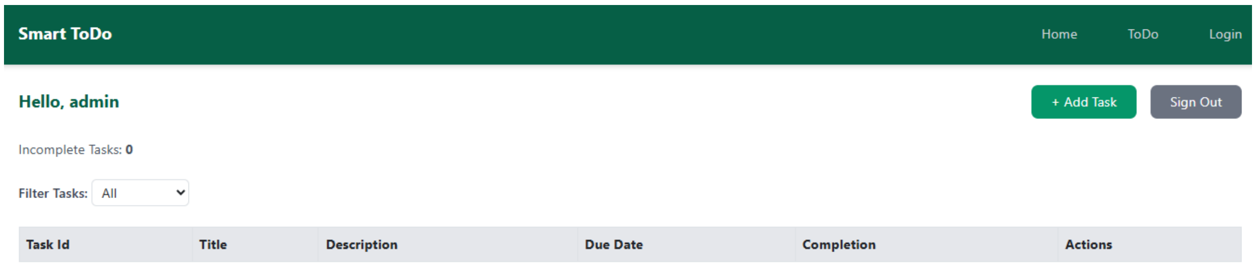


3. ToDo page (without authentication)



You are not logged in.
Click here to log in

4. ToDo page (authenticated)



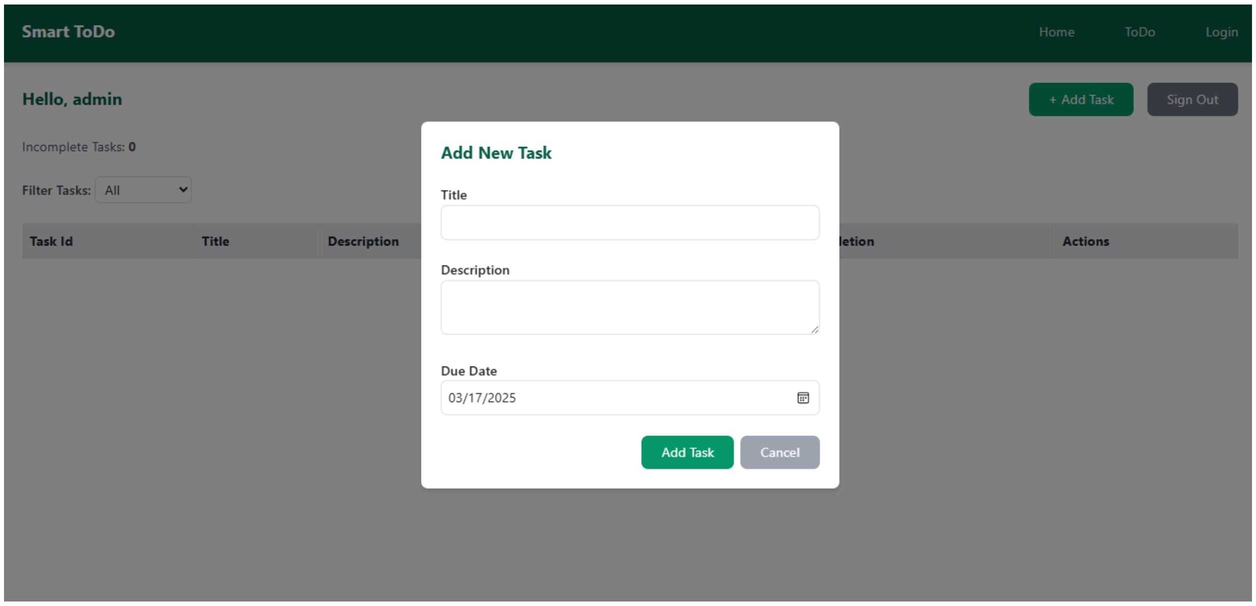
Hello, admin

Incomplete Tasks: **0**

Filter Tasks: All

Task Id	Title	Description	Due Date	Completion	Actions

5. Add task



Incomplete Tasks: **0**

Filter Tasks: All

Task Id	Title	Description	…letion	Actions

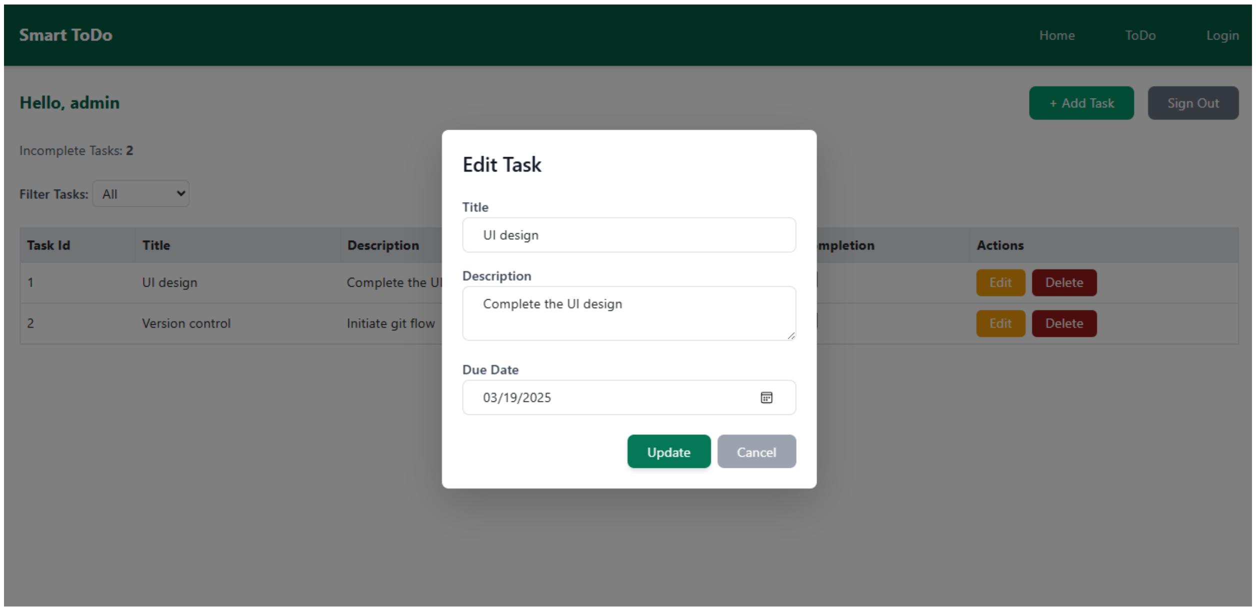
Add New Task

Title

Description

Due Date: 03/17/2025

6. Update task



Incomplete Tasks: **2**

Filter Tasks: All

Task Id	Title	Description	…mpletion	Actions
1	UI design	Complete the UI		Edit Delete
2	Version control	Initiate git flow		Edit Delete

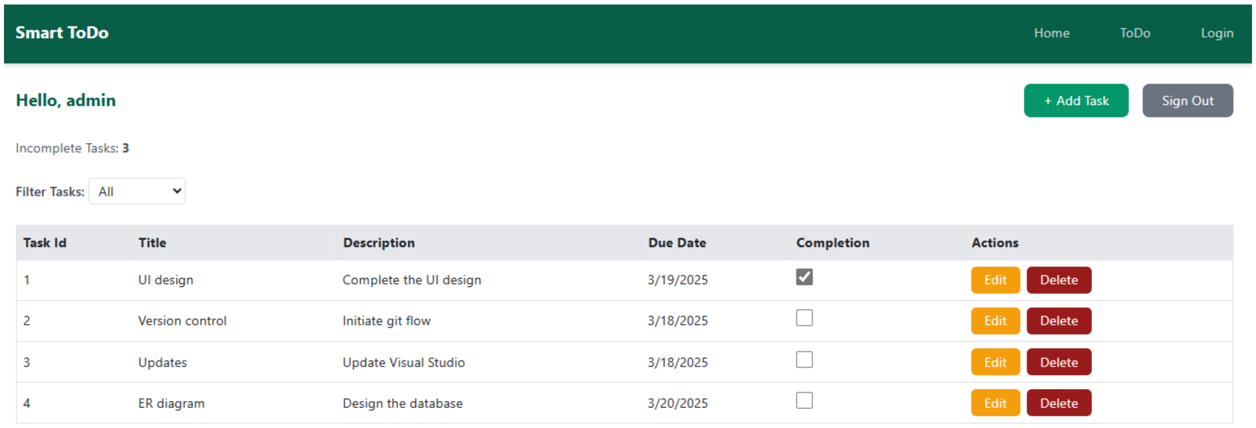
Edit Task

Title: UI design

Description: Complete the UI design

Due Date: 03/19/2025

7. Filtering

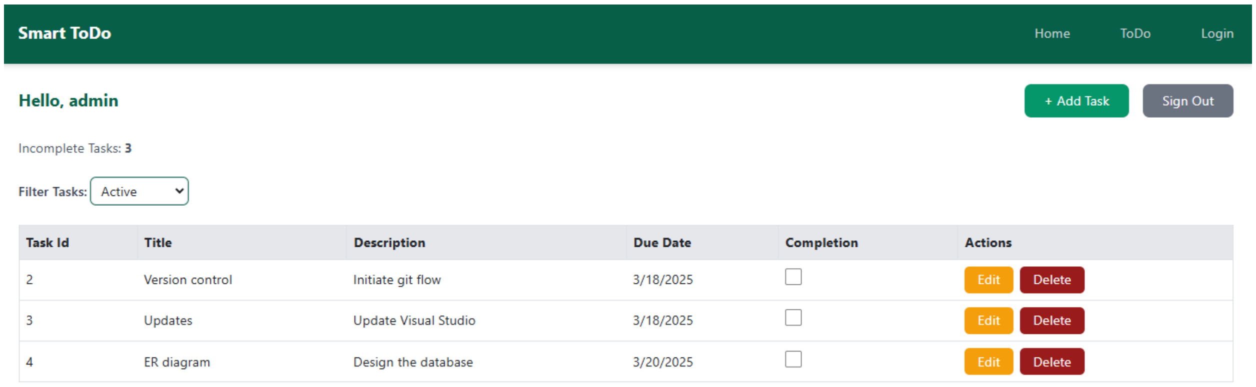


Hello, admin

Incomplete Tasks: **3**

Filter Tasks: All

Task Id	Title	Description	Due Date	Completion	Actions
1	UI design	Complete the UI design	3/19/2025	☑	Edit Delete
2	Version control	Initiate git flow	3/18/2025	☐	Edit Delete
3	Updates	Update Visual Studio	3/18/2025	☐	Edit Delete
4	ER diagram	Design the database	3/20/2025	☐	Edit Delete

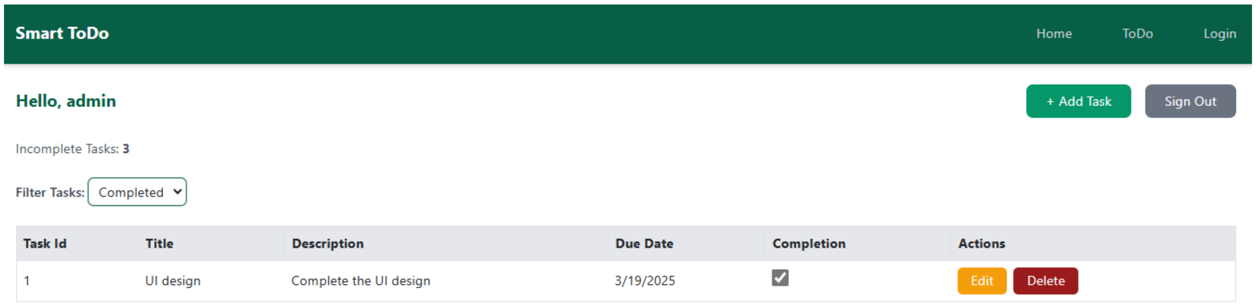


Hello, admin

Incomplete Tasks: **3**

Filter Tasks: Active

Task Id	Title	Description	Due Date	Completion	Actions
2	Version control	Initiate git flow	3/18/2025	☐	Edit Delete
3	Updates	Update Visual Studio	3/18/2025	☐	Edit Delete
4	ER diagram	Design the database	3/20/2025	☐	Edit Delete



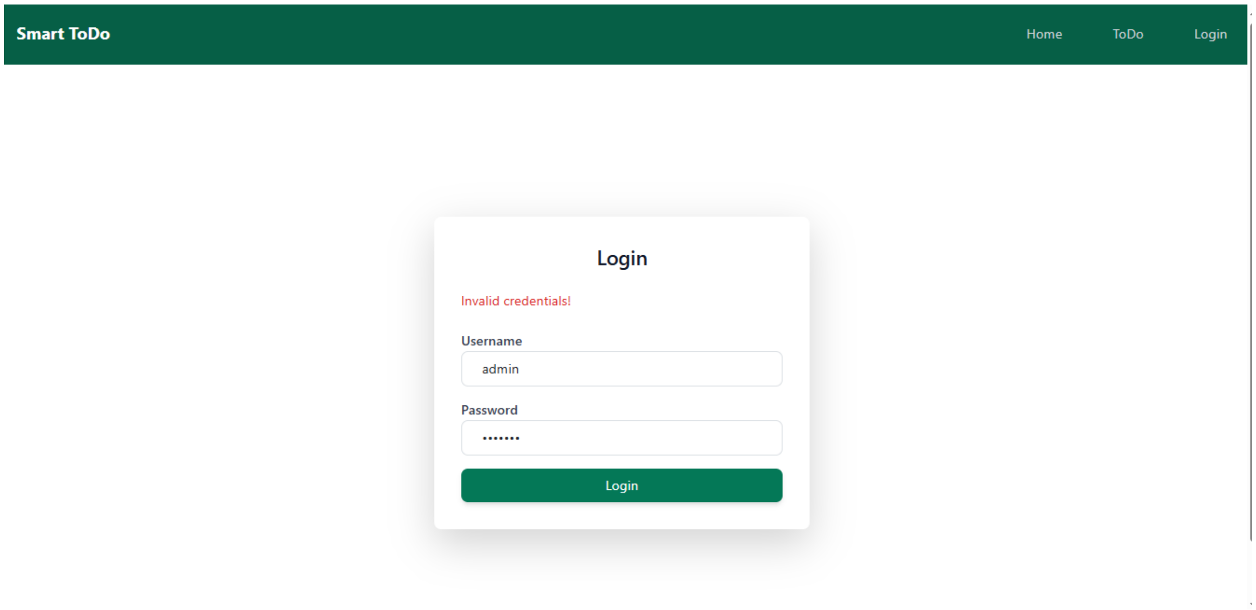
Hello, admin

Incomplete Tasks: **3**

Filter Tasks: Completed

Task Id	Title	Description	Due Date	Completion	Actions
1	UI design	Complete the UI design	3/19/2025	☑	Edit Delete

8. Invalid Credentials



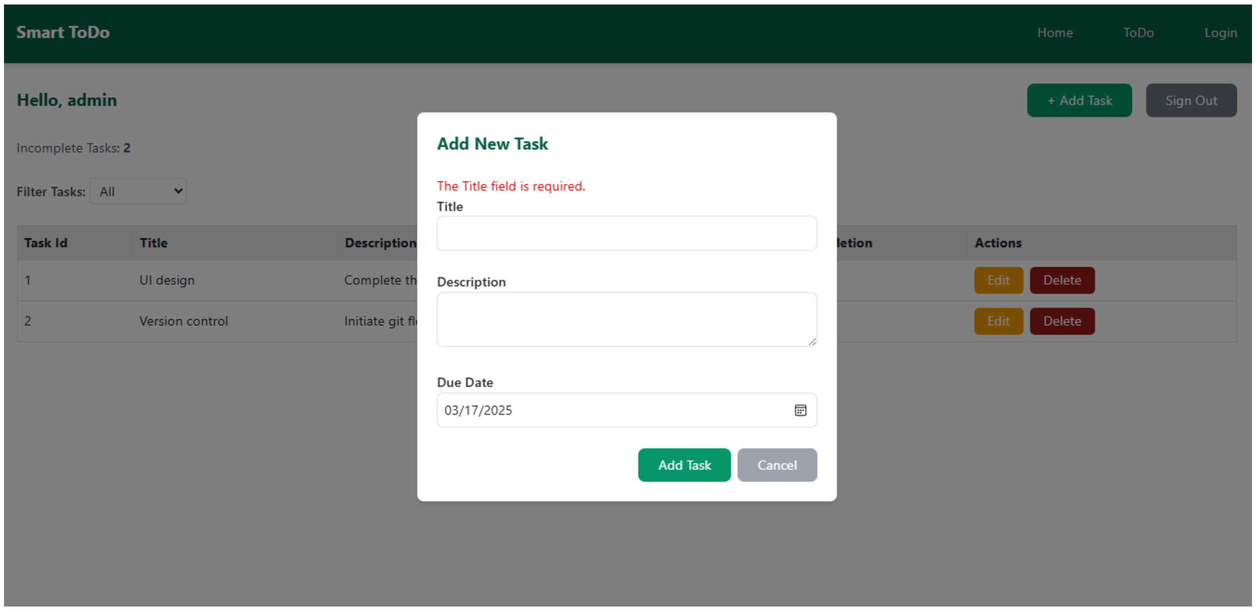
Login

Invalid credentials!

Username: admin

Password: •••••••

9. Validation



Hello, admin

Incomplete Tasks: **2**

Filter Tasks: All

Task Id	Title	Description	letion	Actions
1	UI design	Complete th		Edit Delete
2	Version control	Initiate git fl		Edit Delete

Add New Task

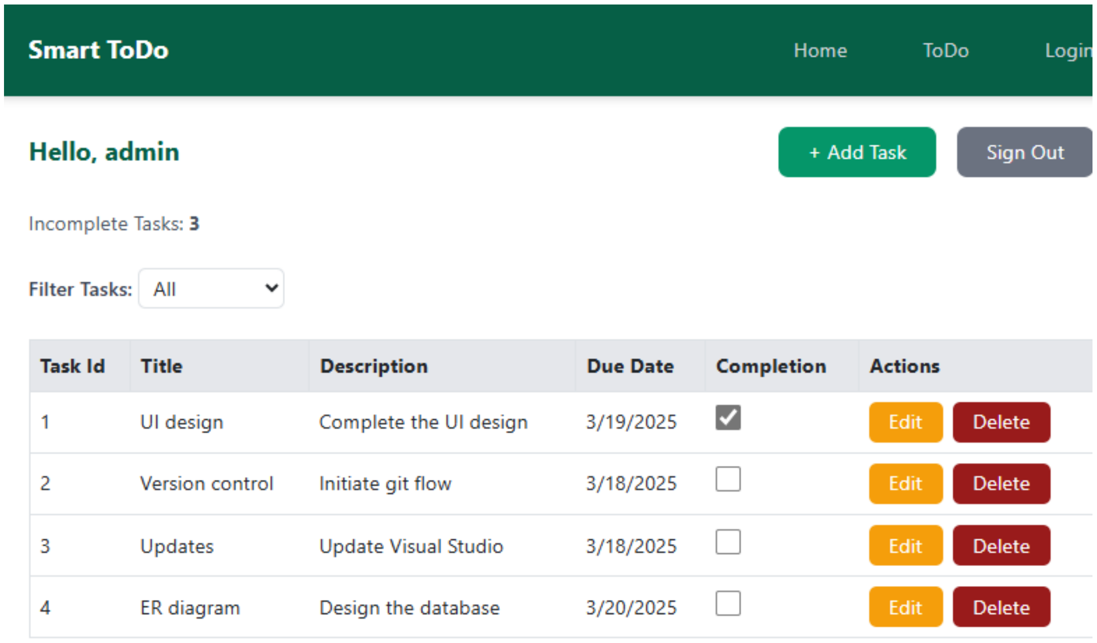
The Title field is required.

Title:

Description:

Due Date: 03/17/2025

10. Responsiveness



Smart ToDo — Home | ToDo | Login

Hello, admin

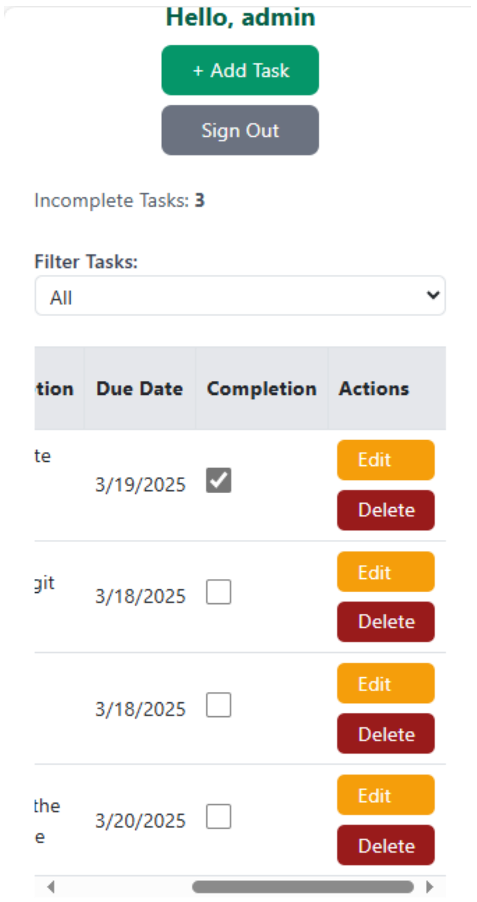
+ Add Task | Sign Out

Incomplete Tasks: **3**

Filter Tasks: All

Task Id	Title	Description	Due Date	Completion	Actions
1	UI design	Complete the UI design	3/19/2025	✓	Edit Delete
2	Version control	Initiate git flow	3/18/2025	☐	Edit Delete
3	Updates	Update Visual Studio	3/18/2025	☐	Edit Delete
4	ER diagram	Design the database	3/20/2025	☐	Edit Delete

Table scroll effect:



Hello, admin

+ Add Task

Sign Out

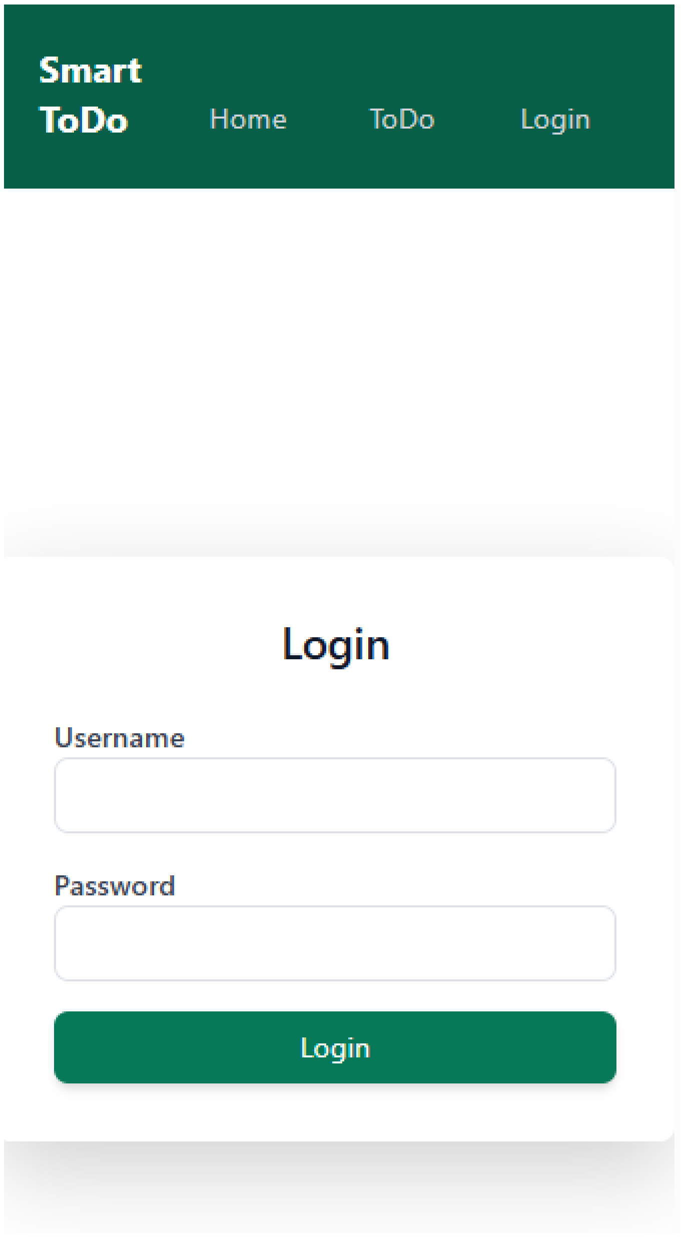
Incomplete Tasks: **3**

Filter Tasks:

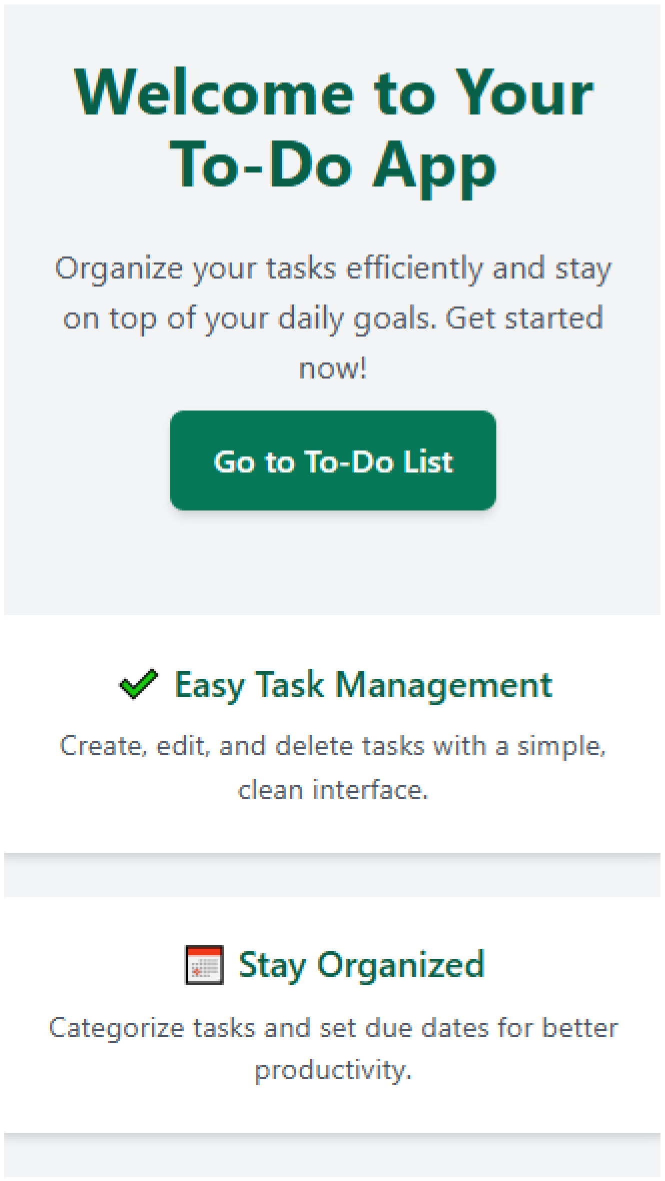
All

tion	Due Date	Completion	Actions
te	3/19/2025	✓	Edit Delete
ȷit	3/18/2025	☐	Edit Delete
	3/18/2025	☐	Edit Delete
the e	3/20/2025	☐	Edit Delete

Login (Responsive)



Home (Responsive)



Welcome to Your To-Do App

Organize your tasks efficiently and stay on top of your daily goals. Get started now!

✓ Easy Task Management

Create, edit, and delete tasks with a simple, clean interface.

📅 Stay Organized

Categorize tasks and set due dates for better productivity.