**Smart ToDo**

**System Information**

A person holding a pencil next to a clipboard

AI-generated content may be incorrect.

**Developer:** Nishani Dissanayake

**Date:** 17. 03. 2025

Table of Contents

[**1. Introduction** 2](#_Toc193106348)

[**2. Project Structure** 2](#_Toc193106349)

[**3. Technologies Used** 2](#_Toc193106350)

[**4. Features Implemented** 2](#_Toc193106351)

[**5. User Login** 2](#_Toc193106352)

[**6. Database Design** 2](#_Toc193106353)

[**7. UI/UX** 2](#_Toc193106354)

[**8. Test Cases** 2](#_Toc193106355)

# **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

A screenshot of a computer

AI-generated content may be incorrect.

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.

A screenshot of a computer

AI-generated content may be incorrect.

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

**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**

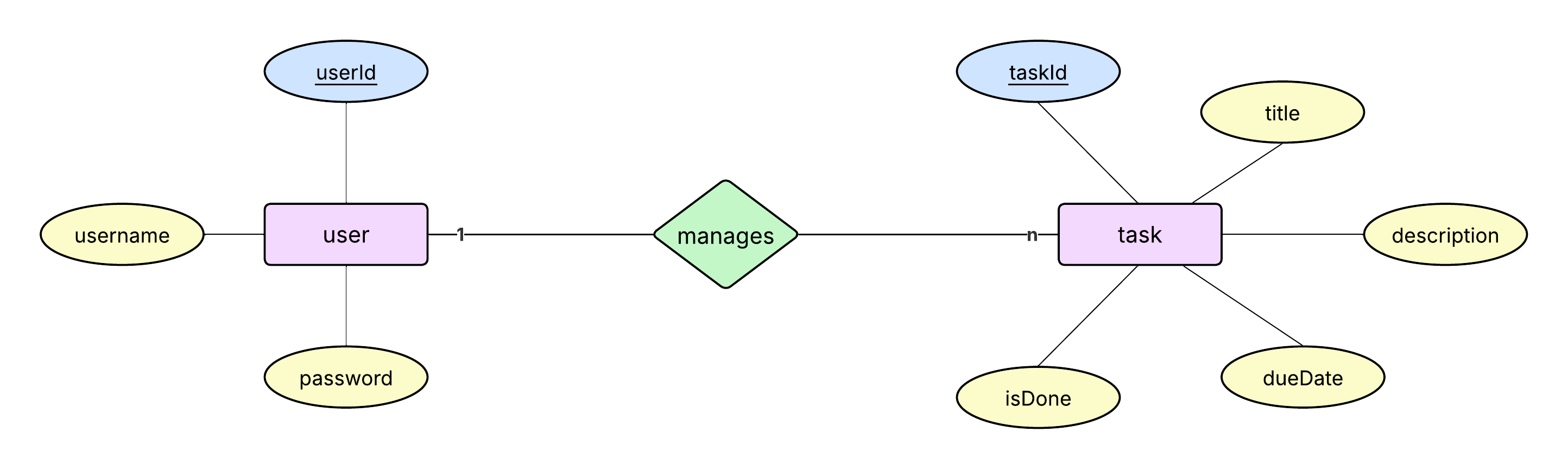
* Used modals for adding and editing tasks instead of separate pages
* Prevents clutter and improves user experience
* Forms are reset (cleared) after saving or canceling

**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:



# **6. UI/UX**

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

**Home**

**User login**

**ToDo page (without authentication)**

**ToDo page (authenticated)**

**Add task**

**Update task**

**Filtering**

**Responsiveness**

# **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  cd ToDoApp |

**Step 2: Install dependencies**

|  |
| --- |
| dotnet restore |

**Step 3: Run the application**

|  |
| --- |
| dotnet run |

**Step 4: Open in browser**

[**https://localhost:5001**](https://localhost:5001)

You should see the Smart ToDo Homepage.