Technical Documentation

Task Manager Application

A person sitting at a desk working on a computer

Description automatically generated

Table of Contents

[Diagrams 4](#_Toc180453758)

[Unified Modeling Language (UML) 4](#_Toc180453759)

[Class Relationships and Design Patterns 5](#_Toc180453760)

[Sequence Diagram 6](#_Toc180453761)

[Task Creation 6](#_Toc180453762)

[Activity Diagram 7](#_Toc180453763)

[Task Completion 7](#_Toc180453764)

[Component Diagram 9](#_Toc180453765)

[State Diagram 10](#_Toc180453766)

Architecture Overview

The application follows a Client-Server Model using the MVC (Model-View-Controller) architecture:

* **Client (Front-End):** Built with WPF, providing a dynamic user interface. Users interact with the application through various UI components, allowing them to create, view, and manage tasks easily.
* **Server (Back-End):** Developed using the .NET Framework, handling business logic and data processing. It communicates with the client to perform operations based on user input.
* **Database:**
  + **File Storage:** Utilizes local file storage (e.g., JSON, XML, or plain text files) to save and retrieve task data, ensuring data persistence.
  + **Data Access Layer:** A layer that abstracts the file operations, handling reading from and writing to the data files, which ensures separation of concerns within the application architecture.
* **Deployment:** The application is designed as a desktop application, suitable for Windows environments, with considerations for future enhancements to support cloud-based storage.

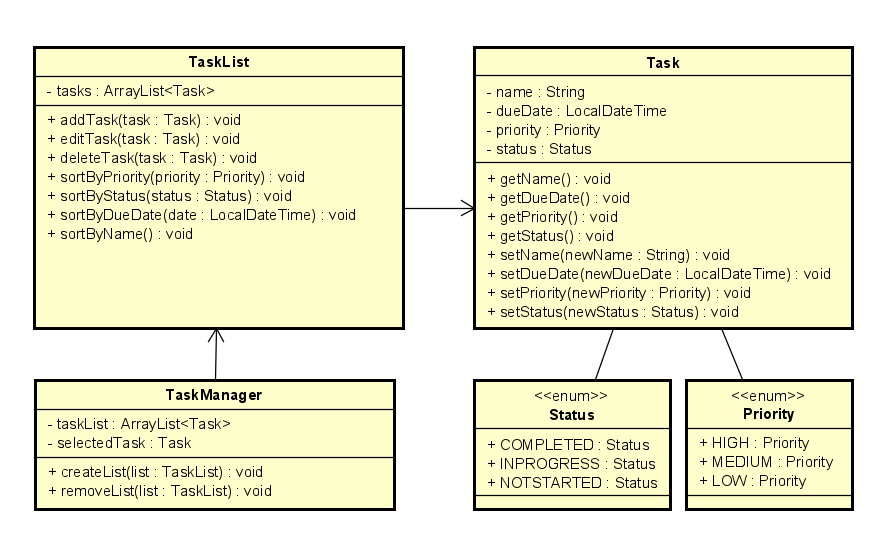
# Diagrams

## Unified Modeling Language (UML)

|  |  |  |
| --- | --- | --- |
| Class | Description | Methods |
| TaskManager | Contains a list of task lists (TaskList objects) and tracks the currently selected task. Acts as the "Controller" in an MVC architecture. | Methods for creating, removing, and updating task lists, along with methods for sorting and filtering tasks at the application level. |
| TaskList | Represents a list of tasks, containing a collection of Task objects. | Methods for adding, editing, removing tasks, and performing list-level operations like sorting, filtering, or searching tasks. |
| Task | Represents an individual task with attributes including name, description, due date, priority, and status. | Implements INotifyPropertyChanged to notify the UI when properties (such as status) are updated. |
| Priority | An Enum representing task priority levels (High, Medium, Low). | Used for prioritizing tasks for better management and display. |
| Status | An Enum representing task status (Completed, In Progress, Not Started). | Provides the foundation for task tracking and filtering. |
| TaskView | A UI component displaying details for a selected task. | Provides buttons and forms to edit, mark as completed, or delete tasks, communicating changes to TaskList. |
| ListView | Displays all tasks in a selected TaskList. | Allows users to sort, filter, or search tasks, updating based on the TaskList data source. |

## Class Relationships and Design Patterns

|  |  |
| --- | --- |
| Name | Description |
| TaskManager | Follows the Singleton pattern to ensure a single point of control for task management across the application. |
| TaskList | Contains multiple Task objects, forming a one-to-many relationship. |
| Task | Implements the Observer pattern using events (e.g., INotifyPropertyChanged) to notify the UI when task properties change, ensuring real-time updates across relevant views. |
| TaskView and ListView | Bound to the user interface using data-binding principles; UI views are automatically updated whenever task attributes change. |
| Status and Priority | Used for filtering, sorting, and intuitively displaying tasks. They define business logic (e.g., only "In Progress" tasks are displayed by default on the home screen). |

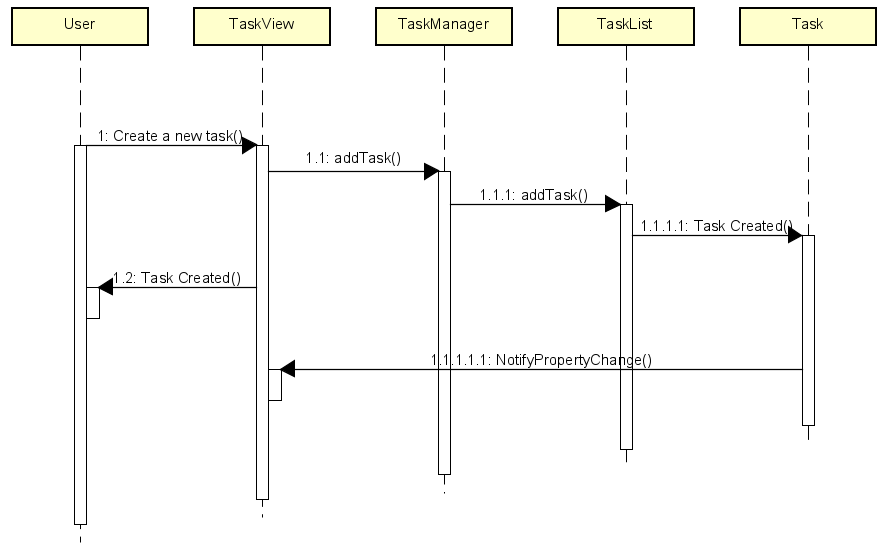


## Sequence Diagram

This section captures interactions between different components or objects within the system over time.

### Task Creation

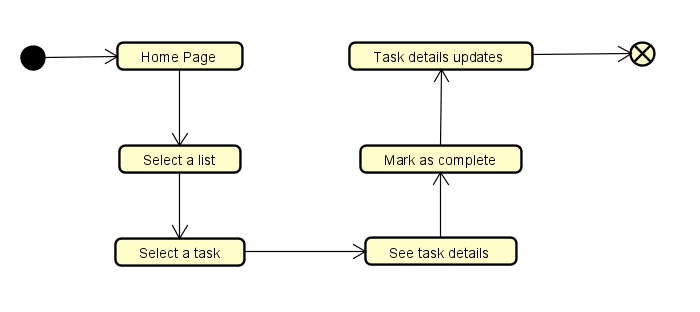
Outlines the steps for a user to create a task, involving appropriate methods for list and UI updates.



## Activity Diagram

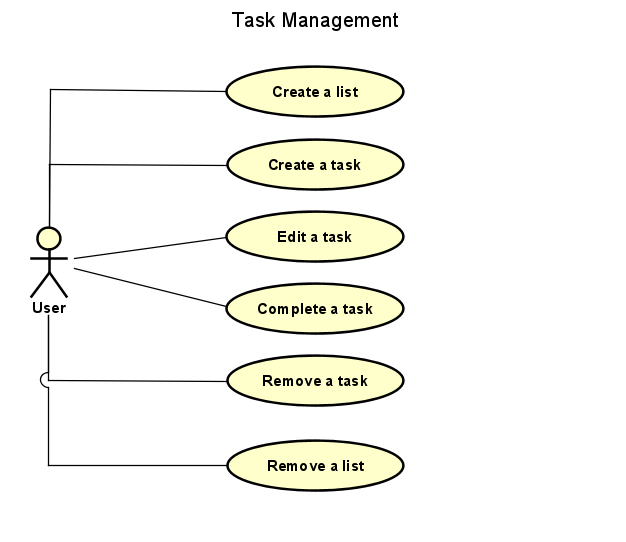
Displays the workflow of tasks and operations within the system.

### Task Completion

Illustrates the workflow for completing a task within the app.

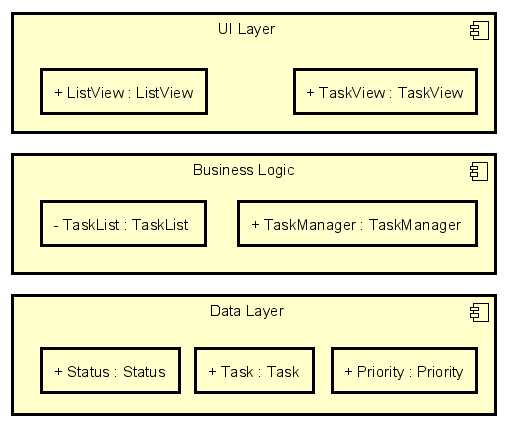
Use Case

Represents the various actions users can perform within the application.

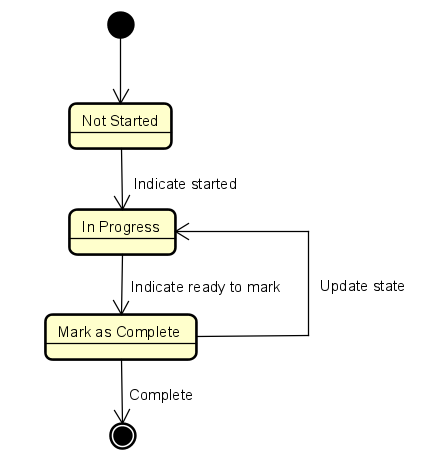


## Component Diagram

Highlights the high-level architecture of the application, showing the various components that constitute the system.



## State Diagram

Shows the process of a task and how its state can change.