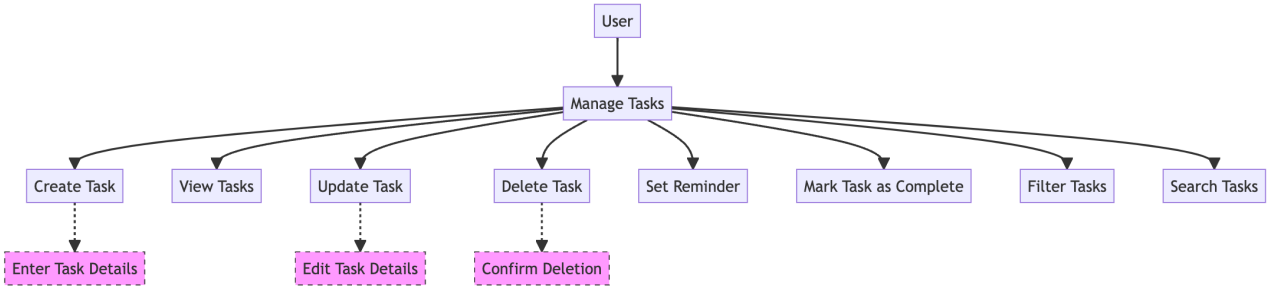
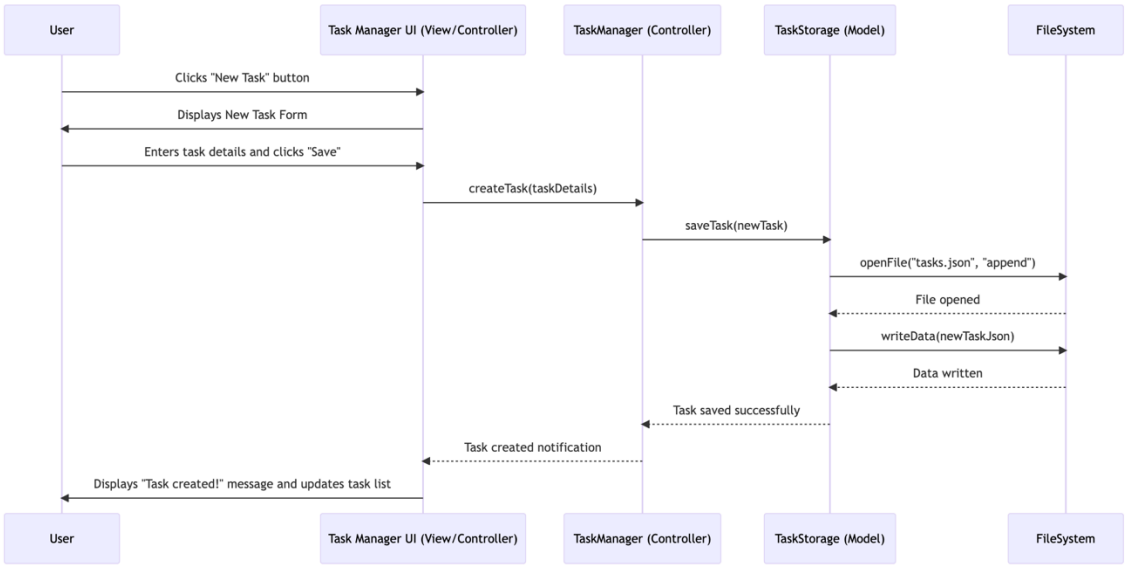
1. **Use Case diagram**



2. **Sequence Diagram**



3. The **Model** represents the application's data, business logic, and rules. It manages the data, responding to requests from the Controller and updating the View.

- **Responsibilities:**
  - **Data Management:** Stores and retrieves task data (e.g., task name, description, due date, status).
  - **Business Logic:** Implements rules for tasks (e.g., validating task details, marking tasks as complete, setting reminders).
  - **Data Persistence:** Handles interaction with the underlying data storage mechanism (in this case, reading from and writing to files). It doesn't know *how* the data is displayed, only that it's manipulated according to business rules.
  - **Notifications:** Notifies the Controller when its data changes.
- **Concerns:**
  - Data integrity and consistency.
  - Implementing core business rules.
  - Abstracting data storage details from the rest of the application.

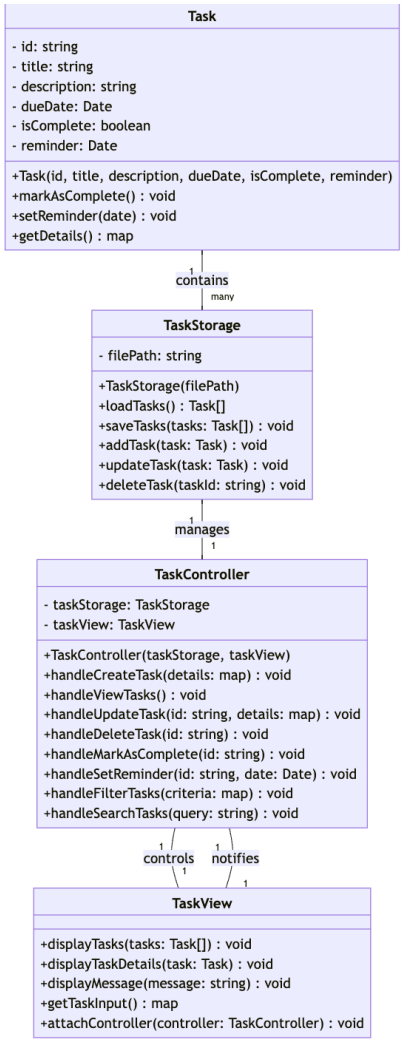
The **View** is responsible for presenting the data from the Model to the user. It's the user interface of the application.

- **Responsibilities:**
  - **Display:** Renders the user interface components (e.g., list of tasks, task details form, buttons).
  - **Presentation Logic:** Formats and presents the data received from the Model (via the Controller) in a user-friendly way.
  - **User Interaction:** Captures user input (e.g., clicks, text entry) and forwards it to the Controller.
- **Concerns:**
  - User experience and interface aesthetics.
  - Responsiveness and accessibility.
  - Displaying the current state of the Model.

The **Controller** acts as an intermediary between the Model and the View. It handles user input from the View, processes it, updates the Model, and then tells the View to update its display.

- **Responsibilities:**
  - **Input Handling:** Receives user input (e.g., "create new task," "edit task," "delete task") from the View.
  - **Logic Orchestration:** Translates user actions into operations on the Model. It determines which business logic should be executed based on user input.
  - **Model Interaction:** Calls methods on the Model to perform data manipulation (e.g., createTask(), updateTask(), deleteTask()).
  - **View Update:** Updates the View based on changes in the Model or the outcome of user actions. It might tell the View to re-render the task list or display a success message.
- **Concerns:**
  - Handling user events and requests.
  - Coordinating interactions between the Model and View.
  - Flow control of the application.

4. Class Diagram



5. CRUD Matrix

Entity	Role	Create	Read	Update	Delete
Task	**User**	Yes	Yes	Yes	Yes