**SOFTWARE DOCUMENTATION: TASK MANAGER CLI APPLICATION**

### **1. System Architecture**

The Task Manager CLI application is structured using Object-Oriented Programming (OOP) principles and follows a modular architecture to promote reusability, maintainability, and clarity.

#### **Key Layers:**

1. **Core Layer (OOP Classes)**
   * Task Class: Encapsulates properties of a task (title, description, due date, status, priority, etc.).
   * TaskManager Class: Handles business logic for managing the task list (CRUD operations).
2. **File Management Layer**  
   * Reads from and writes to a MongoDB Database.
3. **User Interface Layer (CLI)**
   * Uses Inquirer.js for interactive prompts.
   * Parses and handles user input from the terminal.
4. **Utilities Layer**
   * Contains helper functions for date formatting, validation, and filtering.

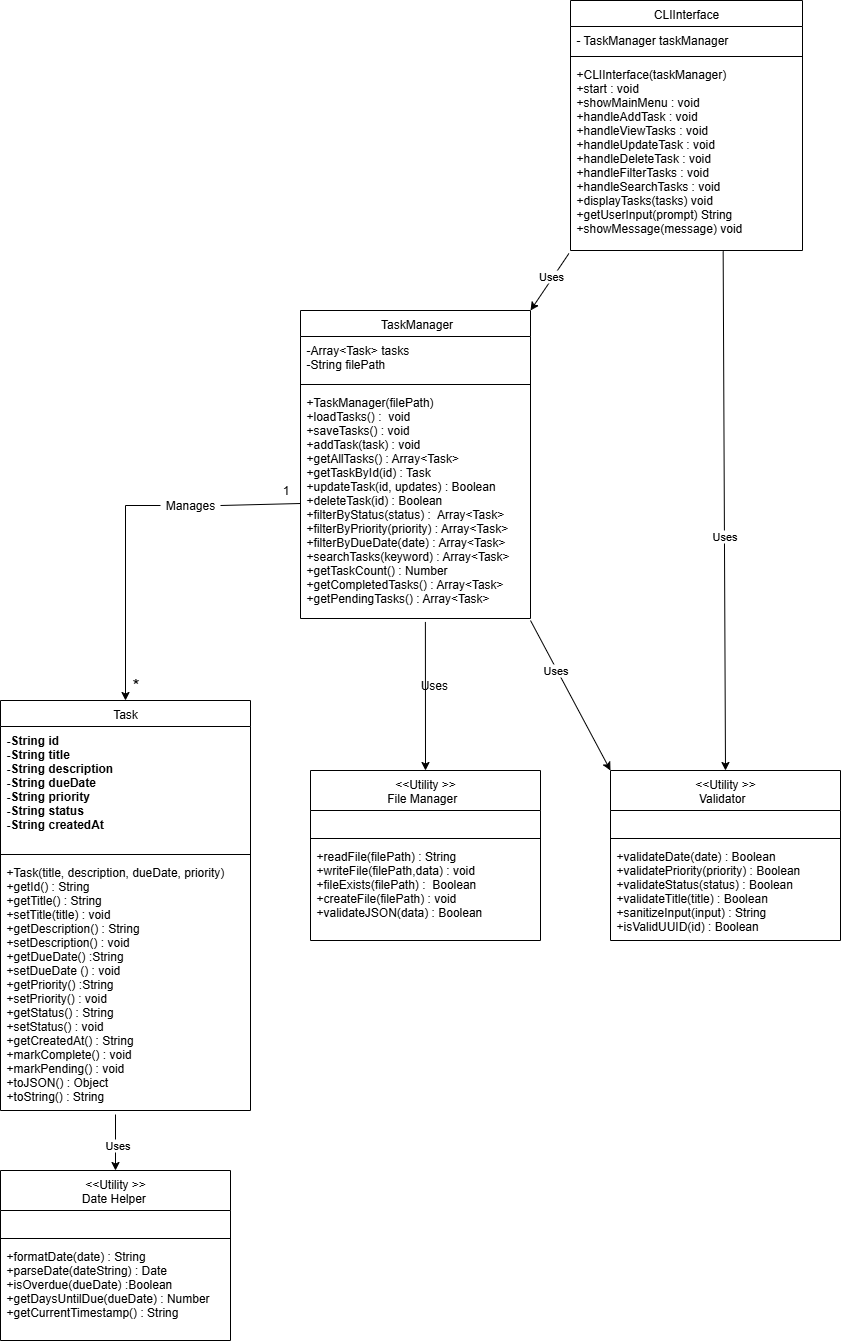
#### **Design Considerations:**

* **Encapsulation**: Each class manages its own data and behavior.
* **Abstraction**: Complex file operations and task filtering are abstracted behind simple interfaces.
* **Polymorphism**: In future, different task types can inherit from a base Task class and override behaviors.
* **Error Handling**: All file operations and user inputs are wrapped in try/catch blocks with user-friendly messages.

### **2. UML Diagrams**

#### **A. Class Diagram**

* Shows the structure and relationships of the core classes.

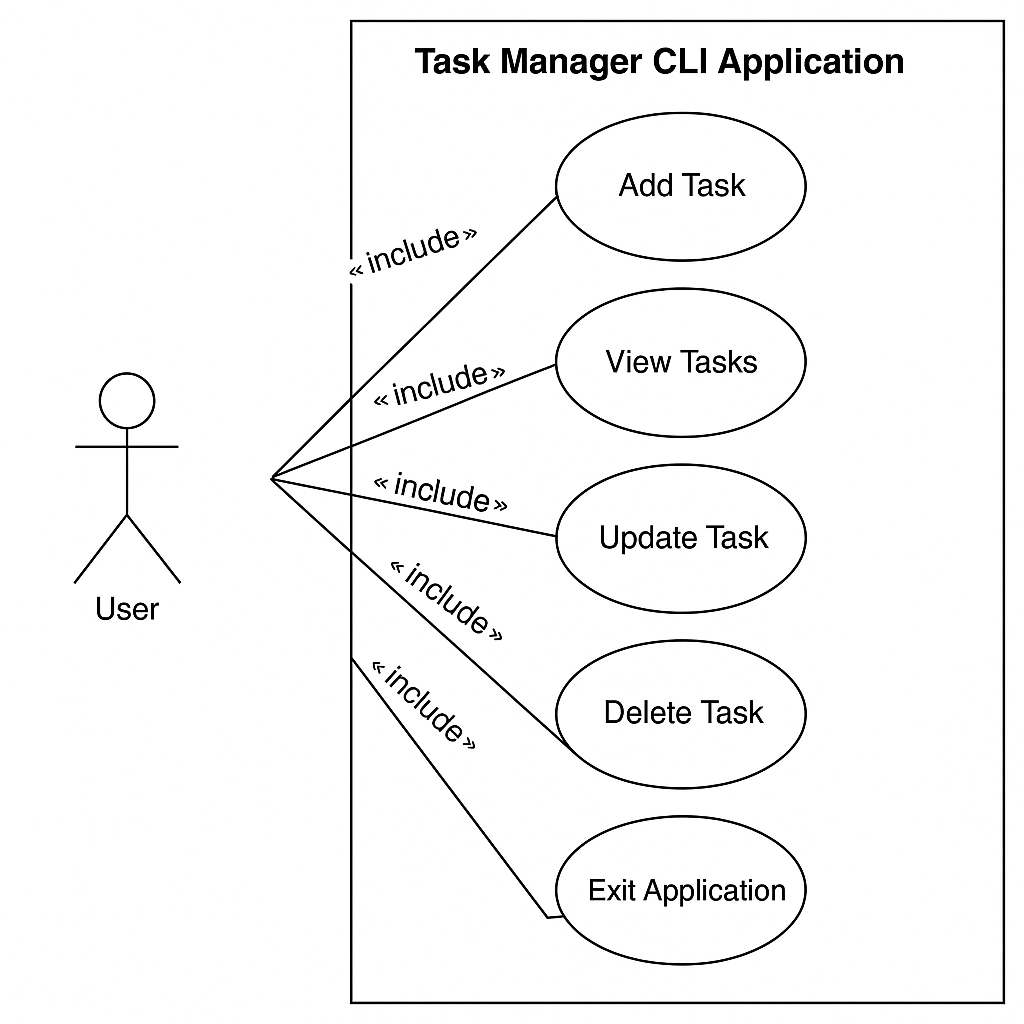


#### 

#### 

#### **B. Use Case Diagram**

* Visualizes the interaction between the user and the system.



### **3. API Documentation (N/A)**

This application is **not API-based**. It operates solely via the command line and local file storage. There are no RESTful endpoints.

### 

### 

### 

### **4. Database Schema (File-Based Equivalent)**

Since this project uses a traditional database instead of file storage and it’s stored in our mongodb database.

[

{

"id": "uuid",

"title": "Finish Assignment",

"description": "Complete the CLI app",

"dueDate": "2025-08-01",

"priority": "High",

"status": "Pending",

"createdAt": "2025-07-21T10:00:00Z"

},

...

]

Each task object includes:

* id: Unique identifier (UUID)
* title: Task name
* description: Optional details
* dueDate: Deadline
* priority: (Low, Medium, High)
* status: (Pending, Completed)
* createdAt: Timestamp of creation

### **5. Setup & Running Instructions**

#### **Prerequisites:**

* Node.js (v14 or higher)
* Git (for version control)

#### 

#### 

#### **Installation Steps:**

1. Clone the repository:

git clone https://github.com/Daniel-IRYIVUZE/task\_management\_group\_8

cd task\_management\_group\_8

1. Install dependencies:

npm install

1. Run the application:

node index.js

#### **Available Commands (via CLI prompts):**

* Add a new task
* View all tasks
* Update a task (mark complete, edit title/description)
* Delete a task
* Filter tasks by priority/status/date

#### **Directory Structure:**

task-manager-cli/

├── classes/

│ ├── Task.js

│ └── TaskManager.js

├── tests/

│ └── task.test.js

├── utils/

│ └── helpers.js

├── index.js

├── package.json

└── README.md

### 

### **5. Setup & Running Instructions**

#### **Prerequisites:**

* Node.js (v14 or higher)
* Git (for version control)

#### **Installation Steps:**

1. Clone the repository:

git clone https://github.com/your-username/task-manager-cli.git

cd task-manager-cli

1. Install dependencies:

npm install

1. Run the application:

node index.js

#### **Available Commands (via CLI prompts):**

* Add a new task
* View all tasks
* Update a task (mark complete, edit title/description)
* Delete a task
* Filter tasks by priority/status/date

#### **Directory Structure:**

task-manager-cli/

├── classes/

│ ├── Task.js

│ └── TaskManager.js

├── tests/

│ └── task.test.js

├── utils/

│ └── helpers.js

├── index.js

├── package.json

└── README.md