## **Project Proposal: Task Manager using OOP and File Management**

### **1. Detailed Description of the Chosen Problem**

In our fast-paced world, managing tasks efficiently is essential for personal productivity and team collaboration. However, many task management applications are either too complex, too expensive, or require an internet connection and advanced setup. There is a gap in providing a simple, offline, command-line task manager that is lightweight, user-friendly, and extensible for developers or non-technical users who want to manage daily tasks locally.

The specific problem this project aims to address is the lack of a lightweight, offline, and easy-to-use task management system for users who prefer or require command-line interfaces. Users often want to perform CRUD (Create, Read, Update, Delete) operations on their tasks without dealing with setup overhead, logins, or complex UIs. Moreover, in many educational or development scenarios, there's a need for learning tools that apply Object-Oriented Programming (OOP) principles and teach file persistence techniques without relying on full-scale databases.

### **2. Outline of the Proposed Solution**

We propose to develop a Task Manager CLI Application using Node.js and Object-Oriented Programming principles in JavaScript. The application will run entirely on the terminal and offer the following functionalities:

#### **Core Functionalities:**

* **Add Tasks**: Users can create a new task with a title, description, due date, and priority.
* **View Tasks**: List all tasks with filters like status (completed/pending), due dates, and priorities.
* **Update Tasks**: Modify any property of a task (e.g., mark as complete, change title, etc.).
* **Delete Tasks**: Remove a task permanently.
* **Persistent Storage**: All tasks will be stored in a database using MongoDB.
* **Search/Filter**: Allow users to search tasks based on keywords or filter by due date, status, or priority.

#### **Software Principles to be Applied:**

* Full use of Object-Oriented Programming (OOP) concepts including:  
  + **Encapsulation**: Task data will be wrapped inside task classes with proper accessors.
  + **Inheritance**: A possible hierarchy of task types (e.g., WorkTask, PersonalTask) for future extensibility.
  + **Polymorphism**: Common task operations will be applied polymorphically.
  + **Abstraction**: Hide complex implementation details via interfaces or base classes.
* Robust error handling and input validation.
* Clean modular code using best practices.

### **3. List of Technologies to be Used**

| **Component** | **Technology** |
| --- | --- |
| Programming Language | JavaScript |
| Runtime Environment | Node.js |
| File Storage | MongoDB |
| Development Tools | VS Code, ESLint, Prettier, Jest |
| Version Control | Git & GitHub |
| Documentation | Markdown |
| CLI Interaction | Inquirer.js (for improved CLI UX) |

### **4. Preliminary Project Timeline with Milestones**

| **Week** | **Milestone** |
| --- | --- |
| Day 1 | **Planning & Setup**: Finalize project proposal, set up repo and dev environment, research OOP structure and file system management. |
| Day 2 | **OOP Architecture Design**: Create classes (Task, TaskManager), write UML diagrams, plan command structure. |
| Day 3 | **Implement Core Features**: Add, view, update, delete tasks; implement file-based storage. |
| Day 4 | **Enhancements**: Add search, filters, and better CLI experience using Inquirer.js. Start testing modules. |
| Day 5 | **Finalization**: Complete documentation, polish code, write setup instructions, test thoroughly. |
| Day 6 | **Submission & Review**: Review all deliverables, finalize working application, submit. |

### **5. Division of Responsibilities Among Group Members**

#### **1. Daniel Iryivuze – Project Lead & OOP Architect**

* Coordinate overall project direction and ensure adherence to the timeline and milestones.
* Design the system’s OOP architecture, including the class structure and relationships.
* Lead the development of core logic for task operations using encapsulation, inheritance, and abstraction.
* Create UML class diagrams and contribute to architectural documentation.
* Manage GitHub repository setup, pull requests, and version control standards.

#### **2. Yusuf Molumo – Backend & File System Engineer**

* Implement all task-related logic for adding, updating, viewing, and deleting tasks.
* Handle data persistence using Node.js to read/write task data to MongoDB.
* Ensure robust error handling and input validation throughout the system.
* Develop helper modules and utilities that support the main application.

#### **3. Latjor Wuon Lat Dak – CLI Interaction & User Experience Developer**

* Design and implement an intuitive and user-friendly command-line interface using libraries like Inquirer.js.
* Handle user input/output flow and provide filtering, search, and feedback mechanisms in the CLI.
* Assist with testing the user interaction flow and making it as seamless and intuitive as possible.
* Ensure accessibility, help prompts, and proper CLI labeling.

#### **4. Nickitta Umuganwa Asimwe – Documentation & Quality Assurance Lead**

* Write comprehensive software documentation, including setup instructions, usage guides, and codebase overview.
* Create use case diagrams and contribute to UML visualization.
* Conduct rigorous testing of all functionalities and report bugs for resolution.
* Ensure that all documentation and diagrams are submitted according to required academic standards.