**TaskPad App**

**Project Overview:**

The goal of this project is to create a simple Todo application in C# that allows users to manage their tasks. The application should use files to store task data and LINQ for data manipulation. The user will be able to add, view, update, and delete tasks.

**Features:**

* Add a new task with a title and description.
* View all tasks.
* View a specific task by its ID.
* Mark a task as completed.
* Update task details (title, description, completion status).
* Delete a task by its ID.
* Save tasks to a file.
* Load tasks from the file.

**Requirements:**

*User Interface:*

* The application should be console-based, using text-based menus and input prompts.
* Implement a clear and user-friendly interface to guide users through different operations.

*Data Structure:*

* Create a class named "TaskItem" to represent a single task.
* The TaskItem class should have properties for Task ID, Title, Description, and Completion Status.

*Task Management:*

* Implement a class named "TodoManager" responsible for managing tasks.
* The TodoManager class should contain methods to Add, View, Update, and Delete tasks.
* Use LINQ queries to perform data manipulation operations on the task list.

*File Handling:*

* Implement a class named "FileHandler" to handle file operations.
* The FileHandler class should have methods to save and load tasks from a text file.
* The file should store task data in a structured format (e.g., JSON, CSV) for easy parsing.

**User Workflow:**

*Main Menu:*

Upon launching the application, display a main menu with the following options:

a. Add a task

b. View all tasks

c. View a specific task

d. Mark a task as completed

e. Update a task

f. Delete a task

g. Save tasks to a file

h. Load tasks from a file

i. Exit

*Add a Task:*

* Prompt the user to enter a title and description for the task.
* Generate a unique ID for the task.
* Add the task to the task list.

*View Tasks:*

* Display all tasks with their IDs, titles, descriptions, and completion status.

*View Specific Task:*

* Prompt the user to enter the ID of the task they want to view.
* Display the details of the specified task.

*Mark Task as Completed:*

* Prompt the user to enter the ID of the task they want to mark as completed.
* Update the task's completion status to "Completed."

*Update Task:*

* Prompt the user to enter the ID of the task they want to update.
* Display the task's current details and allow the user to modify the title, description, or completion status.

*Delete Task:*

* Prompt the user to enter the ID of the task they want to delete.
* Remove the task from the task list.

*Save and Load Tasks:*

* Provide options to save the current task list to a file and load tasks from a file.
* Handle file-related errors gracefully.

*Exit:*

* Terminate the application when the user chooses to exit.

**Error Handling:**

* Implement error handling for invalid user inputs and file-related errors.
* Ensure that the application does not crash due to exceptions.

**Project Features :**

* Allow users to set due dates for tasks.
* Implement sorting and filtering options for the task list based on different attributes.
* Add the ability to prioritize tasks.
* Create an interactive console interface with colors and formatting to improve user experience.
* Remember to break down the project into smaller tasks and modules for better development and organization.

**Testing:**

* Test the application thoroughly with various scenarios, including adding, updating, deleting tasks, saving/loading from files, and handling edge cases.
* Provide clear and concise code comments to explain the functionality of different methods and classes.