FEATURES IMPLEMENTED

1. Added functionality to view, add, complete, delete, tasks, search (search by description and task name (case sensitive)) and show statistics of completer against incompleted tasks (with percentage), showing completer, pending and total tasks.

2. Made tasks have properties: title, description, category, completion status, and creation date which takes the date from systems time and date.

3. Search Functionality:

4. used getCategoryColor function to assign specific colors to task categories for better visualization.

5. Tasks are saved to and loaded from data.json to ensure data persistence across sessions.

WHAT LEARNT ALONG THE WAY

1. Converted the project to TypeScript for better type safety and maintainability.

2. Defined interfaces and classes (Task and TaskData) to model task data.

3. Learned how to use TypeScript features like interfaces, classes, and type annotations to improve code quality.

4. Understanding how to handle type errors and ensure type safety in a Node.js application.

5. Gained experience working with the Node.js fs module to read, write, and manage files.

6. Learned to use libraries like chalk for terminal styling and inquirer for interactive prompts.

7. Implemented error handling for file operations and user input validation.

8. Improved skills in organizing code into reusable functions and modules for better readability and maintainability.

CHALLENGES I FACED

1. Encountered issues with type annotations, such as using typeof chalk instead of Chalk for type safety.

2. Resolved errors related to missing properties by updating the Task class and ensuring proper type definitions.

3. Faced challenges in converting plain JSON objects into Task instances. Solved this by implementing a fromObject static method in the Task class.

4. Dealt with issues between ES modules and CommonJS, like when i was importing libraries like chalk.