Vityarthi project
Todo List

Name:- Anmol Sharma
Reg. No.:- 25BCE10140

INTRODUCTION

A simple and beginner-friendly Command Line To-Do List written in Python.
This program allows users to add, edit, delete, and list tasks easily using a simple menu interface.

PROBLEM STATEMENT

People tend to forget the tasks they have at hand, so they need a visual list to keep it in their mind, but most apps are too complicated. They require accounts, internet access, or have too many confusing buttons.

FUNCTIONAL REQUIREMENTS

- **Task Management:** The system must allow users to manage a list of tasks.
- **Add Tasks:** Users must be able to input a new task description, which is then added to the list.
- **List Tasks:** The system must display all current tasks in an ordered, numbered list format. If no tasks exist, a message indicating an empty list should be displayed.
- **Edit Tasks:** Users must be able to select an existing task by its index number and replace its content with a new description.
- **Delete Tasks:** Users must be able to select an existing task by its index number and remove it from the list.
- **User Interface:** The system must provide a command-line interface (CLI) menu that clearly presents all available options (Add, Edit, Delete, List, Exit).
- **Exit Application:** Users must have a specific menu option to terminate the program gracefully.
- **Error Handling (User Input):** The system must handle invalid menu choices and non-numeric inputs for task indices, providing informative error messages to the user without crashing.

NON-FUNCTIONAL REQUIREMENTS

Usability:

- **The interface should be intuitive for a command-line environment, relying on simple numeric choices for navigation.**
- **Feedback messages should confirm successful actions (e.g., "Task added successfully") or explain errors (e.g., "Invalid choice. Please try again.").**

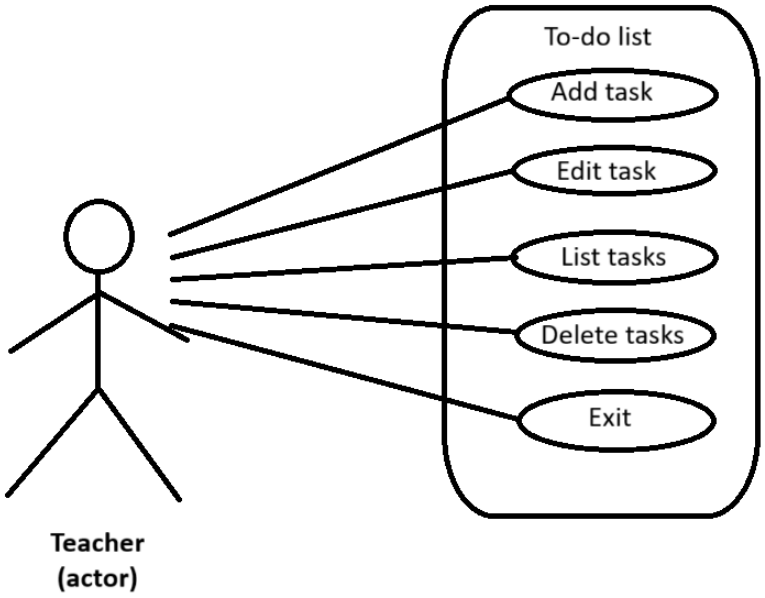
- **Reliability:**
  - **The application should be robust enough to handle common user input errors (like typing letters instead of numbers for menu options or indices) without halting execution.**
- **Performance:**
  - For a small, in-memory list application, performance should be immediate, with negligible delay in executing any command.
- **Maintainability:**

    - The code is structured into separate, focused functions (`add_task`, `edit_task`, etc.), which enhances readability and simplifies future modifications or debugging.

- **Scalability:**
    - The current application stores data entirely in a volatile in-memory list (`tasks = []`). When the program exits, all data is lost. This design is suitable for short-term use but not scalable for persistent data storage (e.g., it would not meet the non-functional requirement for *data persistence*).

DESIGN DIAGRAMS

1. Use Case Diagram:



IMPLEMENTATION

1. **Language:** Python3
2. **Libraries Used:** no libraries were used in this code

TESTING APPROACH

1. Enter many tasks by using ‘enter task’ command
2. Use ‘list task’ command to get the list of all the command you have entered before
3. Use ‘edit’ task command to make changes to preexisting tasks
4. Check if the tasks have been edited by using ‘list task’ command
5. Delete some by ‘delete task’ command
6. Again check if the tasks have been deleted by using ‘list task’ command
7. Now exit the terminal by using exit command

SCREENSHOTS

1. Code

todo.py > ...

```python
tasks = []

def menu():
    print("\n1. Add Task")
    print("2. Edit Task")
    print("3. Delete Task")
    print("4. List Tasks")
    print("5. Exit")

def add_task():
    task = input("Enter task: ")
    tasks.append(task)
    print("Task added successfully.")

def edit_task():
    if tasks:
        list_tasks()
        try:
            task_index = int(input("Enter task index to edit: ")) - 1
            if 0 <= task_index < len(tasks):
                new_task = input("Enter new task: ")
                tasks[task_index] = new_task
                print("Task edited successfully.")
            else:
                print("Invalid index.")
        except ValueError:
            print("Invalid input. Please enter a number.")
    else:
        print("No tasks to edit.")

def delete_task():
    if tasks:
        list_tasks()
        task_index = int(input("Enter task index to delete: ")) - 1
        if 0 <= task_index < len(tasks):
            del tasks[task_index]
```

```
            print("Task deleted successfully.")
        else:
            print("Invalid index.")
    else:
        print("No tasks to delete.")

def list_tasks():
    if tasks:
        print("\nYour Tasks")
        for index, task in enumerate(tasks):
            print(f"{index + 1}. {task}")
    else:
        print("No tasks in the list.")

while True:
    menu()
    choice = input("Select an option: ")

    if choice == '1':
        add_task()
    elif choice == '2':
        edit_task()
    elif choice == '3':
        delete_task()
    elif choice == '4':
        list_tasks()
    elif choice == '5':
        print("Exiting To-Do app.")
        break
    else:
        print("Invalid choice. Please try again.")
```

2. Result

```
PS C:\Users\HP\OneDrive\Desktop\project> py todo.py

1. Add Task
2. Edit Task
3. Delete Task
4. List Tasks
5. Exit
Select an option: 1
Enter task: study
Task added successfully.

1. Add Task
2. Edit Task
3. Delete Task
4. List Tasks
5. Exit
Select an option: 2

Your Tasks
1. study
Enter task index to edit: 1
Enter new task: study python
Task edited successfully.

1. Add Task
2. Edit Task
3. Delete Task
4. List Tasks
5. Exit
Select an option: 4

Your Tasks
1. study python
```

```
1. Add Task
2. Edit Task
3. Delete Task
4. List Tasks
5. Exit
Select an option: 3

Your Tasks
1. study python
Enter task index to delete: 1
Task deleted successfully.

1. Add Task
2. Edit Task
3. Delete Task
4. List Tasks
5. Exit
Select an option: 5
Exiting To-Do app.
PS C:\Users\HP\OneDrive\Desktop\project>
```

FUTURE ENHANCEMENTS

1. Save tasks to a file (JSON/text file)
2. Mark tasks as completed
3. Add deadlines or priority levels
4. Create a GUI version using Tkinter

REFERENCES

1. https://www.markdownguide.org/
2. https://git-scm.com/docs
3. https://docs.github.com/en