To-Do List Application

Sumi Akter

IT-23055

September 2024

Department Of Information & Communication Techology

Mawlana Bhasani Science & Technology University

Santosh,Tangail.

Project Title:

To-Do List Application

Project Executive Summery

This project aims to develop a simple yet functional **To-Do List Application** using the C++ programming language. The primary goal of this project is to help users manage their daily tasks efficiently by providing features like adding tasks, viewing tasks, marking tasks as completed, and deleting tasks. The application will store tasks persistently in a text file, enabling users to save and retrieve their tasks even after exiting the program.

The key objectives of this project are:

- **Task Management**: Enable users to manage their tasks easily by adding, viewing, marking them as done, and deleting them.
- **Persistent Storage**: Save tasks to a file so that users can retrieve their tasks when they reopen the program.
- **User-Friendly Interface**: Develop a command-line interface (CLI) that is intuitive and easy to use for managing tasks.

Anticipated Deliverables

The To-Do List Application will provide the following features:

- **Add a Task**: Users can input a new task with a description, which will be added to the list of tasks.
- **View All Tasks**: Display all tasks, including the ones marked as completed.
- **Mark a Task as Done**: Allow users to select a task and mark it as completed.
- **Delete a Task**: Allow users to delete tasks they no longer need.
- **Exit and Save**: Save all tasks to a file before exiting the application, ensuring that tasks are not lost.

This project is designed as a foundational tool that can be expanded in the future to include more advanced features, such as task priorities, due dates, categories, and a graphical user interface (GUI).

Target Audience

- **Students and Professionals**: Individuals looking to manage their daily tasks and to-do lists in a simple and efficient way.
- **Beginner Programmers**: Those learning C++ and looking for hands-on experience in building practical applications with file I/O and basic data structures.

Technologies Used:

- **Programming Language**: C++ (Standard Library features like file I/O, vectors, and strings)
- **Development Environment**: Any C++ compiler (e.g., GCC, Microsoft Visual C++, or an IDE like Code::Blocks or Visual Studio)
- **File System**: Text files will be used to store tasks persistently.

Functional Requirements:

1. **Add Task**: The user will be prompted to enter a description of the task, which will be added to the task list.
2. **View Tasks**: The application will display all tasks, indicating whether each task is completed.
3. **Mark Task as Done**: The user can mark a task as done by entering the task number.
4. **Delete Task**: The user can delete a task by specifying its task number.
5. **File Persistence**: Tasks will be stored in a file (`tasks.txt`) to maintain data after the application is closed.

Non-Functional Requirements:

1. **Usability**: The application should be simple, intuitive, and easy to navigate via a text-based interface.
2. **Efficiency**: The application should handle adding, marking, and deleting tasks with minimal delay.
3. **Portability**: The code should be portable and run on any system with a C++ compiler.
4. **Maintainability**: The code will be modular and well-commented to facilitate future maintenance and potential expansion of features.

Implementation Plan:

The project will be developed in the following stages:

1. **Phase 1: Project Setup**
    - Set up a basic C++ project.
    - Define the data structure (task object with description and status).
2. **Phase 2: Basic Features**
    - Implement functions to add tasks and display them.
    - Implement marking tasks as completed.
3. **Phase 3: File I/O**
    - Implement saving and loading tasks from a text file to ensure data persistence.
4. **Phase 4: Task Deletion**
    - Add functionality to delete tasks from the list.
5. **Phase 5: Testing and Optimization**
    - Test the application for bugs and optimize code performance.
    - Ensure that file I/O is robust and handles errors effectively.
6. **Phase 6: Documentation and Deployment**
    - Document the code and instructions for use.

- Prepare the final version for deployment and demonstration.

Timeline:

- **Week 1**: Requirement gathering and project setup.
- **Week 2**: Implementation of core features (adding, viewing, marking tasks as done).
- **Week 3**: Implementation of file I/O for task storage and retrieval.
- **Week 4**: Implementation of task deletion and final testing.
- **Week 5**: Code review, optimization, and documentation.

Budget:

As this is a small-scale project, the development costs will be limited to:

- **Development tools**: Free (using open-source tools like GCC or Code::Blocks)
- **Time Investment**: The project can be completed with around 20-30 hours of development work, depending on the complexity of features added.

Risks and Challenges:

- **File Corruption**: There is a risk that tasks may not be saved correctly due to improper file handling. This will be mitigated by using proper file I/O techniques.
- **Task Management Limitations**: This is a simple command-line application, so scalability and usability may be limited compared to GUI-based applications. However, the project is designed with potential for future expansion.

Conclusion:

The To-Do List Application in C++ is a practical project aimed at helping users manage their tasks effectively. It serves as an excellent demonstration of C++ programming fundamentals, such as file handling, dynamic data structures, and user interaction. While simple in its initial scope, this project has the potential to grow into a more feature-rich and widely used tool.

The successful completion of this project will provide a valuable learning experience and a useful tool for day-to-day task management.

This proposal outlines the plan, scope, and implementation of a simple To-Do List application in C++. The project can be extended with advanced features, such as deadlines and task sorting, as time and needs evolve.