



Computer Aptitude Minor Project

TO DO LIST

**Submitted by:**

Name: Sonam Namgyel

UID: 23BCA10413

Section: 23BCA - 4”A”

Name: Pema Dorji

UID: 23BCA10400

Section: 23BCA - 4”A”

**Submitted to:**

Mrs. Shuchi Sherma (E13768)

Computing Aptitude

# **To-Do List Project Report**

## **1. Introduction**

In the fast-paced digital era, organizing daily tasks efficiently has become an essential part of both personal and professional life. The To-Do List application is designed to provide users with a simple, effective, and user-friendly system to manage and track tasks. Developed using C++, this console-based program enables users to create, view, edit, delete, and mark tasks as completed with ease. By utilizing object-oriented programming principles, it ensures code molecularity, re-usability, and maintainability. This project demonstrates how even a console-based application can deliver powerful task management capabilities by leveraging clean design, intuitive interaction, and robust data handling.

The To-Do List application aims to help users stay productive and organized by providing an intuitive platform for managing daily activities. With clear options and minimal complexity, this project emphasizes efficient data manipulation through the use of C++ Standard Template Library (STL) components like vectors and strings, while adhering to best software development practices.

## **2. Objectives**

* Develop a console-based To-Do List application using C++.
* Implement object-oriented principles to manage task creation, modification, and deletion.
* Provide a user-friendly interface with menu-driven operations.
* Allow users to mark tasks as completed or pending.
* Ensure efficient data handling through STL vectors and functions.

## **3. System Requirements**

### **3.1 Hardware Requirements**

* Processor: Intel Core i3 or higher
* RAM: Minimum 2 GB
* Storage: Minimum 500 MB of free disk space

### **3.2 Software Requirements**

* Programming Language: C++ (C++11 or above)
* Compiler: GCC / MinGW / Code::Blocks / Visual Studio
* Operating System: Windows / Linux / macOS
* Text Editor: Any IDE supporting C++ (e.g., Code::Blocks, VS Code)

## **4. System Design**

### **4.1 Class Design**

The system is designed using two main classes: Task and ToDoList.  
  
• **Task Class:** Responsible for defining individual tasks with attributes such as name, description, due date, and completion status.  
**• ToDoList Class:** Manages a collection of Task objects and handles all user interactions including adding, deleting, editing, and displaying tasks.

### **4.2 System Architecture**

The application follows an object-oriented architecture with the following structure:  
  
• **Input Layer:** Handles user input through console prompts.  
• **Logic Layer:** Processes user commands and performs CRUD (Create, Read, Update, Delete) operations.  
• **Output Layer:** Displays results and task information on the console.

## **5. Implementation**

### **5.1 Modules Description**

1. **Add Task:** Allows the user to create a new task by providing task name, description, and due date.
2. **Delete Task:** Removes a selected task from the list.
3. **Display Tasks:** Displays all tasks with details including status and due date.
4. **Mark Task Completed:** Marks a chosen task as completed, changing its status accordingly.
5. **Edit Task:** Allows the user to modify existing task details.

### **5.2 Key Code Features**

1. **Use of Classes and Objects:** Implements encapsulation and modular design.
2. **Vector-based Storage:** Uses std::vector to store dynamic lists of tasks efficiently.
3. **Menu-driven Interface:** Provides a clear, easy-to-navigate console interface.
4. **String Handling:** Uses getline() for multi-word inputs such as task descriptions.
5. **Status Tracking:** Maintains task completion state using boolean flags.

## **6. Testing**

1. **Add Task:** Verify that a new task can be added successfully with all details.
2. **Delete Task:** Ensure that the correct task is deleted when chosen by index.
3. **Display Tasks:** Confirm that all existing tasks are displayed with accurate details.
4. **Mark Task Completed:** Check that selected tasks update their completion status properly.
5. **Edit Task:** Validate that existing task information can be edited and updated correctly.

## **7. Challenges & Solutions**

* **Input Handling Issues:** Frequent use of cin and getline caused input buffering issues. Solution: Implemented cin.ignore() appropriately to clear the input buffer.
* **Dynamic Memory Management:** Ensured no manual memory allocation to avoid leaks by relying on STL vector for dynamic storage.
* **Menu Loop Stability:** Used do-while loops and switch statements for stable and continuous user interaction.

## **8. Conclusion & Future Enhancements**

The To-Do List project successfully demonstrates the implementation of a console-based task management system using C++. It showcases the power of object-oriented programming principles to structure and manage user data effectively. The modular design ensures ease of maintenance and scalability while providing essential functionalities like adding, deleting, editing, and completing tasks.

Future enhancements may include the following improvements:

1. Implementing file handling to save and load tasks between sessions.
2. Adding priority levels and task sorting features.
3. Introducing a graphical user interface (GUI) using Qt or wxWidgets.
4. Integrating date/time libraries for better due date validation.
5. Adding categories and search functionality for efficient task management.

**9. Project Image**

