ABBOTTABAD UNIVERSITY OF

SCIENCE AND TECHNOLOGY:

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

NAME: IZHARULHAQ KHAN:

ROLL NO: 14967:

PROGRAM: BSSE:

SEMESTER: 4{B}:

SUBJECT: SOFTWARE CONSTRUCTUON AND DEVELOPMENT:

SESSION: SPRING 2025:

SUBMITTED TO: MS SAMAN SHAHEEN:

SEMESTER ASSIGNMENT:

A

Software Requirements Specification (SRS):

1: Introduction:

Purpose:

The purpose of this document is to define the software requirements for a console-based Task Manager system. This system will allow users to create, read, update, delete, and filter tasks efficiently. It is intended for educational use in demonstrating core principles of software construction and development using C++.

Scope:

This Task Manager is a command-line application designed for individual use. The system stores tasks in a persistent format (`tasks.json`) and offers features like status updates, date validation, and task filtering. It demonstrates modular design, clean coding practices, and testing in C++.

Intended Audience:

- **Developers**: To understand the system design and contribute effectively.
- **Testers**: To evaluate system correctness and robustness.
- **Students/Instructors**: For learning software construction techniques.

Definitions and Abbreviations:

- **CRUD**: Create, Read, Update, Delete
- **CLI**: Command-Line Interface
- **JSON**: JavaScript Object Notation
- **Task**: A unit of work defined by the user

2:Overall Description:

Product Perspective:

The application is self-contained and designed to run in a terminal/console environment. It has no external service dependencies and operates on local file storage.

A

Product Functions:

- Add new tasks with title, description, and due date
- View all tasks with details
- Update existing tasks
- Delete tasks
- Mark tasks as completed/incomplete
- View tasks by completion status
- Save/load tasks persistently using JSON

User Classes and Characteristics:

User Class:	**Description:**
End User	Uses the application to manage tasks
Developer	Modifies or enhances system functionality
Tester	Validates application features and stability

Operating Environment:

- OS: Windows, Linux, or macOS
- Compiler: `g++`, `clang++`
- Language: C++17 or later

Constraints:

- No GUI (CLI-based only)
- Persistent storage limited to JSON files
- Must be compiled using standard C++ compiler

3:Functional Requirements:

Add Task:

- Prompt user for task title, description, and due date
- Assign unique task ID
- Save task to persistent storage

A

View Tasks:

- Display all tasks with details
- Display only completed or incomplete tasks

Update Task:

- Select task by ID
- Modify title or description
- Save updated data

Delete Task:

- Select task by ID
- Remove task from storage

Mark Task Status:

- Select task by ID
- Toggle completion status

Load/Save Tasks:

- Load tasks from `tasks.json` at startup
- Save tasks after every update or exit

4:Non-Functional Requirements:

Performance:

- Application should respond within 1 second for all operations

Usability:

- Clear prompts and feedback for user actions
- Menu-based navigation

A

Maintainability:

- Modular design with separate `.cpp` files for each feature
- Clear code comments and documentation

Reliability:

- Handle invalid input gracefully
- Prevent crashes due to bad data

5:System Design Overview:

Modular Components:

- `main.cpp` – Entry point and menu logic

- `task.cpp` – Task structure and logic
- `file_manager.cpp` – File operations
- `date_util.cpp` – Date validation
- `search.cpp` – Search by ID
- `storage.cpp` – JSON-based persistence
- `filter.cpp` – Task filtering

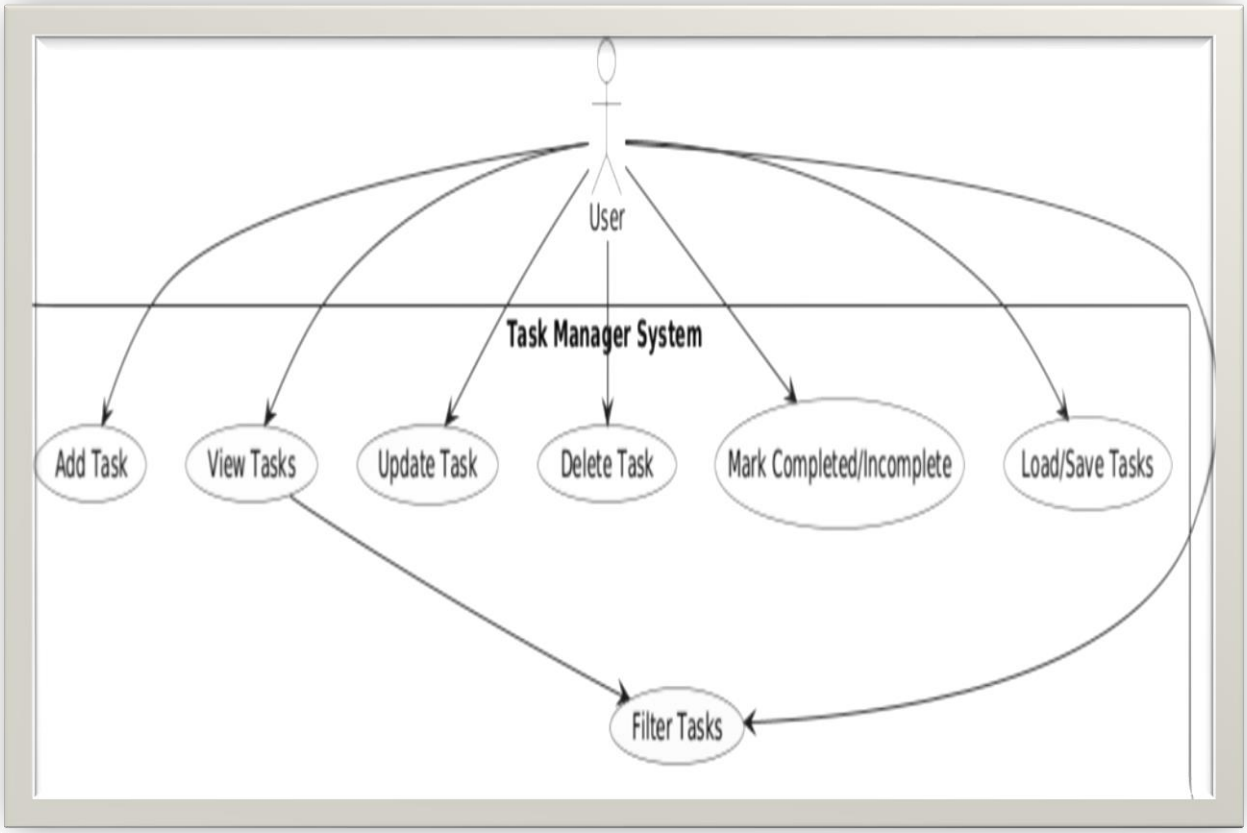
6:Appendices:

JSON Task Format

```
{
  "id": 1,
  "title": "Buy groceries",
  "description": "Milk, eggs, bread",
  "completed": false,
  "creationDate": "2025-05-24",
  "dueDate": "2025-05-26"
}
```

A

USECASE Diagram:



A

CLASS DIAGRAM: