The University of Faisalabad

Data Structures

Course Code: CS-216

Software Requirement Specification(SRS)

Submitted by:

Tayyab Imran	2023-BS-AI-019
Abdul Haseeb Arif	2023-BS-AI-033
Muhammad Zain	2023-BS-AI-052

Submitted to:

Ms. Irsha Qureshi

Department:

Computer Science

Table of Contents

6. **Other Non-functional Requirements**

- Performance Requirements
- Security Requirements
- Maintainability
- Portability

Software Requirement Specification for To-Do List Manager

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed specification for the "To-Do List Manager" application. The application enables users to efficiently manage their tasks, including adding, editing, viewing, searching, deleting, and marking tasks as done.

1.2 Scope

This application is a console-based task management system designed for individual users to manage personal or work-related tasks. It includes features for storing, retrieving, and updating tasks. Tasks are stored persistently in text files (todo.txt and done.txt) for future access.

1.3 Definitions, Acronyms, and Abbreviations

- **Task**: A unit of work consisting of a title and description.
- **ID**: A unique identifier assigned to each task.
- **Done Task**: A task that has been completed and marked accordingly.

1.4 References

- C++ Programming Language Documentation
- File I/O in C++ Reference
- Console-based UI Standards

1.5 Overview

This document outlines the functional and non-functional requirements of the To-Do List Manager application, detailing system features, interfaces, and constraints.

2. Overall Description

2.1 Product Perspective

The To-Do List Manager is an independent application that runs on any system supporting C++ and provides users with a minimalistic interface to manage tasks. It uses text files for data persistence, ensuring simplicity and accessibility.

2.2 Product Functions

The main functions of the application are:

1. Add new tasks with a title and description.
2. Edit existing tasks by modifying the title and/or description.

3. View all tasks, displaying their status (Pending or Done).
4. Search for tasks by their unique ID.
5. Delete tasks.
6. Mark tasks as done.

2.3 User Characteristics

- The primary users are individuals with basic computer literacy.
- Users should be familiar with console-based applications and file management.

2.4 Constraints

- The application runs in a console/terminal environment.
- Tasks are stored in plain text files (todo.txt and done.txt), limiting scalability.
- The system does not support concurrent access by multiple users.

2.5 Assumptions and Dependencies

- The user has read/write access to the directory containing the application files.
- The application is run on a system with a C++ runtime environment.

3. Specific Requirements

3.1 Functional Requirements

1. **Task Management**:
    - Users can add tasks with a title and description.
    - Users can edit tasks using their unique ID.
    - Users can view all tasks and their statuses (Pending/Done).
    - Users can search for a task by its ID.
    - Users can delete tasks by confirming the operation.
    - Users can mark tasks as done.
2. **File Handling**:
    - Tasks must be stored in todo.txt.
    - Done task IDs must be stored in done.txt.
    - The application should load tasks and statuses from the files at startup.
    - The application should save tasks and statuses back to the files after any update.

3. **Error Handling**:
    - Display meaningful error messages for invalid inputs or file handling issues.
    - Prevent actions on non-existent tasks.

3.2 Non-functional Requirements

1. **Usability**:
    - The interface must be simple and intuitive.
    - Provide clear instructions and error messages.
2. **Performance**:
    - Load and save tasks efficiently, even for large files.
3. **Reliability**:
    - Ensure data integrity during file operations.
4. **Portability**:
    - Must run on any platform with a C++ compiler and runtime environment.

4. System Features

Feature 1: Add Task

Description: Allows the user to add a new task with a title and description.

- Input: Task title and description.
- Output: Task added to todo.txt.
- Constraints: Title and description cannot be empty.

Feature 2: Edit Task

Description: Enables editing of task title and/or description using the task ID.

- Input: Task ID, new title, and/or description.
- Output: Task updated in todo.txt.
- Constraints: Task ID must exist.

Feature 3: View All Tasks

Description: Displays all tasks and their statuses (Pending/Done).

- Input: None.
- Output: List of tasks with details.

Feature 4: Search Task

Description: Search for a task by its ID.

- Input: Task ID.
- Output: Task details if found, error message otherwise.

Feature 5: Delete Task

Description: Deletes a task based on its ID after user confirmation.

- Input: Task ID.
- Output: Task removed from todo.txt and done.txt (if applicable).
- Constraints: Task ID must exist.

Feature 6: Mark Task as Done

Description: Marks a task as completed.

- Input: Task ID.
- Output: Task ID added to done.txt.
- Constraints: Task ID must exist.

5. External Interface Requirements

5.1 User Interface

- Console-based interaction with menus and prompts.

5.2 Hardware Interface

- Requires a keyboard for input.

5.3 Software Interface

- Operates on systems with C++ runtime support.

5.4 Communication Interfaces

- No external communication interfaces required.

6. Other Non-functional Requirements

6.1 Performance Requirements

- Task load and save operations should execute in under 1 second for files with up to 1,000 tasks.

6.2 Security Requirements

- File operations should ensure no data corruption.

- Prevent unauthorized access to todo.txt and done.txt.

6.3 Maintainability

- Code should be modular and easy to extend.

6.4 Portability

- Should compile and run on any standard C++ compiler.