## PROGRAMMING FUNDAMENTALS FINAL PROJECT DOCUMENTATION

## Project Title: Inventory Management System

Student Name(s): **Faizan Mughari, Kabeer, Osama Athar**

Semester/Section: **First Semester (Computer Science)**

University: **Salim Habib University**

Submission Date: **28/January/2025**

# Introduction

## Project Overview

This Inventory Management System is a C++ console-based application that allows users to:

• Add new products to inventory

• View the list of available products

• Edit product details

• Save and load inventory data from a file

## Purpose

The project is designed to simplify inventory management and introduce fundamental programming concepts such as file handling, data structures, and user input processing.

# Objectives

• To understand the use of data structures like vector for dynamic storage.

• To practice file handling in C++ for data persistence.

• To implement basic CRUD (Create, Read, Update, Delete) operations on inventory data.

• To improve user interaction using console-based menus.

# Tools and Technologies

• Programming Language: C++

• IDE Used:, Dev-C++

• File Handling: fstream for reading/writing data

# Project Design

The inventory system follows this flow:

1. User selects an option (Add, View, Edit, Exit).

2. If adding, the product details are collected and stored in memory.

3. If viewing, the inventory is displayed in a structured format.

4. If editing, the user searches by Product ID, and updates details.

5. Upon exit, data is saved to a file (inventory.txt).

6. When restarted, the program loads inventory data from the file.

# Implementation

# Main Functionalities

1. Adding a Product (addProduct())

• User enters Product ID, Name, Quantity, and Price.

• The product is added to a vector storage (inventory).

2. Viewing Products (showInventory())

• Displays all products in a formatted table.

• If the inventory is empty, it notifies the user.

3. Editing a Product (editProduct())

• User enters Product ID to locate the item.

• If found, the name, quantity, or price can be updated.

4. Saving to File (saveToFile())

• Saves all inventory data to inventory.txt in a comma-separated format.

5. Loading from File (loadFromFile())

• Reads inventory.txt, parses the data, and restores the inventory.

6. Data Parsing (parseInt() and parseDouble())

• Converts string inputs to integers or floating-point numbers for processing.

# Testing

|  |  |  |
| --- | --- | --- |
| Input | Expected Output | Actual Output |
| Add Product (ID: 101, Name: Apple, Qty: 50, Price: 1.5) | Product added successfully | Product added successfully |
| View Inventory | Displays inventory table | Displays inventory table |
| Edit Product (ID: 101, Change Qty to 60) | Product updated successfully | Product updated successfully |
| Save & Load Inventory | Data is saved and loaded correctly | Data is saved and loaded correctly |

# Results

• The program successfully manages inventory data with CRUD operations.

• It persists data across sessions using file handling.

• Users can efficiently add, view, and modify products.

# Conclusion

This project helped in:

• Understanding file handling in C++.

• Implementing dynamic data structures like vector.

• Practicing user input handling and data validation.

• Learning the importance of debugging and error handling in programs.