

**Content Page**

1. **Problem**
2. **Methodologies Used**
3. **Code**
4. **Results**
5. **References**

**Problem**

In the competitive landscape of small businesses, effective management of sales transactions is crucial for maintaining customer satisfaction and ensuring accurate financial records. Traditional methods of invoice management, such as handwritten invoices or basic spreadsheets, can lead to several issues, including:

* **Human Error**: Manual entry of item prices and quantities can result in mistakes, leading to incorrect billing and customer dissatisfaction.
* **Time Consumption**: Generating invoices manually can be time-consuming, diverting valuable time away from other essential business operations.
* **Lack of Validation**: Without a systematic approach, there is no mechanism to ensure that the entered data is valid, which can lead to financial discrepancies.

To address these challenges, we developed a **Simple Invoice Generator** that allows small business owners to create invoices efficiently and accurately. The system enables users to input their name, the items they wish to sell, and their respective quantities and prices. Importantly, the program includes input validation to ensure that all data entered is correct, thus enhancing the overall user experience and maintaining data integrity.

The primary objective of this project is to develop a simple console-based Invoice Management System. The system must handle invalid inputs gracefully, prompting users to re-enter data without terminating the program. This will enhance user experience and ensure data integrity.

We had a problem for the customer item list we used Vectors items.in the for loop After the item details are collected ”**c.items.push\_back(temp)”** adds that item to the customer's list of items, allowing the program to keep track of all items the customer is purchasing.

**Methodologies Used**

To achieve the objectives of this project, the following methodologies were employed:

1. **Requirements Gathering**:
   * Conducted discussions to identify the need for a user-friendly interface that allows for easy input of item details.
   * Determined the necessity for input validation to ensure that users cannot proceed with invalid data.
2. **Design**:
   * Designed a simple console-based application using C++.
   * Structured the program using **struct** to represent items and customers, facilitating organized data management and easy access to item attributes.
3. **Implementation**:
   * Developed the program using the C++ programming language, focusing on input validation and user interaction.
   * Utilized loops and conditionals to ensure that the program handles invalid inputs effectively, allowing users to correct their mistakes without losing progress.
4. **Testing**:
   * Conducted various tests to ensure that the program correctly calculates totals and handles invalid inputs as expected.
   * Tested edge cases, such as entering zero or negative values for prices and quantities, to confirm that the program prompts for re-entry.

**Code**

#include <iostream>

#include <iomanip>

#include <vector>

using namespace std;

struct item {

string nam;

int quan{};

int price{};

};

struct customer {

string name;

vector<item> items;

};

int main() {

int a;

customer c;

cout << "Enter your name: ";

cin >> c.name;

cout << "Enter Number of items: ";

cin >> a;

for (int i = 0; i < a; i++) {

item temp;

cout << "Enter Name of item= ";

cin >> temp.nam;

cout << "Enter Quantity of Items= ";

cin >> temp.quan;

cout << "Enter Price of Item= ";

cin >> temp.price;

if (temp.quan <= 0) {

cout << "Error! Quantity must be greater than zero." << endl;

return 1;

}

do {

cout << "Enter Price of Item= ";

cin >> temp.price;

if (temp.price <= 0) {

cout << "Error! Price must be greater than zero. Please re-enter." << endl;

}

} while (temp.price <= 0);

c.items.push\_back(temp);

}

cout << "\nInvoice for " << c.name << ":\n";

cout << left << setw(20) << "Item Name"

<< setw(10) << "Quantity"

<< setw(10) << "Price"

<< setw(10) << "Total" << endl;

int grandTotal = 0;

for (int i = 0; i < c.items.size(); i++)

{

int total = c.items[i].quan \* c.items[i].price;

grandTotal = grandTotal + total;

cout << left << setw(20) << c.items[i].nam

<< setw(10) << c.items[i].quan

<< setw(10) << c.items[i].price

<< setw(10) << total << endl;

}

cout << "--------------------------------------\n";

cout << "Grand Total: " << grandTotal << endl;

return 0;

}

**Results**

The Invoice Management System successfully allows users to input their name, the number of items, and the details of each item, including quantity and price. The program validates the inputs, ensuring that quantities and prices are greater than zero. If invalid data is entered, the program prompts the user to re-enter the information without exiting the program.

**References**

* Online C++ documentation and tutorials
* User feedback and testing results