\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Departmental Store Management System

# Group 10

# Nicolas Wiesner, Wayne Seide

# Programming 1 COP 2006

# Deepa Devasenapathy

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Contents:

1. Summary
2. Goals/What it is going to be able to do
3. Software Requirements
4. Introduction
5. Why you chose this project?
6. Existing technique and your improvisations to the existing concepts
7. Project Concept and procedure in detailed steps (flow diagrams, steps)
8. Reflection

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Summary**

We are making a program that will be able to handle the inventory of a department store and the various functions that it will need to have to function properly. To make the product inventory readily available to the User.

**Goals**

What will the program be able to do:

* Enter products and display that data to the user
* Editing products either at an existing location or adding that data
* Removing that product from the data entirely

**Software Requirements:**

**Operating System:**

* Compatible with Windows, macOS, or Linux.

**Memory (RAM):**

* Minimum: 512 MB.
* Recommended: 2 GB or more for smoother operation with IDEs.

**Storage:**

* A few MBs of free space for compiler and IDE installation.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Introduction

* We wanted to learn how to create a real-world application that people or businesses could use to store and manage data.
* Even if it's not exactly storing inventory of a department store, we can use what we learn from this project and apply it to other applications.

Why we Choose this project?

* We wanted to learn how to create a real-world application that people or businesses could use to accomplish something.
* Being able to manage updates and information is an important part that most companies are in need of, from accounting to store management, like in this project.
* Being able to apply what we learned in class to a real-world problem was an important part in us choosing this application.

Existing Techniques and Improvements

* We created a project that is able to manage data the way we accomplished this by using vectors, we also made it so we can update the information of specific objects and change that items quantity to see the changes at the beginning of the day to the end of the day.

Conclusion

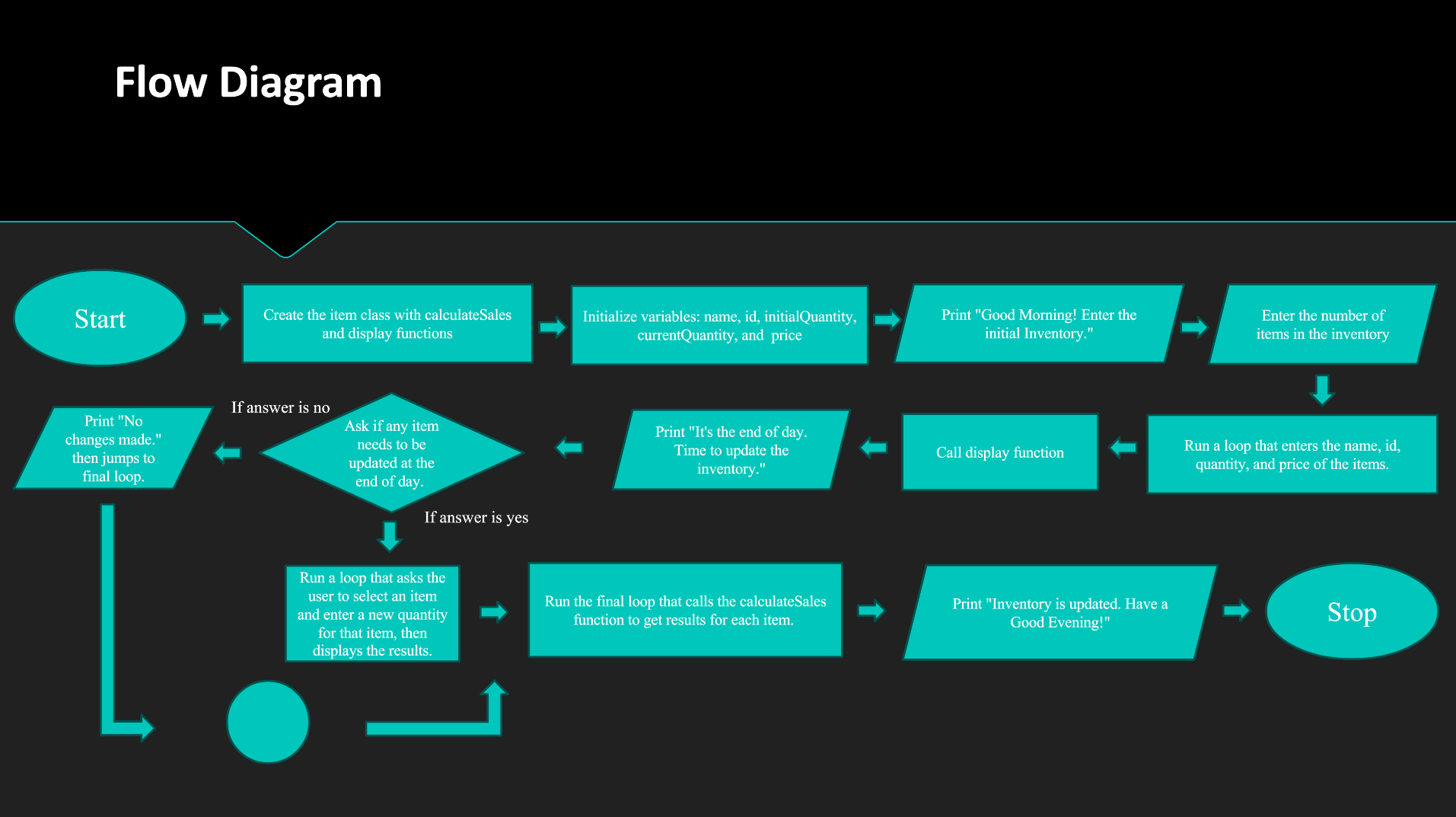
The project is a good example of what we could accomplish but I feel like there are some improvements we could make to simulate a department store more:

* Being able to save the data In a file
* Being able to add more inventory
* Changing the price of an item
* Getting rid of a items
* Exception handling

GitHub

[TacticalFoolery/DepartmentStoreManagement](https://github.com/TacticalFoolery/DepartmentStoreManagement)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** [Wayne-N-S/Inventory\_Management\_System](https://github.com/Wayne-N-S/Inventory_Management_System)



Code:

#include <iostream>  
#include <vector>  
#include <string>  
using namespace std;  
  
class Item {  
public:  
 string name;  
 int id, initialQuantity, currentQuantity;  
 double price;  
  
 // Constructor  
 Item(int id, string name, double price, int initialQuantity) : id(id), name(name), price(price), initialQuantity(initialQuantity), currentQuantity(initialQuantity) {}  
  
 // Display method  
 void display() const {

cout << "ID: " << id << ", Name: " << name << ", Price: $" << price << ", Initial Quantity: " << initialQuantity << ", Current Quantity: " << currentQuantity << endl;  
 }  
  
 // Method to calculate sales for the item  
 double calculateSales() const {  
 return price \* (initialQuantity - currentQuantity);  
 }  
};  
  
int main() {  
 vector<Item> items;  
 int numItems;  
 bool edit = true;  
 string decision;  
  
 // Simulate the beginning of the day  
 cout << "Good morning! Let's get started with entering the initial inventory for the day.\n";  
 cout << "How many items do you want to enter? ";  
 cin >> numItems;  
 cin.ignore(); // To ignore the newline character left by cin  
  
 for (int i = 0; i < numItems; ++i) {  
 int id, quantity;  
 string name;  
 double price;  
  
 cout << "Enter details for item " << i + 1 << ":\n";  
 cout << "ID: ";  
 cin >> id;  
 cin.ignore();  
 cout << "Name: ";  
 getline(cin, name);  
 cout << "Price: $";  
 cin >> price;  
 cout << "Quantity: ";  
 cin >> quantity;  
  
 items.emplace\_back(id, name, price, quantity);  
 }  
  
 // Display all items  
 cout << "\nItems entered:\n";  
 for (const auto& item : items) {  
 item.display();  
 }  
  
 // Simulate the end of the day  
 cout << "\nIt's the end of the day. Let's update the inventory based on today's sales.\n";  
 while (edit) {  
 cout << "Would you like to edit the quantity of an item? Y/N: ";  
 cin >> decision;  
 edit = (decision == "Y" || decision == "y");  
  
 if (edit) {  
 int id, quantity;  
  
 cout << "Enter the ID of the item you want to edit: ";  
 cin >> id;  
 cout << "Enter new quantity: ";  
 cin >> quantity;  
 cout << endl;  
  
 // If ID already exists, update the current quantity  
 if (id - 1 >= 0 && id - 1 < items.size()) {  
 items.at(id - 1).currentQuantity = quantity;  
 } else {  
 cout << "Invalid ID. No changes made." << endl;  
 }  
  
 // Display updated items  
 cout << "\nUpdated items:\n";  
 for (const auto& item : items) {  
 item.display();  
 }  
 }  
 }  
  
 // Display sales for each item at the end  
 cout << "\nSales for each item at the end of the day:\n";  
 for (const auto& item : items) {  
 cout << "Sales for ID " << item.id << " (" << item.name << "): $" << item.calculateSales() << endl;  
 }  
  
 cout << "\nThank you for updating the inventory. Have a great evening!\n";  
  
 return 0;  
}

**Reflection:**

**List of future improvements:**

1. Being able to save the data In a file
2. Being able to add more inventory
3. Changing the price of an item
4. Getting rid of a items
5. Exception handling
6. Creating a function to handle taxes
7. Creating a class for items in separate categories
8. Creating a class to track profit margins for each category
9. Adding a total sales function