Contents

[Development Document 2](#_Toc212722510)

[1) Specification 2](#_Toc212722511)

[2) System Flowchart 2](#_Toc212722513)

[3) Technical Description & Walkthrough 2](#_Toc212722517)

[Program Structure 2](#_Toc212722518)

[Decision Making 3](#_Toc212722519)

[Output 3](#_Toc212722520)

[Learning Value 3](#_Toc212722522)

[4) Critical Reflection 3](#_Toc212722523)

[Appendix (Code) 3](#_Toc212722524)

# 

# Development Document

## 1) Specification

## I was told to use C++ to make a very easy Vending Machine program. The program shows a short list of things with prices and codes (A1, A2, A3). The items are Cola, Water, and Crisps. The person types in how much money they have and then uses the item's code to choose it. The program checks to see if the user has enough money, gives the item if it's available, and shows the user how much change they got back. It also shows messages when the code is incorrect or there aren't enough funds. Variables, input/output, and if statements are the only very basic C++ code used in this version.

## 2) System Flowchart

## A simple diagram shows how the vending machine works below:

## It goes like this: Start > Show Menu > Enter Money > Enter Code > Check Code > If Invalid > End > If Valid > Check Money > If Not Enough > End > If Enough > Dispense Item > Show Change > End.

## The program asks for money and an item code, checks to see if the code and payment are correct, and either gives the item and change or shows an error message.

## 3) Technical Description & Walkthrough

This machine is very easy to use because it was made with C++. The main() method only has one line of code. It doesn't have any classes, loops, or groups. It is very clear how input, conditional thought, and output style work after reading it.   
At first, the program shows a list of items along with their rates and UPC codes. After that, the person types in how much money they want to add and the product code that they want to use. First, the code is checked to make sure it's right. Then, it compares the amount of money you put in with the price of the thing. It either gives the thing to the person who paid for it or prints a message saying that more money is needed. This easy flow shows how to use choices and information from the user to make a real-world process work in code. It also shows that a program with a clear structure and the right amount of spacing is easier to read and understand.

### Program Structure

1. The program starts by declaring a double variable ‘money’ to store the user’s balance and a string variable ‘code’ for the selected item.

2. It displays three menu items using cout: Cola (£1.50), Water (£1.00), and Crisps (£1.20), each with a code (A1, A2, A3).

3. The user is prompted to insert money and enter an item code. The program reads these using cin.

### Decision Making

A set of if and else if lines check to see what code the user entered after reading the input. Each code is linked to a thing and its cost. The program prints an error message and ends if the code is not acceptable.

Then, another if statement checks to see if the money entered is sufficient to purchase the chosen thing. The person is told they need more money if not. Taking the price away from the amount entered is how the computer figures out the change if the answer is yes.

### Output

### The last thing the program does is send messages that tell you to check what it is sending out and change it. Before the show stops, there is a thank-you message. It seems more real, like a real vending machine, and it's easy for them to pay. It also shows how important it is to learn how to code well and keep track of what you've written down.

### Learning Value

This project shows basic programming skills like reading data, using conditions, doing math, and printing formatted text. It also shows how easy logic can be used to show how something in the real world works, like a vending machine. Because it's so easy, it's good for people who have never played before.

## 4) Critical Reflection

This vending machine works best because it is easy to understand. People who are just starting to learn C++ will find it easy to use because it only has input/output, variables, and conditional statements. The show is short and simple to read. You can see the menu, pay, pick out an item, and get change. It does everything that needs to be done. It's very clear how to use if sentences to decide what to do.

It's easier to use, though, which makes it harder to do. It doesn't keep track of stock, makes it easy to store things, or has loops or groups for making multiple purchases. In a more complex form, arrays, loops, and functions could be used to add more things and cut down on repetition. Another thing it could do better is deal with bad data, like money records that aren't numbers. As I added these methods one by one to make the project better, the code would stay simple. If I knew more about functions, loops, and simple file I/O, I could make my next vending machine software more like the real thing.

## Appendix (Code)

// Super Simple Vending Machine  
#include <iostream>  
using namespace std;  
  
int main() {  
 double money = 0.0;  
 string code;  
  
 cout << "Welcome to the Simple Vending Machine!\n";  
 cout << "Menu:\n";  
 cout << "A1 - Cola (£1.50)\n";  
 cout << "A2 - Water (£1.00)\n";  
 cout << "A3 - Crisps (£1.20)\n";  
  
 cout << "\nInsert money (£): ";  
 cin >> money;  
  
 cout << "Enter item code (A1, A2, A3): ";  
 cin >> code;  
  
 double price = 0.0;  
 string item;  
  
 if (code == "A1") { item = "Cola"; price = 1.50; }  
 else if (code == "A2") { item = "Water"; price = 1.00; }  
 else if (code == "A3") { item = "Crisps"; price = 1.20; }  
 else {  
 cout << "Invalid code.\n";  
 return 0;  
 }  
  
 if (money < price) {  
 cout << "Not enough money. Please add more.\n";  
 } else {  
 double change = money - price;  
 cout << "Dispensing " << item << "...\n";  
 cout << "Your change is £" << change << "\n";  
 cout << "Thank you!\n";  
 }  
  
 return 0;  
}