

SUPER MARKET BILLING SYSTEM

A project by:

Lavanya Sri Chava – 2312069

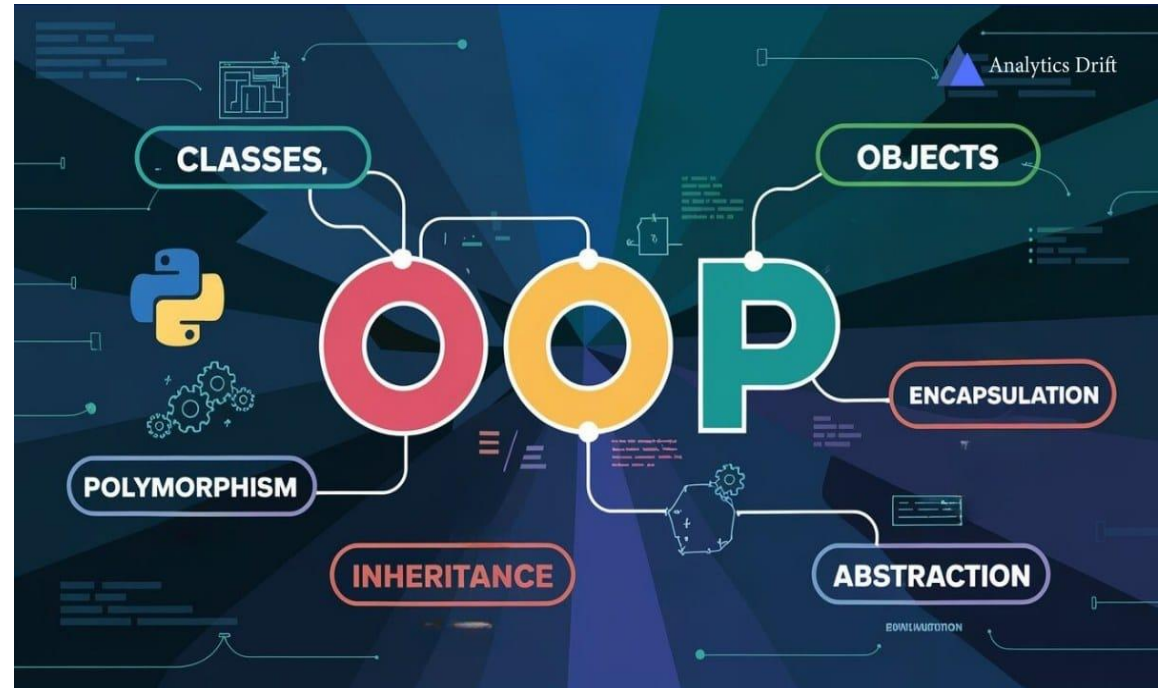
Divya kumari- 2312077

Department of Computer Science and Engineering
National Institute of Technology ,Silchar



PROJECT OUTLINE :

- Introduction
- Problem Statement
- Features
- How OOPS helped?
- Flow Chart
- Functionality
- Final consideration



INTRODUCTION



Objective:

To design and implement a supermarket billing system using C++ programming language.

Project Overview:

The project aims to create a user-friendly interface for user login ,adding items to the cart ,generating bills with discounts and discounts ,providing various payment methods for customers.

It provides functionalities like adding items to the cart ,selecting categories to display the stock and printing receipt.

Developed as a part of the CS 212 Object-Oriented Programming(OOP) course.

PROBLEM STATEMENT

Background:

Nowadays, supermarkets are an important part of our daily lives because they make shopping easy and convenient. But for the people who work there, keeping track of items, handling payments, and making correct bills can be a tough job. To make things easier and faster, there should be a proper Supermarket Billing System. This system can help the staff do their work more smoothly and also make customers happier by saving time and avoiding mistakes.

Problem Description :

You are tasked with designing and implementing a supermarket billing system using C++ programming language. The system should provide functionalities to display stock ,process customer transactions, and generate bills with itemized details.



FEATURES

User-Friendly Interface:

Clear and concise menu options for easy navigation, Visual feedback during item addition and bill generation for enhanced user experience.

Enhanced User Experience:

Incorporation of visual elements and sleep delays for a more interactive and engaging user experience and billing operations.

Error Handling:

Informative error messages for out-of-stock items or invalid user inputs. Robust file handling to prevent data loss or corruption.

Scalability:

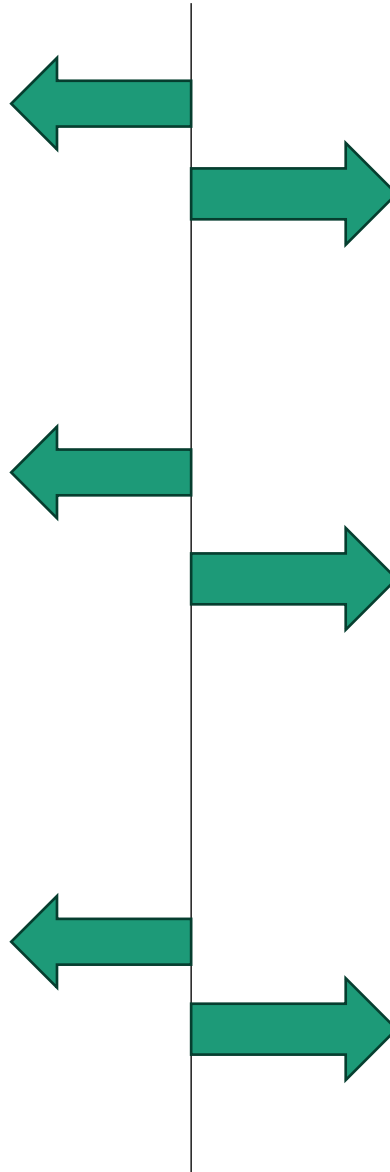
Designed to accommodate future expansions or modifications in functionality. Flexible code structure allowing easy integration of new features.

Customization Options:

Ability to customize bills with cashier ID, bill number, and timestamp for record-keeping and payment can be done based on user preference.

Billing System :

Add items to the cart with the desired quantity, calculate the total bill amount, including discounts and taxes.



IMPORTANT ROLE OF OOPS

Classes

Each class represents a distinct entity with its own set of properties and behaviours. Objects of these classes are created and manipulated throughout the program.



In my code, the Product class defines the properties and behavior of a product, while the Supermarket class defines the properties and behavior of a supermarket.

Encapsulation

implemented by using access specifiers (public, private, protected) to control the access to class members.



In my code, the Product class encapsulates the data members name, price, quantity, barcode, and category, as well as methods that operate on that data, such as getName() and getPrice().

Abstraction

Abstraction is achieved by hiding the internal implementation details and exposing only the essential features of an object.



In my code, the Product class abstracts the implementation details of a product, such as how the price is calculated, and only exposes the necessary information, such as the product name and price.

Inheritance

Inheritance is utilized to create a specialized class (Bill) from a base class (ItemDetails) to inherit common attributes and behaviours.



Inheritance is not explicitly used in your code. However, if you were to create a new class, such as Discounted Product, that inherits from the Product class, you could reuse the properties and behavior of the Product class and add new properties and behavior specific to discounted products.

Constructor Overloading

the concept of having more than one constructor with different parameters so that every constructor can perform a different task.



In my code, the Product class has a constructor that takes six parameters: name price, quantity, barcode, category, and bogoooffer. This is an example of constructor overloading, where a single constructor can be used to initialize objects with different parameters.

Destructor

a member function that is invoked automatically when the object goes out of scope or is explicitly destroyed by a call to delete or delete[]



My code does not explicitly define a destructor for the Product or Supermarket classes. However, the compiler will generate a default destructor for each class, which will automatically release any dynamically allocated memory.

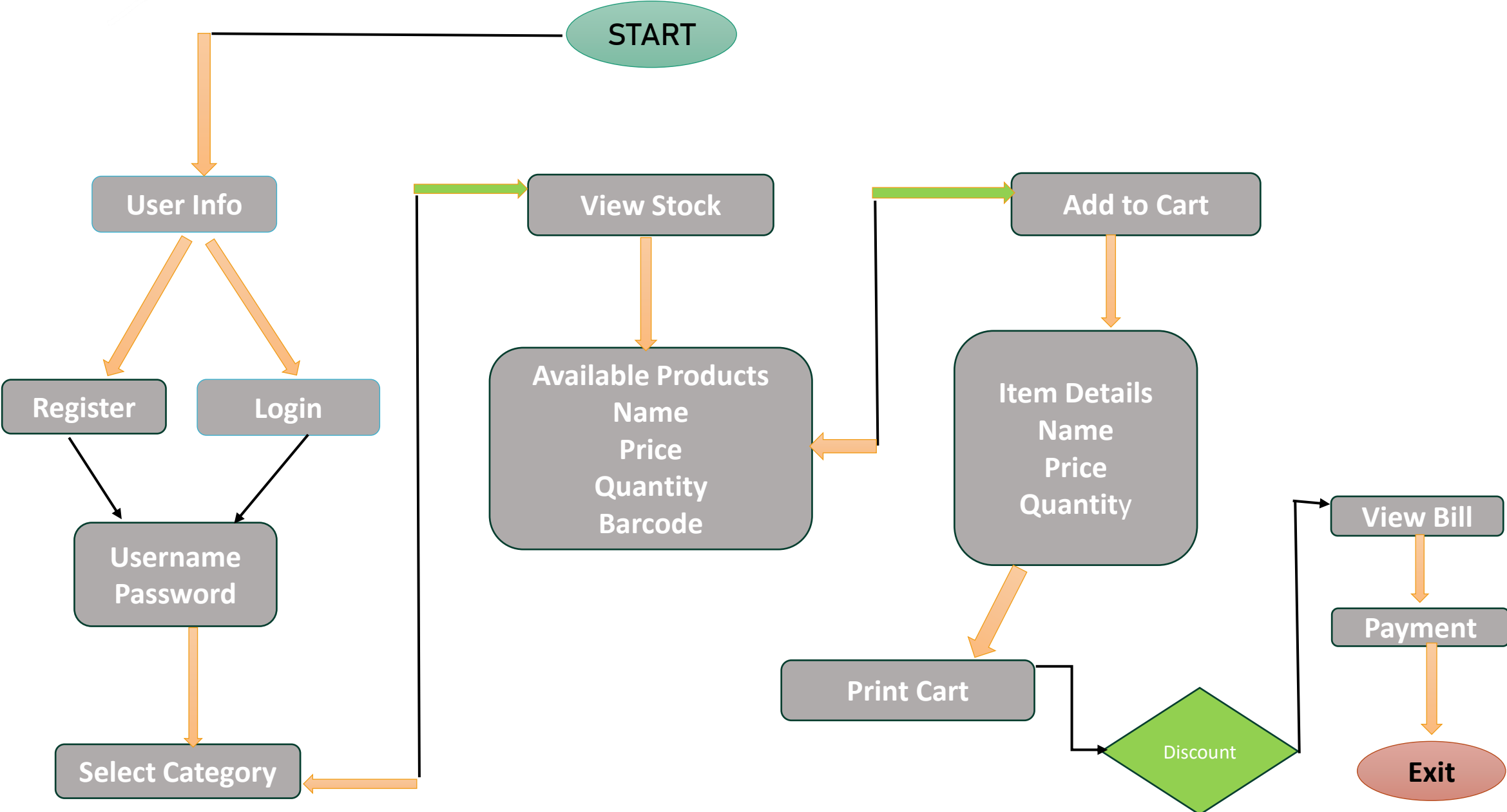
File Handling (I/O)

File handling is utilised to read and write data from/to external files, such as maintaining the inventory (shop.txt) of the super market.

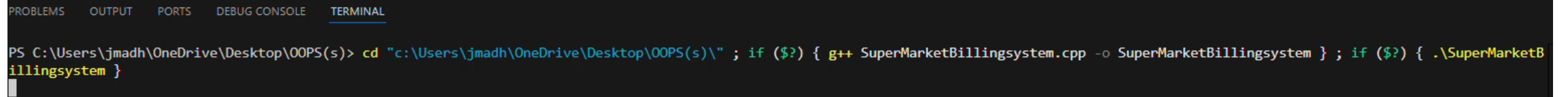


In my code, file handling is used to generate receipts and save them to a file. The generateReceipt function uses an ofstream object to create an output file stream, and the << operator is used to write data to the file.

FLOW CHART



FUNCTIONALITY OF PROJECT



A screenshot of a Visual Studio Code terminal window. The terminal has a dark background with a light-colored border at the top containing tabs for 'PROBLEMS', 'OUTPUT', 'PORTS', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active. The command prompt shows the user is in a directory 'C:\Users\jmadh\OneDrive\Desktop\OOPS(s)' and has entered a command to compile and run a C++ program. The command is: `cd "c:\Users\jmadh\OneDrive\Desktop\OOPS(s)" ; if ($?) { g++ SuperMarketBillingsystem.cpp -o SuperMarketBillingsystem } ; if ($?) { .\SuperMarketBillingsystem }`. The command is partially visible, with the first line showing the directory change and the compilation command, and the second line showing the execution command. A cursor is visible at the end of the first line.

```
PS C:\Users\jmadh\OneDrive\Desktop\OOPS(s)> cd "c:\Users\jmadh\OneDrive\Desktop\OOPS(s)" ; if ($?) { g++ SuperMarketBillingsystem.cpp -o SuperMarketBillingsystem } ; if ($?) { .\SuperMarketB
illingsystem }
```

MAIN MENU

```
graph TD; MM[MAIN MENU] --> R[1. Register<br/>2. Login<br/>Choose option: 1<br/>Enter username: lavanya<br/>Enter password: 123456<br/>Registration successfull]; MM --> C[10 x Chips added to the cart.<br/>1 x Chain added to the cart.<br/>2 x Track pant added to the cart.<br/>4 x Moongdal added to the cart.<br/>Special Offer Applied: Buy One Get One Free on sprite!<br/>10 x sprite added to the cart.]; MM --> P[Select Category:<br/>1. Fruits<br/>2. Snacks<br/>3. Juices<br/>4. Accessories<br/>5. Clothes<br/>6. All<br/>Enter your choice: 6]; R --> C; C --> P; P --> PM[Select Payment Method:<br/>1. Cash<br/>2. Credit/Debit Card<br/>3. Net Banking<br/>4. UPI (Paytm, Google Pay, PhonePe)<br/>Enter choice (1-4): ]; PM --> RPT[Receipt has been saved as 'receipt_1041.txt'.]
```

1. Register
2. Login
Choose option: 1
Enter username: lavanya
Enter password: 123456
Registration successfull

10 x Chips added to the cart.
1 x Chain added to the cart.
2 x Track pant added to the cart.
4 x Moongdal added to the cart.
Special Offer Applied: Buy One Get One Free on sprite!
10 x sprite added to the cart.

Receipt has been saved as 'receipt_1041.txt'.
Select Payment Method:
1. Cash
2. Credit/Debit Card
3. Net Banking
4. UPI (Paytm, Google Pay, PhonePe)
Enter choice (1-4):

Select Category:
1. Fruits
2. Snacks
3. Juices
4. Accessories
5. Clothes
6. All
Enter your choice: 6

TOTAL
BILL
OF
SELECTED
ITEMS

```
receipt_laavanya_1745499776.txt
1  ----- SUPERMARKET RECEIPT -----
2  Customer: laavanya
3  Date: 2025-4-24
4  -----
5  T-Shirt x3 = ₹897
6  Soda x5 = ₹185 (BOGO)
7  Subtotal: ₹1082
8  Discount: ₹100.2
9  GST: ₹162.324
10 Total: ₹1064.12
11 -----
12
```

FINAL CONSIDERATION

OUR TEAM MEMBERS



LAVANYA SRI CHAVA
2312069



DIVYA KUMARI
2312077

**THANK
YOU**

CONTACT US:
9985175689
9508940077