D Y PATIL
COLLEGE of
ENGINEERING & TECHNOLOGY
(AN AUTONOMOUS INSTITUTE)
KASABA BAWADA, KOLHAPUR

A

PROJECT REPORT

On

" Supermarket Billing system"

Submitted by

Name	Roll no.
Prasanna Pradeep Bhurke	155
Pranav Buddham Karnik	138
Pratik Dattatray Patil	142
Vyanktesh Harishchandra Madhyapgol	149
Satyajeet Sanjay kasbekar	139

Mrs. S. G. Ambi
Project Guide

Mrs. R. J. Dhanal
Head of the Department

- ## Description :-

The **Supermarket Billing System** is a simple yet efficient C++ program designed to make everyday supermarket operations easier. It helps manage products in the store by allowing users to add, update, or remove items from the inventory and automatically saves the data for future use. During billing, the system lets the cashier select items, calculates the total amount, applies discounts, adds taxes, and prints a neat receipt. It ensures accuracy in transactions and reduces manual effort through automation. The system is built using object-oriented programming, making it organized, modular, and easy to maintain. With its clear, menu-driven console interface, it provides a smooth and reliable experience for both shopkeepers and customers.

- ## C++ program :-

```
1// supermarket_ooc.cpp
// Object-Oriented Supermarket Billing System
// Compile: g++ -std=c++17 supermarket_ooc.cpp -o supermarket

#include <bits/stdc++.h>
using namespace std;

// ==================== ITEM ====================
class Item {
public:
    int id;
    string name;
    double price;
    int stock;
```

```
    Item() : id(0), price(0.0), stock(0) {}
    Item(int i, const string &n, double p, int s) : id(i), name(n), price(p), stock(s) {}
};

// ==================== INVENTORY ====================
class Inventory {
    unordered_map<int, Item> items;

public:
    bool addItem(const Item &it) {
        if (items.count(it.id)) return false;
        items[it.id] = it;
        return true;
    }

    bool updateItem(int id, const string &name, double price, int stock) {
        auto it = items.find(id);
        if (it == items.end()) return false;
        it->second.name = name;
        it->second.price = price;
        it->second.stock = stock;
        return true;
    }

    bool removeItem(int id) {
        return items.erase(id) > 0;
    }

    Item *findItem(int id) {
        auto it = items.find(id);
```

```
    if (it == items.end()) return nullptr;
    return &it->second;
}

vector<Item> listItems() const {
    vector<Item> out;
    for (auto &kv : items) out.push_back(kv.second);
    sort(out.begin(), out.end(), [](auto &a, auto &b) { return a.id < b.id; });
    return out;
}

bool saveToFile(const string &filename) const {
    ofstream ofs(filename);
    if (!ofs) return false;
    for (auto &kv : items) {
        auto &it = kv.second;
        string name = it.name;
        for (char &c : name) if (c == ',') c = ';';
        ofs << it.id << ',' << name << ',' << fixed << setprecision(2)
            << it.price << ',' << it.stock << '\n';
    }
    return true;
}

bool loadFromFile(const string &filename) {
    ifstream ifs(filename);
    if (!ifs) return false;
    items.clear();
    string line;
    while (getline(ifs, line)) {
```

```
            if (line.empty()) continue;
            stringstream ss(line);
            string id_s, name, price_s, stock_s;
            getline(ss, id_s, ',');
            getline(ss, name, ',');
            getline(ss, price_s, ',');
            getline(ss, stock_s, ',');
            int id = stoi(id_s);
            double price = stod(price_s);
            int stock = stoi(stock_s);
            items[id] = Item(id, name, price, stock);
        }
        return true;
    }

    void show() const {
        cout << left << setw(6) << "ID" << setw(25) << "Name"
             << setw(10) << "Price" << setw(8) << "Stock" << '\n';
        cout << string(55, '-') << '\n';
        for (auto &it : listItems()) {
            cout << left << setw(6) << it.id << setw(25) << it.name
                 << setw(10) << fixed << setprecision(2) << it.price
                 << setw(8) << it.stock << '\n';
        }
    }
};

// ==================== BILLING ====================
struct LineItem {
    int id;
```

```
    string name;
    double unitPrice;
    int qty;
    double total() const { return unitPrice * qty; }
};

class Billing {
    vector<LineItem> lines;
    const double TAX_RATE = 0.05;
    const double DISCOUNT_THRESHOLD = 1000.0;
    const double DISCOUNT_RATE = 0.03;

public:
    bool addLine(Item &item, int qty, Inventory &inv) {
        if (qty <= 0 || item.stock < qty) return false;
        item.stock -= qty;
        lines.push_back({item.id, item.name, item.price, qty});
        return true;
    }

    double subtotal() const {
        double s = 0;
        for (auto &l : lines) s += l.total();
        return s;
    }

    double discount() const {
        return subtotal() >= DISCOUNT_THRESHOLD ? subtotal() * DISCOUNT_RATE : 0;
    }
```

```
double tax() const {
    return (subtotal() - discount()) * TAX_RATE;
}

double total() const {
    return subtotal() - discount() + tax();
}

static string currentDateTime() {
    auto t = chrono::system_clock::now();
    time_t tt = chrono::system_clock::to_time_t(t);
    string s = ctime(&tt);
    if (!s.empty() && s.back() == '\n') s.pop_back();
    return s;
}

static string truncate(const string &s, size_t n) {
    return s.size() <= n ? s : s.substr(0, n - 3) + "...";
}

void printReceipt(const string &cashier) const {
    cout << "\n------------------ RECEIPT ------------------\n";
    cout << "Cashier: " << cashier << "    Date: " << currentDateTime() << "\n\n";
    cout << left << setw(6) << "ID" << setw(20) << "Name" << setw(8) << "Price"
         << setw(6) << "Qty" << setw(10) << "Total" << "\n";
    cout << string(58, '-') << '\n';
    for (auto &l : lines) {
        cout << left << setw(6) << l.id << setw(20) << truncate(l.name, 18)
             << setw(8) << fixed << setprecision(2) << l.unitPrice
             << setw(6) << l.qty << setw(10) << l.total() << '\n';
```

```
        }
        cout << string(58, '-') << '\n';
        cout << setw(40) << right << "Subtotal: " << setw(10) << subtotal() << '\n';
        cout << setw(40) << right << "Discount: " << setw(10) << discount() << '\n';
        cout << setw(40) << right << "Tax: " << setw(10) << tax() << '\n';
        cout << setw(40) << right << "Total: " << setw(10) << total() << '\n';
        cout << "------------------------------------------\n";
        cout << "Thank you for shopping with us!\n";
    }
};

// ==================== UTILS ====================
class InputHelper {
public:
    static int getInt(const string &prompt = "") {
        if (!prompt.empty()) cout << prompt;
        int x;
        while (!(cin >> x)) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid input. Enter integer: ";
        }
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return x;
    }

    static double getDouble(const string &prompt = "") {
        if (!prompt.empty()) cout << prompt;
        double x;
        while (!(cin >> x)) {
```

```
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid input. Enter number: ";
        }
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return x;
    }

    static string getLine(const string &prompt = "") {
        if (!prompt.empty()) cout << prompt;
        string s;
        getline(cin, s);
        size_t a = s.find_first_not_of(" \t\r\n");
        if (a == string::npos) return "";
        size_t b = s.find_last_not_of(" \t\r\n");
        return s.substr(a, b - a + 1);
    }

    static void pause() {
        cout << "\nPress Enter to continue...";
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    }
};
```

```
// ==================== SUPERMARKET SYSTEM ====================
class SupermarketSystem {
    Inventory inv;

public:
    SupermarketSystem() {
        inv.loadFromFile("inventory.csv");
    }

    void run() {
        while (true) {
            int choice = mainMenu();
            if (choice == 0) {
                cout << "Exiting. Goodbye!\n";
                break;
            }
            switch (choice) {
                case 1: manageInventory(); break;
                case 2: createBill(); break;
                case 3: loadInventory(); break;
                case 4: saveInventory(); break;
                default: cout << "Invalid option.\n"; break;
            }
        }
    }

private:
    int mainMenu() {
        cout << "\n=== Supermarket Billing System ===\n"
```

```
        << "1. Manage Inventory\n"
        << "2. Billing (Create new bill)\n"
        << "3. Load inventory from file\n"
        << "4. Save inventory to file\n"
        << "0. Exit\n"
        << "Choose option: ";
    return InputHelper::getInt();
}

void manageInventory() {
    while (true) {
        cout << "\n--- Inventory Management ---\n"
             << "1. Add item\n2. Update item\n3. Remove item\n4. Show all items\n0.
Back\nChoose: ";
        int c = InputHelper::getInt();
        if (c == 0) break;
        switch (c) {
            case 1: addItem(); break;
            case 2: updateItem(); break;
            case 3: removeItem(); break;
            case 4: showItems(); break;
            default: cout << "Invalid option.\n";
        }
    }
}

void addItem() {
    int id = InputHelper::getInt("Enter ID: ");
    string name = InputHelper::getLine("Enter name: ");
    double price = InputHelper::getDouble("Enter price: ");
    int stock = InputHelper::getInt("Enter stock: ");
```

```
  if (inv.addItem(Item(id, name, price, stock)))
        cout << "Item added successfully.\n";
    else
        cout << "Item with this ID already exists.\n";
  }

  void updateItem() {
    int id = InputHelper::getInt("Enter ID to update: ");
    Item *p = inv.findItem(id);
    if (!p) { cout << "Item not found.\n"; return; }
    cout << "Current Name: " << p->name << " | Price: " << p->price << " | Stock: " << p->stock << '\n';
    string name = InputHelper::getLine("New name: ");
    double price = InputHelper::getDouble("New price: ");
    int stock = InputHelper::getInt("New stock: ");
    inv.updateItem(id, name, price, stock);
    cout << "Item updated.\n";
  }

  void removeItem() {
    int id = InputHelper::getInt("Enter ID to remove: ");
    if (inv.removeItem(id)) cout << "Item removed.\n";
    else cout << "Item not found.\n";
  }

  void showItems() {
    inv.show();
  }
```

```
    void createBill() {
        Billing bill;
        string cashier = InputHelper::getLine("Enter cashier name: ");
        if (cashier.empty()) cashier = "Cashier";
        while (true) {
            cout << "\n--- Billing Menu ---\n"
                 << "1. Add item to bill\n2. Show subtotal\n3. Print receipt\n0. Cancel\nChoose: ";
            int c = InputHelper::getInt();
            if (c == 0) break;
            else if (c == 1) {
                int id = InputHelper::getInt("Enter item ID: ");
                Item *p = inv.findItem(id);
                if (!p) { cout << "Item not found.\n"; continue; }
                cout << "Found: " << p->name << " | Price: " << p->price << " | Stock: " << p-
>stock << '\n';
                int qty = InputHelper::getInt("Enter quantity: ");
                if (bill.addLine(*p, qty, inv)) cout << "Added to bill.\n";
                else cout << "Failed to add item.\n";
            } else if (c == 2) {
                cout << "Subtotal: " << fixed << setprecision(2) << bill.subtotal() << '\n';
            } else if (c == 3) {
                bill.printReceipt(cashier);
                break;
            } else cout << "Invalid option.\n";
        }
    }
```

```
    void loadInventory() {
        string f = InputHelper::getLine("Enter filename (default inventory.csv): ");
        if (f.empty()) f = "inventory.csv";
        if (inv.loadFromFile(f)) cout << "Loaded successfully.\n";
        else cout << "Failed to load.\n";
    }

    void saveInventory() {
        string f = InputHelper::getLine("Enter filename (default inventory.csv): ");
        if (f.empty()) f = "inventory.csv";
        if (inv.saveToFile(f)) cout << "Saved successfully.\n";
        else cout << "Failed to save.\n";
    }
};

// ==================== MAIN ====================
int main() {
    SupermarketSystem system;
    system.run();
    return 0;
}
```

- **OUTPUT :-**

```
PS C:\Users\PRASANNA> cd "C:\Users\PRASANNA\

=== Supermarket Billing System ===
1. Manage Inventory
2. Billing (Create new bill)
3. Load inventory from file
4. Save inventory to file
0. Exit
Choose option:
```

```
0. Exit
Choose option: 1

--- Inventory Management ---
1. Add item
2. Update item
3. Remove item
4. Show all items
0. Back
Choose: 1
Enter ID: 10
Enter name: milk
Enter price: 100
Enter stock: 5
Item added successfully.

--- Inventory Management ---
1. Add item
2. Update item
3. Remove item
4. Show all items
0. Back
Choose: 4
ID    Name                    Price     Stock
-----------------------------------------------------
10    milk                    100.00    5
```

```
1. Manage Inventory
2. Billing (Create new bill)
3. Load inventory from file
4. Save inventory to file
0. Exit
Choose option: 2
Enter cashier name: Prasanna

--- Billing Menu ---
1. Add item to bill
2. Show subtotal
3. Print receipt
0. Cancel
Choose: 1
Enter item ID: 10
Found: milk | Price: 100.00 | Stock: 5
Enter quantity: 1
Added to bill.

--- Billing Menu ---
1. Add item to bill
2. Show subtotal
3. Print receipt
0. Cancel
Choose: 3

------------------ RECEIPT ------------------
Cashier: Prasanna    Date: Tue Nov 11 08:09:21 2025

ID    Name                Price   Qty   Total
-----------------------------------------------------------
10    milk                100.00  1     100.00
-----------------------------------------------------------
                              Subtotal:     100.00
                              Discount:       0.00
                                   Tax:       5.00
                                 Total:     105.00
---------------------------------------------
Thank you for shopping with us!
```