**Mostafa abdelrazik (232601)**

**Mostafa omar (235869)**

**Customer billing management system**

Explination :

The Customer Billing Management System is a console application developed in C++ to facilitate billing operations in various businesses such as departmental stores, shops, cafes, etc. It serves as a practical demonstration of C++ programming language features while providing a functional tool for managing customer records, billing information, and payment tracking.

Objective :

The Objective of this project is to help any stores such as cafes , shops or departmental stores to keep records of their customers such as name , address , mobile number , paid amount ,due amount and payment date , it also make the shop owners edit or view their regular customers records and if they have a new customer they can add the customer's records and edit or view them anytime they want .

CODE :

#include <iostream>

#include <vector>

#include <string>

#include <fstream>

#include <sstream>

using namespace std;

class Customer {

private:

string Name;

string Address;

string MobileNumber;

double PaidAmount;

double DueAmount;

string PaymentDate;

public:

Customer(string name, string address, string mobileNumber, double paidAmount, double dueAmount, string paymentDate)

: Name(name), Address(address), MobileNumber(mobileNumber), PaidAmount(paidAmount), DueAmount(dueAmount), PaymentDate(paymentDate) {}

string GetName() const { return Name; }

void SetName(string name) { Name = name; }

string GetAddress() const { return Address; }

void SetAddress(string address) { Address = address; }

string GetMobileNumber() const { return MobileNumber; }

void SetMobileNumber(string mobileNumber) { MobileNumber = mobileNumber; }

double GetPaidAmount() const { return PaidAmount; }

void SetPaidAmount(double paidAmount) { PaidAmount = paidAmount; }

double GetDueAmount() const { return DueAmount; }

void SetDueAmount(double dueAmount) { DueAmount = dueAmount; }

string GetPaymentDate() const { return PaymentDate; }

void SetPaymentDate(string paymentDate) { PaymentDate = paymentDate; }

void Display() {

cout << "customer name " << Name << endl;

cout << "customer address " << Address << endl;

cout << "customer mob number" << MobileNumber << endl;

char choice;

do {

cout << "Do you want to see the customer receipt? (1or2): ";

cin >> choice;

switch (choice) {

case '1':

cout << "\*\*\*\*\*\*\*" << "RECEIPT" << "\*\*\*\*\*\*" << endl;

cout << "-------------------------" << endl;

cout << "Customer Name: " << Name << endl;

cout << "Address: " << Address << endl;

cout << "Mobile Number: " << MobileNumber << endl;

cout << "-------------------------" << endl;

cout << "Payment Details:" << endl;

cout << "Paid Amount: " << PaidAmount << endl;

cout << "Due Amount: " << DueAmount << endl;

cout << "Payment Date: " << PaymentDate << endl;

cout << "-------------------------" << endl;

cout << "\*\*\*\*\*" << " THANK U FOR UR PURCHASE <3" << "\*\*\*\*" << endl;

cout << " printed at " << PaymentDate << endl;

break;

case' 2':

break;

default:

cout << "Invalid choice. Please enter 1or 2." << endl;

break;

}

} while (choice == 1 || choice == 2);

}

}; class CustomerManager {

private:

vector<Customer> Customers;

string Filename;

public:

CustomerManager(const string& filename) : Filename(filename) {}

void AddCustomer(const Customer& customer) {

Customers.push\_back(customer);

}

void EditCustomer(const string& name, const Customer& newCustomerInfo) {

for (auto& customer : Customers) {

if (customer.GetName() == name) {

customer = newCustomerInfo;

break;

}

}

}

Customer GetCustomerByName(const string& name) {

for (const auto& customer : Customers) {

if (customer.GetName() == name) {

return customer;

}

}

return Customer("", "", "", 0.0, 0.0, "");

}

void LoadFromFile() {

ifstream file(Filename);

if (file.is\_open()) {

string line;

while (getline(file, line)) {

stringstream ss(line);

string name, address, mobileNumber, paymentDate;

double paidAmount, dueAmount;

getline(ss, name, ',');

getline(ss, address, ',');

getline(ss, mobileNumber, ',');

ss >> paidAmount;

ss.ignore();

ss >> dueAmount;

ss.ignore();

getline(ss, paymentDate, ',');

Customer customer(name, address, mobileNumber, paidAmount, dueAmount, paymentDate);

Customers.push\_back(customer);

}

file.close();

}

else {

cout << "Unable to open file: " << Filename << endl;

}

}

void SaveToFile() {

ofstream file(Filename);

if (file.is\_open()) {

for (const auto& customer : Customers) {

file << customer.GetName() << "," << customer.GetAddress() << "," << customer.GetMobileNumber() << ","

<< customer.GetPaidAmount() << "," << customer.GetDueAmount() << "," << customer.GetPaymentDate() << endl;

}

file.close();

}

else {

cout << "Unable to create file: " << Filename << endl;

}

}

};

class MainApplication {

private:

CustomerManager customerManager;

public:

MainApplication(const string& filename) : customerManager(filename) {}

void DisplayMenu() {

cout << "Main Menu" << endl;

cout << "1. Add Customer" << endl;

cout << "2. Edit Customer" << endl;

cout << "3. View Customer" << endl;

cout << "4. Save Data" << endl;

cout << "5. Load Data" << endl;

cout << "6. Exit" << endl;

}

void AddCustomerOption() {

string name, address, mobileNumber, paymentDate;

double paidAmount, dueAmount;

cout << "Enter Name: ";

cin >> name;

cout << "Enter Address: ";

cin >> address;

cout << "Enter Mobile Number: ";

cin >> mobileNumber;

cout << "Enter Paid Amount: ";

cin >> paidAmount;

cout << "Enter Due Amount: ";

cin >> dueAmount;

cout << "Enter Payment Date: ";

cin >> paymentDate;

Customer newCustomer(name, address, mobileNumber, paidAmount, dueAmount, paymentDate);

customerManager.AddCustomer(newCustomer);

cout << "Customer added successfully." << endl;

char choice;

do {

cout << "Do you want to print a receipt? (1or2): ";

cin >> choice;

switch (choice) {

case '1':

cout << "\*\*\*\*\*\*\*" << "RECEIPT" << "\*\*\*\*\*\*" << endl;

cout << "-------------------------" << endl;

cout << "Customer Name: " << name << endl;

cout << "Address: " << address << endl;

cout << "Mobile Number: " << mobileNumber << endl;

cout << "-------------------------" << endl;

cout << "Payment Details:" << endl;

cout << "Paid Amount: " << paidAmount << endl;

cout << "Due Amount: " << dueAmount << endl;

cout << "Payment Date: " << paymentDate << endl;

cout << "-------------------------" << endl;

cout << "\*\*\*\*\*" << " THANK U FOR UR PURCHASE <3" << "\*\*\*\*" << endl;

cout << " printed at " << paymentDate << endl;

break;

case'2': break;

default:

cout << "Invalid choice. Please enter 1or 2." << endl;

break;

}

} while (choice == 1 || choice == 2);

}

void EditCustomerOption() {

string name;

cout << "Enter Name of Customer to Edit: ";

cin >> name;

Customer existingCustomer = customerManager.GetCustomerByName(name);

if (existingCustomer.GetName() != "") {

string newName, newAddress, newMobileNumber, newPaymentDate;

double newPaidAmount, newDueAmount;

cout << "Enter New Name: ";

cin >> newName;

cout << "Enter New Address: ";

cin >> newAddress;

cout << "Enter New Mobile Number: ";

cin >> newMobileNumber;

cout << "Enter New Paid Amount: ";

cin >> newPaidAmount;

cout << "Enter New Due Amount: ";

cin >> newDueAmount;

cout << "Enter New Payment Date: ";

cin >> newPaymentDate;

char choice;

do {

cout << "Do you want to print a receipt? (1or2): ";

cin >> choice;

switch (choice) {

case '1':

cout << "\*\*\*\*\*\*\*" << "RECEIPT" << "\*\*\*\*\*\*" << endl;

cout << "-------------------------" << endl;

cout << "Customer Name: " << newName << endl;

cout << "Address: " << newAddress << endl;

cout << "Mobile Number: " << newMobileNumber << endl;

cout << "-------------------------" << endl;

cout << "Payment Details:" << endl;

cout << "Paid Amount: " << newPaidAmount << endl;

cout << "Due Amount: " << newDueAmount << endl;

cout << "Payment Date: " << newPaymentDate << endl;

cout << "-------------------------" << endl;

cout << "\*\*\*\*\*" << " THANK U FOR UR PURCHASE <3" << "\*\*\*\*" << endl;

cout << " printed at " << newPaymentDate << endl;

break;

case'2':

break;

default:

cout << "Invalid choice. Please enter 1or 2." << endl;

break;

}

} while (choice == 1 || choice == 2);

Customer newCustomer(newName, newAddress, newMobileNumber, newPaidAmount, newDueAmount, newPaymentDate);

customerManager.EditCustomer(name, newCustomer);

cout << "Customer information updated successfully." << endl;

}

else {

cout << "Customer not found." << endl;

}

}

void ViewCustomerOption() {

string name;

cout << "Enter Name of Customer to View: ";

cin >> name;

Customer customer = customerManager.GetCustomerByName(name);

if (customer.GetName() != "") {

cout << "Customer Details:" << endl;

customer.Display();

}

else {

cout << "Customer not found." << endl;

}

}

void SaveDataOption() {

customerManager.SaveToFile();

cout << "Customer data saved to file." << endl;

}

void LoadDataOption() {

customerManager.LoadFromFile();

cout << "Customer data loaded from file." << endl;

}

};

int main() {

MainApplication app("customers.csv");

int choice;

do {

app.DisplayMenu();

cout << "enter a nuber : ";

cin >> choice;

switch (choice) {

case 1:

app.AddCustomerOption();

break;

case 2:

app.EditCustomerOption();

break;

case 3:

app.ViewCustomerOption();

break;

case 4:

app.SaveDataOption();

break;

case 5:

app.LoadDataOption();

break;

case 6:

cout << " exited the applicaton succesfully ya5oya" << endl;

break;

default:

cout << "please enter a number between 1 and 6." << endl;

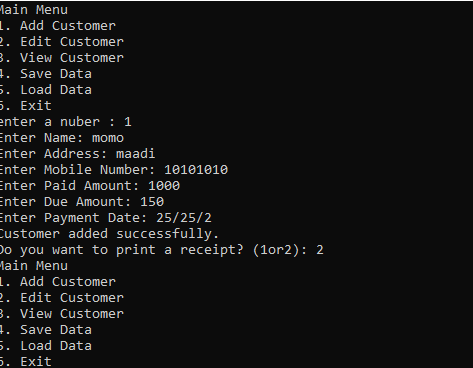
break;

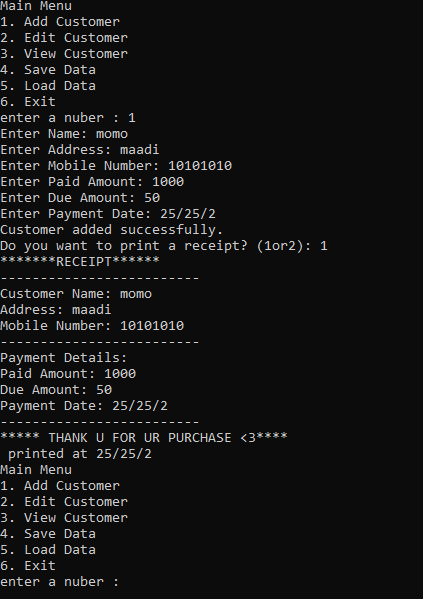
}

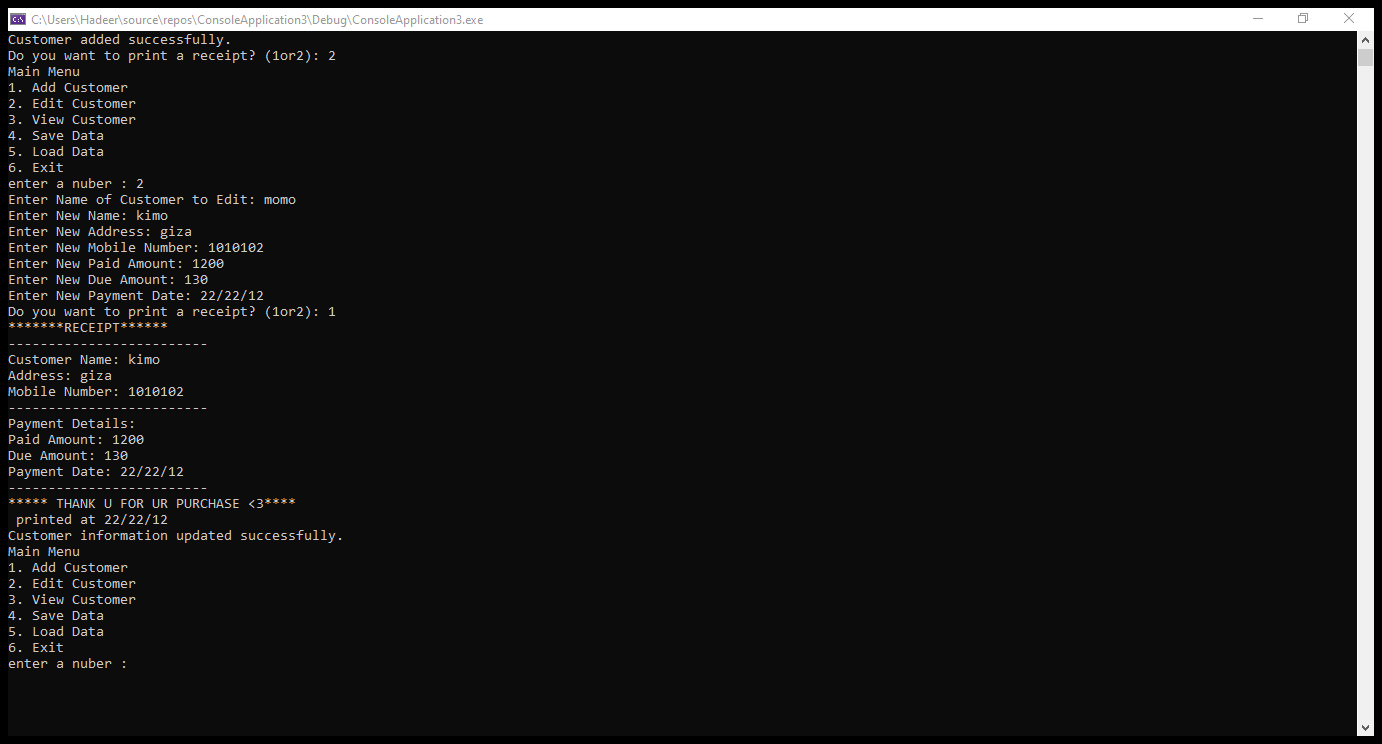
} while (choice != 6);

return 0; }

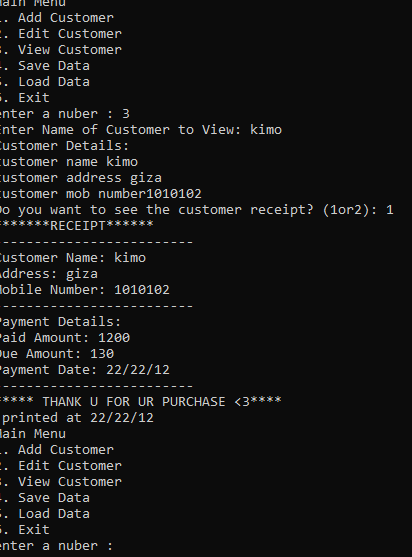
OUTPUTS: here it shows a customer data being added and the user chose not to print a receipt.



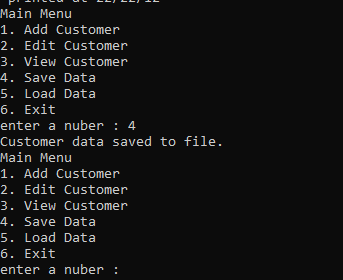
If the user chose to print a receipt :

here after the customer data was added the user chose to edit the data and print a receipt with the new data

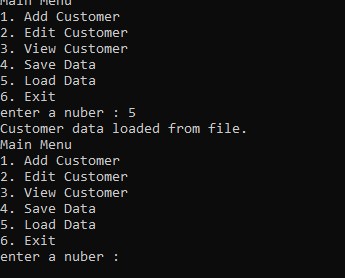
Here the user chose to view customer records and their receipts.



The user chose to save data:



Here the user chose to load data :



The user chose to exit :

