Internship project

-june 2025

banking

Code

#include <iostream>

#include <vector>

#include <ctime>

#include <string>

#include <sstream>

using namespace std;

// Global ID counters

int globalCustomerId = 1000;

int globalAccountNumber = 2000;

int globalTransactionId = 3000;

// ----- Customer Class -----

class Customer {

public:

int customerID;

string name;

string address;

string phone;

Customer(string n, string a, string p) {

customerID = ++globalCustomerId;

name = n;

address = a;

phone = p;

}

};

// ----- Account Class -----

class Account {

public:

int accountNumber;

int customerID;

double balance;

Account(int cID) {

accountNumber = ++globalAccountNumber;

customerID = cID;

balance = 0.0;

}

void deposit(double amt) {

balance += amt;

}

bool withdraw(double amt) {

if (amt <= balance) {

balance -= amt;

return true;

}

return false;

}

};

// ----- Transaction Class -----

class Transaction {

public:

int transactionID;

int fromAccount;

int toAccount;

string type;

double amount;

string timestamp;

Transaction(int from, int to, string t, double amt) {

transactionID = ++globalTransactionId;

fromAccount = from;

toAccount = to;

type = t;

amount = amt;

time\_t now = time(0);

char\* dt = ctime(&now);

timestamp = string(dt);

if (timestamp[timestamp.length() - 1] == '\n') {

timestamp = timestamp.substr(0, timestamp.length() - 1);

}

}

};

// ----- Global Vectors -----

vector<Customer> customers;

vector<Account> accounts;

vector<Transaction> transactions;

// ----- Helper: Int to String (for older compilers) -----

string intToStr(int n) {

stringstream ss;

ss << n;

return ss.str();

}

// ----- Helper Functions -----

Customer\* findCustomer(int id) {

for (int i = 0; i < (int)customers.size(); i++) {

if (customers[i].customerID == id)

return &customers[i];

}

return NULL;

}

Account\* findAccount(int accNo) {

for (int i = 0; i < (int)accounts.size(); i++) {

if (accounts[i].accountNumber == accNo)

return &accounts[i];

}

return NULL;

}

// ----- Display Functions -----

void displayCustomerInfo(Customer c) {

cout << "Customer ID: " << c.customerID << endl;

cout << "Name : " << c.name << endl;

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

cout << "Phone : " << c.phone << endl;

}

void displayAccountInfo(Account a) {

cout << "Account No : " << a.accountNumber << endl;

cout << "Customer ID: " << a.customerID << endl;

cout << "Balance : ?" << a.balance << endl;

}

void displayTransactionInfo(Transaction t) {

cout << "Txn ID : " << t.transactionID << endl;

cout << "Type : " << t.type << endl;

cout << "From : " << (t.fromAccount == -1 ? "-" : intToStr(t.fromAccount)) << endl;

cout << "To : " << (t.toAccount == -1 ? "-" : intToStr(t.toAccount)) << endl;

cout << "Amount : ?" << t.amount << endl;

cout << "Time : " << t.timestamp << endl;

}

void displayAllCustomers() {

for (int i = 0; i < (int)customers.size(); i++) {

cout << "\n--- Customer " << (i + 1) << " ---\n";

displayCustomerInfo(customers[i]);

}

}

void displayAllAccounts() {

for (int i = 0; i < (int)accounts.size(); i++) {

cout << "\n--- Account " << (i + 1) << " ---\n";

displayAccountInfo(accounts[i]);

}

}

void displayAllTransactions() {

for (int i = 0; i < (int)transactions.size(); i++) {

cout << "\n--- Transaction " << (i + 1) << " ---\n";

displayTransactionInfo(transactions[i]);

}

}

// ----- Banking Operations -----

void createCustomer() {

string name, address, phone;

cout << "\nEnter name: ";

cin.ignore();

getline(cin, name);

cout << "Enter address: ";

getline(cin, address);

cout << "Enter phone: ";

getline(cin, phone);

Customer c(name, address, phone);

customers.push\_back(c);

cout << "Customer created with ID: " << c.customerID << endl;

}

void createAccount() {

int cid;

cout << "\nEnter customer ID: ";

cin >> cid;

Customer\* c = findCustomer(cid);

if (c == NULL) {

cout << "Customer not found.\n";

return;

}

Account a(cid);

accounts.push\_back(a);

cout << "Account created. Account No: " << a.accountNumber << endl;

}

void deposit() {

int accNo;

double amt;

cout << "\nEnter account number: ";

cin >> accNo;

Account\* a = findAccount(accNo);

if (a == NULL) {

cout << "Account not found.\n";

return;

}

cout << "Enter amount to deposit: ?";

cin >> amt;

a->deposit(amt);

Transaction t(accNo, -1, "deposit", amt);

transactions.push\_back(t);

cout << "Deposit successful. Balance: ?" << a->balance << endl;

}

void withdraw() {

int accNo;

double amt;

cout << "\nEnter account number: ";

cin >> accNo;

Account\* a = findAccount(accNo);

if (a == NULL) {

cout << "Account not found.\n";

return;

}

cout << "Enter amount to withdraw: ?";

cin >> amt;

if (a->withdraw(amt)) {

Transaction t(accNo, -1, "withdraw", amt);

transactions.push\_back(t);

cout << "Withdrawal successful. Balance: ?" << a->balance << endl;

} else {

cout << "Insufficient balance.\n";

}

}

void transfer() {

int fromAcc, toAcc;

double amt;

cout << "\nEnter FROM account number: ";

cin >> fromAcc;

cout << "Enter TO account number: ";

cin >> toAcc;

Account\* from = findAccount(fromAcc);

Account\* to = findAccount(toAcc);

if (from == NULL || to == NULL) {

cout << "One or both accounts not found.\n";

return;

}

cout << "Enter amount to transfer: ?";

cin >> amt;

if (from->withdraw(amt)) {

to->deposit(amt);

Transaction t(fromAcc, toAcc, "transfer", amt);

transactions.push\_back(t);

cout << "Transfer successful.\n";

} else {

cout << "Insufficient balance.\n";

}

}

void showAccount() {

int accNo;

cout << "\nEnter account number: ";

cin >> accNo;

Account\* acc = findAccount(accNo);

if (acc == NULL) {

cout << "Account not found.\n";

return;

}

Customer\* cust = findCustomer(acc->customerID);

cout << "\n--- Account Details ---\n";

displayCustomerInfo(\*cust);

displayAccountInfo(\*acc);

cout << "\n--- Transactions ---\n";

for (int i = 0; i < (int)transactions.size(); i++) {

if (transactions[i].fromAccount == accNo || transactions[i].toAccount == accNo) {

displayTransactionInfo(transactions[i]);

cout << endl;

}

}

}

void displayAll() {

cout << "\n--- All Customers ---\n";

displayAllCustomers();

cout << "\n--- All Accounts ---\n";

displayAllAccounts();

cout << "\n--- All Transactions ---\n";

displayAllTransactions();

}

// ----- Main Menu -----

void mainMenu() {

int ch;

do {

cout << "\n====== BANKING SYSTEM MENU ======\n";

cout << "1. Create Customer\n";

cout << "2. Create Account\n";

cout << "3. Deposit\n";

cout << "4. Withdraw\n";

cout << "5. Transfer\n";

cout << "6. Show Account Info\n";

cout << "7. Display All Data\n";

cout << "8. Exit\n";

cout << "Enter choice: ";

cin >> ch;

switch (ch) {

case 1: createCustomer(); break;

case 2: createAccount(); break;

case 3: deposit(); break;

case 4: withdraw(); break;

case 5: transfer(); break;

case 6: showAccount(); break;

case 7: displayAll(); break;

case 8: cout << "Exiting...\n"; break;

default: cout << "Invalid choice.\n";

}

} while (ch != 8);

}

int main() {

mainMenu();

return 0;

}



