```
#include <iostream>
#include <string>
#include <vector>
#include <iomanip>
#include <limits>

using namespace std;

// Account class to store customer information
class Account {
private:
    string name;
    string accountNumber;
    double balance;
    string password;
    vector<string> transactionHistory;

public:
    Account(string n, string accNum, string pwd, double bal = 0.0)
        : name(n), accountNumber(accNum), password(pwd), balance(bal) {}

    // Getter methods
    string getName() const { return name; }
    string getAccountNumber() const { return accountNumber; }
    double getBalance() const { return balance; }
    bool checkPassword(string pwd) const { return password == pwd; }

    // Account operations
    void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
            string transaction = "Deposit: +$" + to_string(amount);
            transactionHistory.push_back(transaction);
            cout << "Deposit successful. New balance: $" << balance << endl;
        } else {
            cout << "Invalid deposit amount." << endl;
        }
    }

    void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            string transaction = "Withdrawal: -$" + to_string(amount);
            transactionHistory.push_back(transaction);
```

```
            cout << "Withdrawal successful. New balance: $" << balance << endl;
        } else {
            cout << "Invalid withdrawal amount or insufficient funds." << endl;
        }
}

void transfer(Account &recipient, double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            recipient.balance += amount;

            string senderTransaction = "Transfer to " + recipient.getAccountNumber() + ":
-$" + to_string(amount);
            string recipientTransaction = "Transfer from " + accountNumber + ": +$" +
to_string(amount);

            transactionHistory.push_back(senderTransaction);
            recipient.transactionHistory.push_back(recipientTransaction);

            cout << "Transfer successful. New balance: $" << balance << endl;
        } else {
            cout << "Invalid transfer amount or insufficient funds." << endl;
        }
    }

    void displayTransactionHistory() const {
        cout << "\nTransaction History for Account: " << accountNumber << endl;
        cout << "--------------------------------------" << endl;
        for (const auto &transaction : transactionHistory) {
            cout << transaction << endl;
        }
        cout << "--------------------------------------" << endl;
        cout << "Current Balance: $" << fixed << setprecision(2) << balance << endl;
    }
};

// Banking system class to manage accounts
class BankingSystem {
private:
    vector<Account> accounts;

public:
    void createAccount() {
        string name, accountNumber, password;
```

```
    double initialDeposit;

    cout << "\nEnter your name: ";
    cin.ignore();
    getline(cin, name);

    cout << "Create an account number: ";
    cin >> accountNumber;

    cout << "Create a password: ";
    cin >> password;

    cout << "Enter initial deposit amount: $";
    cin >> initialDeposit;

    accounts.emplace_back(name, accountNumber, password, initialDeposit);
    cout << "\nAccount created successfully!" << endl;
    cout << "Account Number: " << accountNumber << endl;
}

Account* login() {
    string accountNumber, password;
    cout << "\nEnter your account number: ";
    cin >> accountNumber;
    cout << "Enter your password: ";
    cin >> password;

    for (auto &account : accounts) {
        if (account.getAccountNumber() == accountNumber &&
account.checkPassword(password)) {
            return &account;
        }
    }
    cout << "Invalid account number or password." << endl;
    return nullptr;
}

void run() {
    while (true) {
        cout << "\n===== Online Banking System =====" << endl;
        cout << "1. Create Account" << endl;
        cout << "2. Login" << endl;
        cout << "3. Exit" << endl;
        cout << "Enter your choice: ";
```

```
int choice;
cin >> choice;

if (cin.fail()) {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Invalid input. Please enter a number." << endl;
        continue;
}

switch (choice) {
        case 1:
                createAccount();
                break;
        case 2: {
                Account* currentAccount = login();
                if (currentAccount) {
                        accountMenu(currentAccount);
                }
                break;
        }
                case 3:
                        cout << "Thank you for using our banking system. Goodbye!" << endl;
                        return;
                default:
                        cout << "Invalid choice. Please try again." << endl;
                }
        }
}

void accountMenu(Account* account) {
        while (true) {
                cout << "\n===== Account Menu =====" << endl;
                cout << "Welcome, " << account->getName() << "!" << endl;
                cout << "Account Number: " << account->getAccountNumber() << endl;
                cout << "Balance: $" << fixed << setprecision(2) << account->getBalance() <<
endl;
                cout << "1. Deposit" << endl;
                cout << "2. Withdraw" << endl;
                cout << "3. Transfer" << endl;
                cout << "4. View Transaction History" << endl;
                cout << "5. Logout" << endl;
                cout << "Enter your choice: ";
```

```
int choice;
cin >> choice;

if (cin.fail()) {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Invalid input. Please enter a number." << endl;
        continue;
}

switch (choice) {
        case 1: {
                double amount;
                cout << "Enter deposit amount: $";
                cin >> amount;
                account->deposit(amount);
                break;
        }
        case 2: {
                double amount;
                cout << "Enter withdrawal amount: $";
                cin >> amount;
                account->withdraw(amount);
                break;
}
case 3: {
        string recipientAccNum;
        double amount;
        cout << "Enter recipient account number: ";
        cin >> recipientAccNum;

        Account* recipient = nullptr;
        for (auto &acc : accounts) {
                if (acc.getAccountNumber() == recipientAccNum) {
                        recipient = &acc;
                        break;
                }
}

if (recipient) {
        cout << "Enter transfer amount: $";
        cin >> amount;
        account->transfer(*recipient, amount);
```

```
            }else{
                cout << "Recipient account not found." << endl;
            }
            break;
        }
        case 4:
            account->displayTransactionHistory();
            break;
        case 5:
                cout << "Logging out..." << endl;
                return;
            default:
                cout << "Invalid choice. Please try again." << endl;
            }
        }
    }
};

int main(){
    BankingSystem bank;
    bank.run();
    return 0;
}
```