COMP 2710: Project 1 – Phase 3: Implementation

A Secure Teller Terminal System

Design (60 points) – turned in via Canvas

No collaboration between students. Students should NOT share any project code with each other. Collaborations in any form will be treated as a serious violation of the University's academic integrity code.

Rating

- Design difficulty: 4/5
- Implementation difficulty: 2/5
- Time required: 4/5
- Fun: 3/5

Goals of Project 1:

- To design and implement a non-trivial application using classes, Constructors, Vectors, and operator overloading.
- To learn a security issue - Authentication.
- To perform Object-Oriented Analysis, Design, and Testing
- To develop a reasonably user-friendly application

1. Overview

1.1 **(45 points)** Implement the teller terminal system in C++

- ***Vectors.*** Your underlying data structure for lists of branch staff employees, clients, and accounts will be standard Vectors (**no arrays**). Thus, good classes to be considered include StaffVector; ClientVector, and AccountVector.

- ***Operator Overloading.*** You must at a minimum overload the comparison operator "= =" in the Password class to help you to compare the new password with the one stored in the password database.

1.2. **(15 points)** Test Results: *After developing the teller terminal system*, actually try all of your test cases (both system and unit testing). Actually show the results of your testing (a copy and paste from your program output is fine – don't stress too much about formatting as long as the results are clear). You should have test results for every test case you described. If your system doesn't behave as intended, you should note this. Note: Driver output will substitute for this phase.

2. Background Building a Secure Teller Terminal System

In this project, you will design and develop a secure teller terminal system for the Auburn branch of the Tiger Bank, which comprises multiple branches in the Alabama State. After the development of the secure teller terminal system, the system will be widely used in other branches of the Tiger Bank in Alabama. The Auburn branch manages a set of the Tiger bank's accounts, which contains a unique account number, a balance, as well as other important information. Once you complete this project, your terminal system at each Tiger Bank branch will be connected using insecure communications lines. If the Tiger Bank is satisfied with the teller terminal system developed by you, the Bank will sign another contract with you so that you can develop automated teller machines (ATMs) for the Tiger Bank.

In this project, you only will be focusing on the design and implementation of teller terminals for the Tiger Bank. A teller terminal system to be developed is a standard PC computer running Linux and handling local transactions. All human interaction with the teller terminal is through a keyboard. Other specialized input devices like smart-card readers are not considered a cost-effective option for our project.
Staffs at the Auburn bank branch access account information and customer data through the teller terminal system to be developed by you. In this project, let us refer to this information as *client data*. Client data must be carefully controlled in order to improve the security of the teller terminal system. Whether a given Auburn branch employee is permitted to access any particular piece of client data depends on the employee's job level.
In this prototype system, there are two types of users in the system - system

administrators and branch staffs. Any system administrator can add new branch staffs to access the system or delete existing staffs from the system (see sample usage 2.1, 2.2, and 2.3). Branch staff identifiers used in this system are called *user name*.

Your teller terminal system must **authenticate** each access request to the system. You will implement a mechanism to authenticate requests made at the teller terminal.

A teller terminal is either in the *inactive* state or the *active* state. In what follows, we summarize the behaviors of the teller terminal in these two states.

- **In Inactive State:**
  An inactive teller terminal displays a ***login menu*** (see Sample Usage 1). It invites a branch employee to enter a userid and password and either (i) to manage client and account information or (ii) add/delete a user (branch staff) if the user is a system administrator or (iii) to change his/her password (in which case, the teller is first authenticated using the current password, then the password is updated, and finally a session is initiated with the authenticated teller). In all the cases, the information provided is checked for validity. If the user name and password are valid, then the teller terminal system is placed in the active state and a session is started.

- **In Active State:**
  For system administrators, an active teller terminal displays a "System Administration" main menu that invites an administrator to manage information of branch staff employees, clients, and accounts (see sample usage 2.1).

  For branch staffs, an active teller displays a "Branch Staff" main menu that invites a branch staff to manage client and account information or change user password (see sample usage 3.2).

3. Requirement Details

3.1. System Login (see sample usage 1 and 2)

The teller terminal displays a **login menu** (see Sample Usage 1. Login). It invites a branch employee to enter a userid and password. If the user name and password are valid, then the teller terminal system is placed in the active state and a session with the teller terminal is started.

There are two types of users in the system: system administrators and branch staffs. Administrators can manage staff information. Both Administrators and staffs can manage client and account information. After login to the Teller Terminal System as an administrator, the system administrator can choose the fist item - "Client and Account Management" - from the system administration menu (see Sample Usage 1). Compared with system administrators, branch staffs can easily start managing clients and

transactions. In other words, a branch staff needs to login to the Teller Terminal System to manage clients and transactions (see Sample Usage 4)

3.1.1 Login as a system administrator (see sample usage 1)
A system administrator can either (i) to manage client and account information, (ii) add/delete/display users (branch staff employees), or (iii) to change his/her password. When a system administrator login to the system, the administrator can choose to do the following:

- Client and Account Management
- Add a branch staff
- Delete a branch staff
- Display branch staffs
- Change password

3.1.2 Default system administrator and password (see sample usage 1)
When the system is running for the first time, there is only a **default system administrator** – named "admin" - in the system. The **default password** of the default administrator is initially chosen to be **0000**. Of course, this password can be changed later. Initially (i.e., before the first run), there is no system administrators and branch staffs except the default administrator. Thus, only admin can use 0000 as the password to access the system

3.1.3 Login as a branch staff employee (see sample usage 3.2)
When a branch staff employee login to the teller terminal, the staff can perform the following two tasks:

- Client and Account Management
- Change password

3.1.4 Change password (see sample usage 2.4)
Regardless of administrators or staffs, the user is first authenticated using his/her current password. Then, the password is updated. A changed password is valid only if (i) it is not equal to the old password and (ii) it is a non-empty password.

3.1.5 Masking passwords (see sample usage 1)
A sample source code for masking passwords can be downloaded from the following link:

http://www.eng.auburn.edu/~xqin/courses/comp2710/passwordc.cpp

After modify this sample C++ code, you can integrate this code into your teller terminal system.

3.2. System administration management
3.2.1 Add branch staffs to the system: (see sample usage 2.1)
A system administrator can add new user (branch staff employee) to the teller terminal system. When a new user is added into the teller system by the system administrator,

user name and password of the new staff employee must be initialized. **Empty values of the use name and password are not acceptable.**

3.2.2 Display branch staffs: (see sample usage 2.2)

User names and roles of all the branch staff employees including administrators are displayed. Before display a list of staffs, the total number of staffs must be displayed. You can simply follow the format below:

```
There are 3 users in the system.
1. User Name: admin          Role: System Administrator
2. User Name: abc0002  Role: System Administrator
3. User Name: acm0008  Role: Branch Staff
Press any key to continue...
```

3.2.3 Delete a branch staff: (see sample usage 2.3)

To delete a branch staff employee from the teller terminal system, a system administrator must login to the system and use the system administration menu. The system administrator must use a user name to identify which staff should be deleted. After the administrator enters the staff's user name, the administrator needs to confirm this delete action. Sample user interface is given below:

```
Delete a user - User Name: abc0005
1) Confirm
2) Cancel
Please choose an option: 1
```

The system will first search the list of staffs for the staff to be deleted. If the staff's information is not in the system, a warning message (see the sample warning message below) will pop up.

```
Warning - User acm0006 is not in the system. No user is deleted!
```

3.3. Client and Account Management

3.3.1 Add a client (see sample usage 4.2)

If "Add a client" is selected, new client's name, address, social security number, employer, and annual income must be entered. For simplicity, we assume that client names are unique, meaning that we can use client names as client identifiers.

3.3.2 Add an account (see sample usage 4.3)

If "Add an account" is selected, client's name must be entered first. If the client is not found in the system, an error message will pop up. If the client is in the system, then the branch staff has to enter account number, account type, and account balance.

3.3.3 Edit client information (see sample usage 4.4)

If "Edit Client Information" is selected, client's name must be entered. If the client is not found in the system, an error message will pop up. If the client is in the system, then the branch staff can edit the client's information, including address, social security number, employer, and annual income. Before updating the client's information, existing client

information is displayed. If the branch staff selects “Confirm”, the client information can be updated. **Note**: In this prototype, clients can not be deleted. We also assume that client names should not be changed, because we use client names IDs.

3.3.4 Manage an account (see sample usage 4.5)

If “Manage an account” is selected, an account number will be entered. If the account does not exist in the system, an error message will pop up. The format of the error message is given below:

Error – Account <Account_Number> is not in the system!

After the account is chosen, the staff can either deposit or withdraw funds by choosing a menu option. Thus, if the account exists in the system, the following menu will appear:

Manage account <Account_Number> for <Client Name> ...

1) Deposit
2) Withdraw
3) Cancel

3.3.5 Save client and account information (see sample usage 4.6)

If “Save Client and Account Information” is selected, then the Teller Terminal System writes all current accounts to a file called “account-info” and writes all client information to a file called “client-info”. Each time the teller terminal is started, the “account-info” and “client-info” files containing the account and client information are loaded and initialized. For simplicity, the names of the two files are pre-specified. Branch staff employees are not authorized to change the file names.

3.4. Press any key to continue... (see all the sample-usage cases where it is applicable)

The pause command in the teller terminal requests the user to "Press any key to continue…" A sample source code of the implementation for the pause command can be downloaded from the following link:

http://www.eng.auburn.edu/~xqin/courses/comp2710/continue.cpp

After modify this sample C++ code, you can integrate this pause commend into your teller terminal system.

3.5. System Quit

This should safely terminate the system. Information of a list of system administrators and branch staffs (i.e., user names, passwords, roles) must be saved to a file called “staff”

4. Programming Environment

Write a short program in C++. Compile and run it using the g++ compiler on a Linux box (either in Shop 3, computer labs in Shelby, your home Linux machine, a Linux box on a virtual machine, or using an emulator like Cygwin).

5. Requirements

Write a program called <username>-project1.cpp (for example, mine would read

5.1 Vectors

Your underlying data structure for lists of branch staff employees, clients, and accounts will be standard Vectors (**no arrays**). Thus, good classes to be considered include StaffVector; ClientVector, and AccountVector.

5.2 Operator Overloading

You must at a minimum overload the comparison operator "= =" in the Password class to help you to compare the new password with the one stored in the password database.

5.3. Reuse well-written classes

You are welcome to reuse well-written classes from earlier COMP 2710 projects (but indicate where they came from).

5.4 No Global variables

You may not use global variables or global functions – all your data/operations must be contained within your objects.

5.5 Usability concerns and error-checking

Your program's output does not necessarily need to match the style of the sample output, but the contents should be understandable and no functionality should be lost. You should appropriately prompt your user and assume that they only have basic knowledge of the system.

You should provide enough error-checking that a moderately informed user will not crash your program. This should be discovered through your unit-testing of functions.

5.6 Comments

Follow the comment standard posted on the web or some alternate, approved standard.

6. Deliverables

Write a program called <username>-project1.cpp (for example, mine would read "xzq0001-project1.cpp").

Use comments to provide a heading at the top of your code containing your name, Auburn Userid, and filename. Please describe any help or sources that you used (as per the syllabus).

You will lose points if you: do not use the specific program file name, or do not have a

comment block on **EVERY** program you hand in.

Please submit your source code through the Canvas system (e-mail submission will **not** be accepted). You need to submit the following two files:

- **<username>-project1.cpp:** this file contains a class diagram and a system sequence diagram.
- **<username>-project1-.pdf or**
- **<username>-project1-.doc:** your pdf or doc file should contain the testing results.

For example, mine would read "xzq0001-project1.cpp" and "xzq0001-project1.pdf").

7. Grading Criteria

7.1 (45 points) Implementation

- (2 points) submitted file names
- (3 points) Use comments to provide a heading at the top of your code containing your name, Auburn Userid, and filename. Please describe any help or sources that you used (as per the syllabus).
- (5 points) a single menu class.
- (5 points) operator overloading
- (5 points) vectors
- (5 points) coding style
- (5 points) authentication
- (5 points) Administrator management
- (5 points) Client management
- (5 points) Account management

7.2 (15 points) Test Results

- (3 points) Authentication
- (4 points) Administrator management
- (4 points) Client management
- (4 points) Account management

8. Late Submission Penalty

- Twenty percent (20%) penalty per day for late submission. For example, an assignment submitted after the deadline but up to 1 day (24 hours) late can achieve a maximum of 80% of points allocated for the assignment. An assignment submitted after the deadline but up to 2 days (48 hours) late can achieve a maximum of 60% of points allocated for the assignment.
- Assignment submitted more than 3 days (72 hours) after the deadline will not be graded.

9. Rebuttal period

- You will be given a period of 72 hours to read and respond to the comments and grades of your homework or project assignment. The TA may use this opportunity to address any concern and question you have. The TA also may ask for additional information from you regarding your homework or project.

10. Hints

- Start early, you have a good deal of time but you may need it to debug your program. Although the following timeline is not mandated, it is a suggestion of milestone:
    - 1/4 time: Finish process planning. Implement infrastructure and be able to load the files.
    - 2/4 time: Implement the basic classes as well as vectors for lists of branch staff employees, clients, and accounts sort.
    - 3/4 time: Implement system login, administration management, client/account management
    - 4/4 time: Complete interfaces, appropriate testing information, complete testing and finish final documentation.
- If you bring your documents by early, I will give you comments and help point you in the right direction on this project.
- Note that this teller terminal system is not secure, because passwords are saved in plaintext in a file. In a future project, all the passwords will be encrypted.