Bear bank documentation

Missouri State University

Group Members:

Jaron Ritter – [Ritter01@live.missouristate.edu](mailto:Ritter01@live.missouristate.edu), Fall 2020

Cole Kassing – [Cole115@live.missouristate.edu](mailto:Cole115@live.missouristate.edu), Fall 2020

Zoheb Ullah Sohel – [ms98s@MissouriState.edu](mailto:ms98s@MissouriState.edu), Fall 2020

Table of Contents

1. Features (2-4)
2. Data Structures Used (5-6)
3. Data Structures Instances (7)
4. Complexity Analysis and Justification (8-9)
5. Repository Information and Setup (10)

Features

Admin Users:

ID – Unique identifier for the admin. There is only 1 admin.

Login -Will have a login used for verification; This is changeable. Retrievable; Encrypted

Password – Password for login verification. This is changeable but not retrievable. Encrypted

Enable/Disable Accounts – Admin can set the accounts to disabled and enabled. If disabled, the account can no longer be used. Can be set back to enabled.

Modify Account Specifics- Admin can change the interest rate and term length of specific accounts

Create/Delete Accounts – Can created a new account. The delete will just disable the account, so the transaction history can still be retrieved.

Retrieve Login – Can retrieve the login username for any user

Change Password – Can change the password for any user, cannot see the password

Bank Official Users

ID – Unique identifier for each Bank Official

Login – Used for verification; Encrypted; Changeable

Password – Used for verification; Encrypted;

Open/Close Accounts – This user can create new accounts for a specified account holder

Account Information Retrieval – Can get Interest Rate, Term Length, balance, and account number for a specified account, given the account holder

Deposit/Withdraw – Can deposit and withdraw from any account, if the account holder verifies the transaction, this will create a transaction and save it to the transaction history

Search Accounts by Account Number/Name/ Phone Number – If the search is done by name or phone number it will return an account holder. If done by account number, it will return an account.

Account Holder Users

ID – Unique identifier for each Account Holder

Login – Used for verification; Changeable

Password – Used for verification

Phone Number – Used for searching

Address – Additional Account Holder information

Can have several accounts – Each account holder can have several accounts of varying type

Change Password – An account holder can change their password, if the login

first

Check accounts – Can check the transaction history and can see the balance of their account/accounts

Check Transaction History by date – Can see all the transactions from a specified month

Accounts

Account Number – Each Account has a unique Account Number

Type – Can be CD: fixed-rate certificate of deposits, C: Checking, or S: Savings

Creation Time – This is the date the account was created.

Closing Time – This is the date the account was closed.

Balance – Returns the balance of the account

Interest Rate – The set interest rate of the account, can be changed by Admin or Bank Official

Term Length – How long it takes the account to mature

Service Charge – applied monthly to certain accounts

Transactions

Date – date of transaction

Action – Can be withdraw, deposit, open, or closed

Amount – The amount the transaction was for.

Official – The official that was in charge of the transaction.

Data Structures Used

**Vector** – We used this fairly often throughout the project because we were very familiar with it, we needed something that could be accessed with an index accessor but we needed something that could grow in size, and it was provided by the standard template library.

**Array** – This was used only in one place. In that particular situation a fixed size container was right. The elements needed to be integer and in a related space.

String – This was used everywhere in the assignment. It was provided in the standard template library, and shortened our code considering the alternative

**Account** – This was the first of the user defined data type used in this project. It was created to store several things in relation to a particular account. It has the data members to store things like the activation status, interest rate, term length, account number, balance, service fee, account type, and time stamps for time created, closed, last time the account was accessed, maturity, and it has a vector of transaction objects, which hold the record of the transaction. There are many methods for this data type, but most of them are getters ands setters, with exceptions like calculate interest rate and print transactions.

**AccountHolder** – This was used to store things to do with a specific account holder. It has data members like first name, last name, phone number, password, login, id, and a vector of account objects. It has several methods but most of them are getters and setters, except for creating different accounts and printing the information about the accounts.

**BankOfficial –** This class is fairly similar to the AccountHolder class with a few notable exceptions. It does not have a vector as a data member. It also does not have a phone number or an address data member. Its members are also a little more than just getters and setters. It has an open account method that will open an account for a specified account holder. It has the ability to close an account. It can also make withdraws and deposits for a specified user. It can also search for account holders given a last name, a phone number, or login. It can search for accounts given an account number.

**Admin –** This class will store the information for an admin user. There is only one admin user allowed to exist at a time. There is no way to delete or add an admin user. This class is similar to the BankOfficial data type. It has the data members of first name, last name, login, password and id. It has the capability to enable or disable the bank officials. It can get the login for any user, including itself. It can also change the interest rate and term length for a specific account given the account number, and the accountholder.

**Transaction** – This is a very simple data type. It has a date member, which is an array of integers, with the 0th index being the month, the 1st index being the day and 2nd index being the year. It has data members action, amount, and official. Official is referring to the official that oversaw the transaction. Originally this class was just going to be a string, but it was changed to its own data type late into development.

**Data Structures Instances**

|  |  |  |  |
| --- | --- | --- | --- |
| Vector | Array | String | Account |
| Used to store things like all the account holders and bank officials | Used to store exclusively the dates in the transaction object | Used in every user defined object for things like name, login, and password | User defined data type. Has account number, and stores all the information about a specific account |
| Used in main.cpp, AccountHolder.h, account.h, | Used in Transaction.h | Used in Account.h, main.cpp, AccountHolder.h, BankOfficial.h, Admin.h, Transaction.h | Used in main.cpp, AccountHolder.h, BankOfficial.h, Admin.h |

|  |  |  |  |
| --- | --- | --- | --- |
| AccountHolder | BankOfficial | Admin | Transaction |
| User defined data, Stores everything that has to do with a specific account holder | User defined data, Stores everything that has to do with a specific Bank Official | Only 1 admin is created. Stores things that has to do with the admin | Holds a record of a transaction. Has things like date and actions taken |
| Used in main.cpp, BankOfficial.h,  Admin.h | Used in main.cpp,  Admin.h | Used in main.cpp,  Admin.h | Used in main.cpp, AccountHolder, account.h, main.cpp |

**Complexity and Justification**

Vector – Index accessing: O(1), inserting or removing elements at the end: O(1), insert or

removal at other indexes: O(n).

Justification: For our purposes everything would be O(1). Included in the standard template library.

Array – Index accessing: O(1), inserting or removing element: O(1).

Justification: Didn’t need anything more. Memory efficient.

Account – Getters and setters: O(1), Withdraw and Deposit: O(1), Calculate Interest Rate: O(n);

Calculate monthly term: O(n); Print Transactions: O(n);

Justification: This was the most efficient way to store the data regarding an account.

It is fairly efficient.

AccountHolder – Getters and setters: O(1), Withdraw and Deposit: O(n); getType: O(n);

getAccountBalance: O(n); Creating accounts: O(n); printAccounts: O(n);

printAccountInfo: O(n); Everything else: O(1);

Justification: This was the best and most clear way to store account holder information.

Nothing is more complex than O(n), making it decently efficient.

BankOfficial – OpenAccount: O(n); CloseAccount O(n); openAccountD: O(n); search: O(),

for accounts; search: O(n), for accountholders; Everything else O(1);

Justification: This was the most straight forward way to organize the different users. The

Search for accounts function is not very efficient, but everything else is.

Admin – Modify: O(n); Everything else is O(1);

Justification: This is inline with the way we set up all the other users. It’s efficient. It

is simple.

Transaction – Everything: O(1);

Justification: it has not methods, so it is purely for storage and accessing. It’s very

simple. It was the best way to store the transaction record and keep all of the transactions separate and easily accessible.

**Repository Information and Setup**

**Repository Link -** [**https://github.com/jaronritter01/FinalProjectBankAccount.git**](https://github.com/jaronritter01/FinalProjectBankAccount.git)

**Setup –** 1. Clone the git repo

2. Make sure all the files are in the same folder.

3. Build/Compile the project

4. Run the project.

5. There will be several menus in the program, but they are simple and easy to follow.

**Bonus Justification**

I believe this submission deserves bonus credit because of the obstacles that we had to overcome during development. We had a group member who got very ill and was not able to contribute to the project. We also had quite the communication barrier. One of our group mates lives in a time zone with a 12 hour difference. This made setting up meetings and having time to communicate, very difficult. I feel as though our submission met the goals, despite the challenges that we faced.