CS 499 Senior Project
Project Assignment

Bank Management Utility

Rationale

Banks must maintain information on a variety of accounts for any number of clients and provide on-line access to accounts. Accounts must be easily accessible to on-line customers with a maximum of security features. Account information must also be readily available to bank employees to assist customers.

The objective of this project is to develop a system which will maintain all information on user accounts. This must be a web based application that allows for customers, tellers and administrators to access.

This is a four (4) person project.

Roles

This system shall support the following roles:

Customer - Shall be able to create an account and have the ability to enter and modify their personal data. Personal data does include the following: Name, address, phone, cell phone number, social security number (Tax ID number), etc. The system is to support multiple customer accounts.

Teller - Shall be able to review customer data only. Shall be able to create, modify and delete accounts. Account information may include the following: Account number, 1099-INT information, set login ID, set and reset password. The system is to support multiple teller accounts.

Administrator - Create, modify and delete Teller accounts. Review any teller or customer data in the system. There is only one administrator account.

Features

The features listed below shall be included in the software.

1. Users of the system must have a username and password to log in to the system.
2. Users accounts shall be one of three types: Administrator, Teller, and Customer.
3. The application shall meet the following criteria:
    a. Bank employees may log on to the system as either Administrator or Teller.
    b. The application shall be able to access a database* of all account information.
4. Tellers may perform the following functions:
    a. Create new accounts for customers. These accounts must include at least the following:
        1. Regular checking account.
        2. Savings account which will earn interest at a given percent per month.
        3. Money market account which will earn interest at a given percent per month.
        4. Home mortgage loan account with a given amount due each month.
        5. Credit card account with a given amount of interest due on unpaid balance each month.
    b. Delete accounts for customers. The system shall only allow accounts to be deleted if there is a zero balance in the account.

5. Tellers may perform the following functions:
    a. Record customer deposits to any of a customer's accounts.
    b. Make transfers from one account to another for a single customer.
    c. Record withdrawals from any of a customer's accounts.
6. Customers may perform the following functions:
    a. View a report listing all deposits and withdrawals from any of their accounts.
    b. Transfer funds from any of their accounts to other account. Note some transfers should not be allowed. One example, transferring from a loan account to any other account. The development team is to define which account transfers are not allowed and if a customer attempts to do this transfer the system should explain why the transfer is not allowed.
    c. Customers may pay bills on-line by designating the payee name and address, the amount, the date the bill is due, and the account to pay the bill from.
7. Security of customer information shall be a primary concern.
8. The entire system shall have appropriate Graphical User Interfaces.

*Note: Data does not have to be stored in a "database". This is an implementation or design decision that needs to be made by the team.

Constraints

The application is to be web or browser based and can run under any browser (Chrome, Edge, Firefox, etc.).

The Program must be robust -- Assume most users (customers) are not computer-knowledgeable. The system should be able to recover from errors, particularly input mistakes.

The project description and requirements are vague on purpose. The team is expected to derive additional and more specific requirements that will make this Banking System more usable and more like current online banking systems. For example, how should the system handle and calculate interest rates for credit accounts and money market accounts? What are the security requirements?

Student Feedback from previous semesters.

The following information was provided by students at the end of the term and should be considered as lessons learned. Some of this information may or may not be useful because the project requirements may have changed from previous terms.

Spring 2022

Design the backend before you start thinking about the frontend. Dealing with money, comes with a lot of rounding issues, so be wary of how you handle transactions and balances. Docker is going to be extremely helpful.

Keep in mind that the frontend testing will often require backend work to be done, so having backend done sooner is better.

Carefully look over the requirements and make a plan. Try not to have to make changes later. Some things were confusing for us, make sure you take advantage of asking the professor for clarifications. Make sure you have and keep good communication between your front-end and back-end so that you are working towards the same goals. Get your development environment set up as soon as possible so you can begin real work on the project.

Organization is the best tool to help do this project. Within your own workspace and with your team.

Spring 2024

Do research on about various aspects of banking early on. There are many aspects to banking that most college age kids haven't experienced or understand, such as how home mortgage accounts work, or the regulation behind money market accounts. Also, test your work as you make it. It can be difficult to identify problems if you make large chunks of progress without checking the smaller parts first.

We used IntelliJ as our environment for our frontend and backend, but was a little problematic at times. For example, it would sometimes not display the current version control or let me change branches. I had to restart my laptop to get it working again. However, IntelliJ has many great tools to where it was worth the occasional headache.

Make sure that the tools used to connect the frontend and backend work well together early. I would also recommend using a proper database for security and longevity of data.

The API we started with was not compatible with the backend language, so do research on the best API tool to use with the language you plan to develop in.

If no one in the group is experienced in creating an API, learn it early and learn it well! Every API framework is very different and it can be pretty overwhelming and confusing, even just to setup. I also recommend learning to use a popular frontend framework, like Vue.js/Vuetify, since it helps make the website more user friendly.

Be careful that the API framework you use corresponds to the backend you want to use! I would not recommend Laravel, if you happen to find tutorials using that, unless you really want to use PHP has your backend language.

That the best way we found as a team was to split the work 50/50 with the front and backend. 2 people working on each. The reason I say this as the backend team was able to finish early and to have the time to learn the frontend process and help finish out the last pieces a lot easier than doing each piece as a group of 4.