

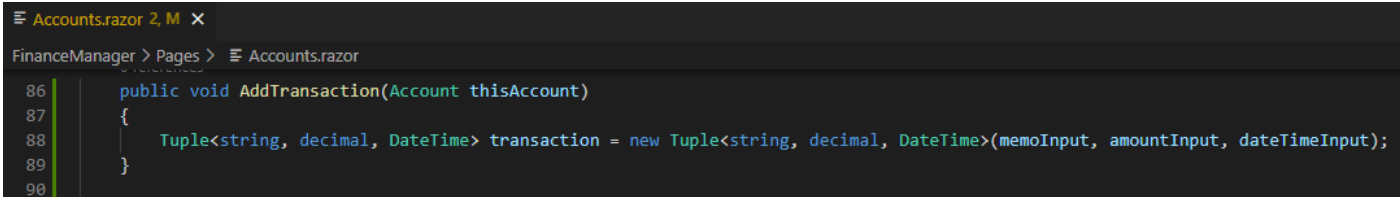
Sarah Martin - CS 1410 Final Project Check-in 3 - Apr. 9, 2022

Link to the repository:

<https://github.com/sarah-strawberries/CS-1410-final-project>

Feature Added: Tuple

This application will eventually have the option for a user to add a "transaction" entry for an account. These transactions will be stored as a `Tuple<string, decimal, DateTime>`. References to each transaction in an account will be stored in a transaction `List<Tuple<string, decimal, DateTime>>` in each account instance.

A screenshot of a code editor window showing the implementation of the `AddTransaction` method in a file named `Accounts.razor`. The code is as follows:

```
86 public void AddTransaction(Account thisAccount)
87 {
88     Tuple<string, decimal, DateTime> transaction = new Tuple<string, decimal, DateTime>(memoInput, amountInput, dateTimeInput);
89 }
90
```

For now, this is the extent of the (functional) tuple code.

How to use the application

So far, the main functionality of this application is that you can add a bank by filling in the fields on the home page. You can actually add multiple banks, and the most recent one you added will show up on the page (at least until you navigate away from the page). You cannot, however, add a bank with the same name as another bank. (Which is probably a good thing, since that could get confusing!)

Finance Manager [About](#)

Home

Welcome to your finance manager!

Fill in the fields below with the name of a bank and with a 9-digit routing number to add a bank.
(Hint: the bank name should be a minimum of 2 characters, and the routing number should not start with 0.)

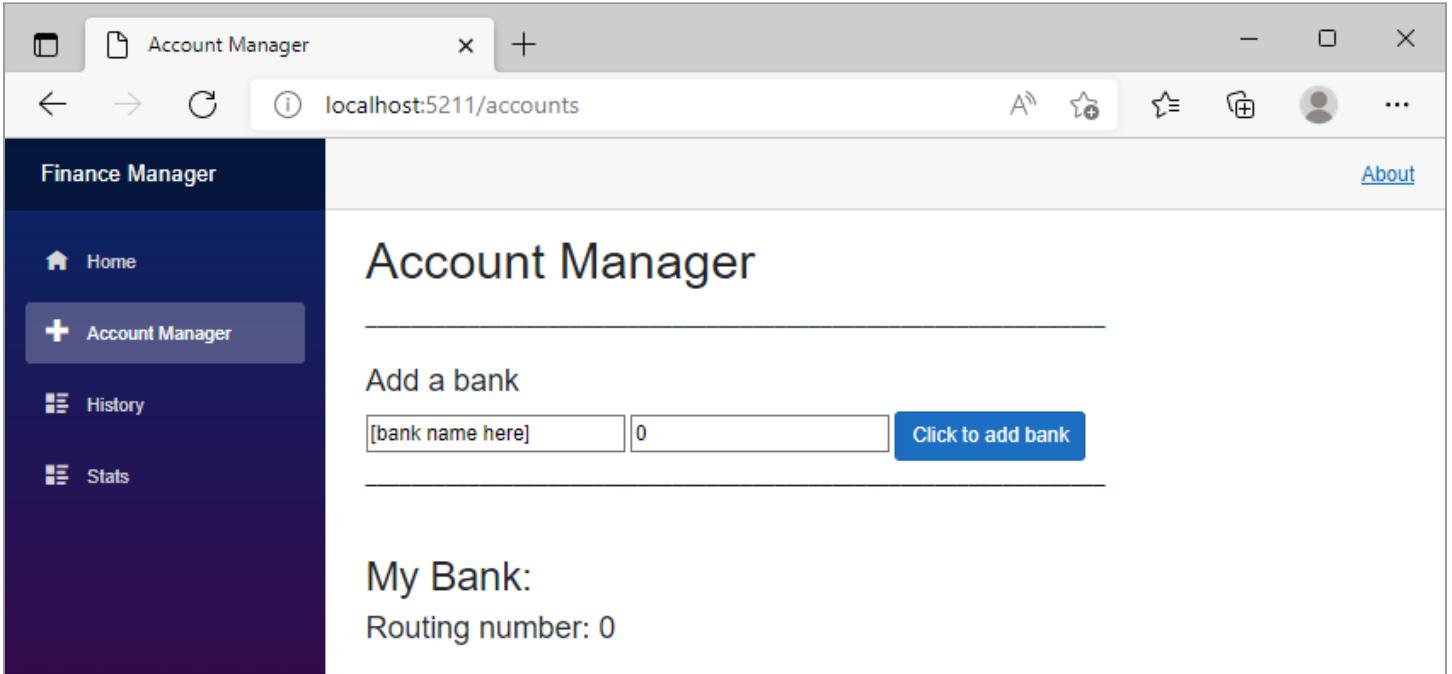
[bank name here] 0 Add a Bank

My Bank: bank name will show up here

Routing number: 0 routing number here

You can also add a bank from the Account Manager page (the formatting of which, in my opinion, is nicer). The application will remember what banks have already been added and not let you add a bank with the same name as

one you've already added.



The screenshot shows a web browser window with a single tab titled "Account Manager". The address bar displays "localhost:5211/accounts". The browser's toolbar includes back, forward, and refresh buttons, as well as icons for search, star, and user profile. The web application has a dark blue sidebar on the left with the title "Finance Manager" and a list of navigation items: "Home", "Account Manager" (highlighted with a plus icon), "History", and "Stats". The main content area has a light gray header with an "About" link. Below the header, the title "Account Manager" is displayed. A section titled "Add a bank" contains two input fields: the first is labeled "[bank name here]" and the second contains the value "0". A blue button labeled "Click to add bank" is positioned to the right of the second input field. Below this section, the text "My Bank:" is followed by "Routing number: 0".

Account Manager

Home

+ Account Manager

History

Stats

About

Account Manager

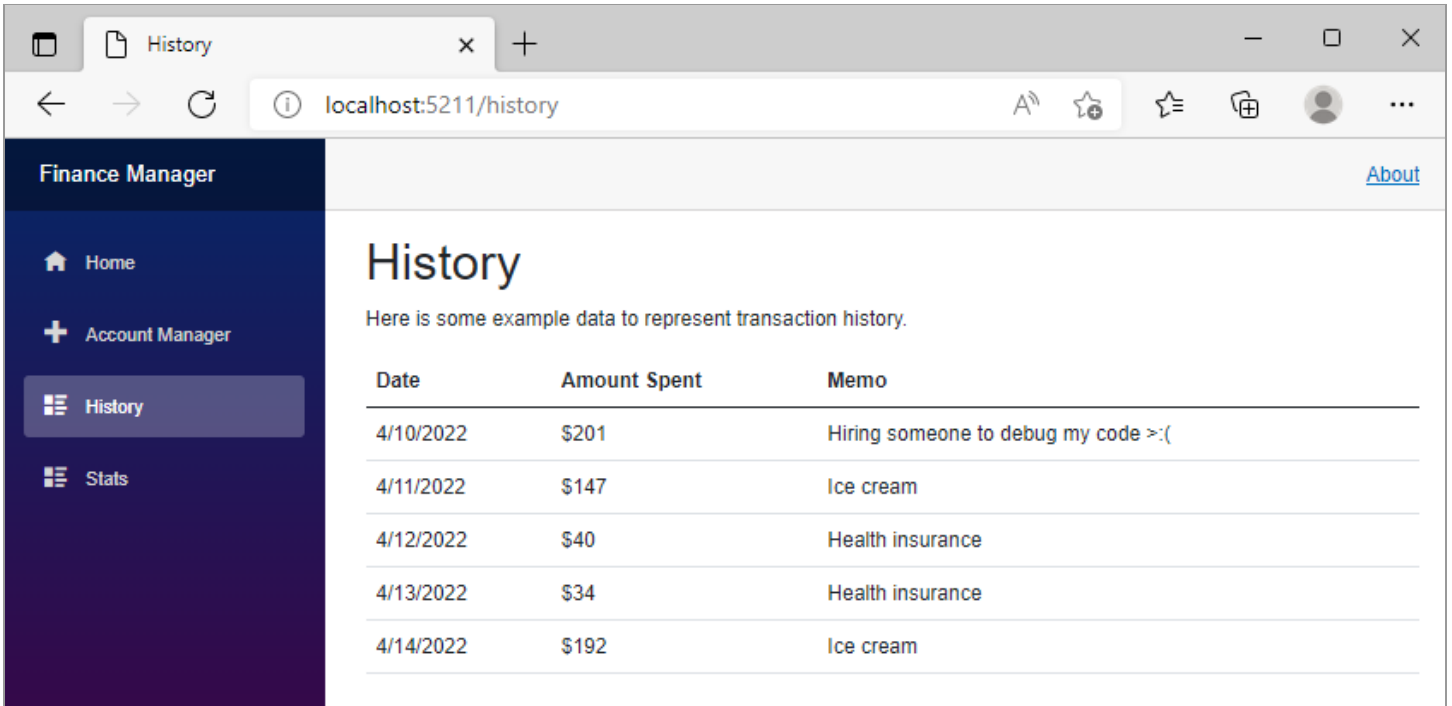
Add a bank

[Click to add bank](#)

My Bank:

Routing number: 0

Additionally, on both the History and the Stats pages, you can view an example list of purchases, which is a placeholder for information that will eventually be stored there to show your past spending and earning history. The list currently generates randomly, so it's kind of fun to refresh the page and see what's on the list. ("Oops, did I really spend that much on ice cream??")



The screenshot shows a web browser window with the address bar displaying `localhost:5211/history`. The application has a dark blue sidebar with the title "Finance Manager" and a navigation menu with options: Home, Account Manager, History (selected), and Stats. The main content area is titled "History" and contains a table of transaction history.

Date	Amount Spent	Memo
4/10/2022	\$201	Hiring someone to debug my code >:(
4/11/2022	\$147	Ice cream
4/12/2022	\$40	Health insurance
4/13/2022	\$34	Health insurance
4/14/2022	\$192	Ice cream

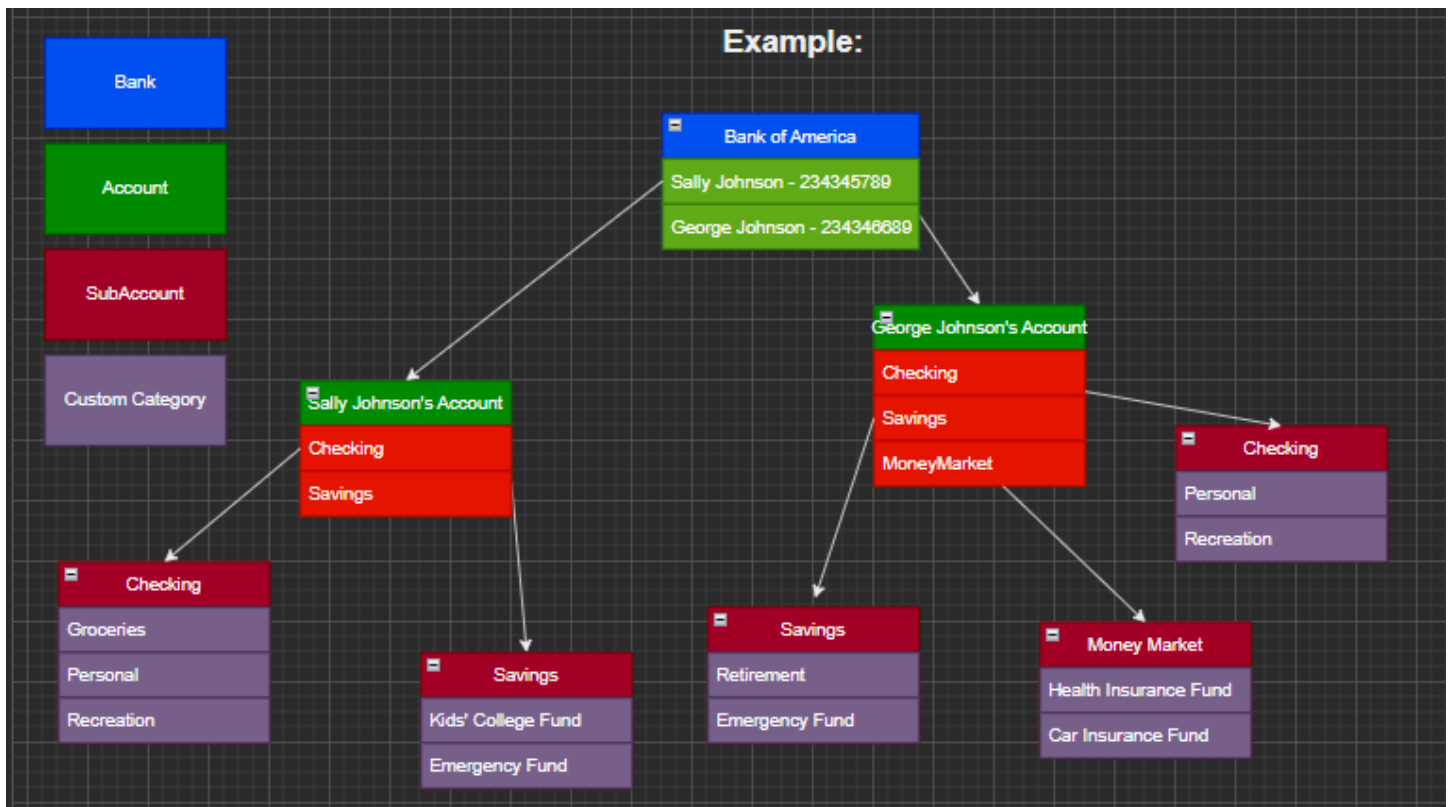
There is actually a lot more functionality built into the classlib than is currently being used in the Blazor app, since I haven't gotten it all implemented in the Blazor pages. For example, there are ways to save things to files, return the contents of dictionaries in nicely formatted ways if you want to print out a list of them, and to retrieve information from the dictionaries. The goal is just to get it to work in Blazor now.

Progress Report

Stumbling Blocks I've Overcome

One stumbling block I had to overcome was related to the Blazor pages. Since I wasn't very familiar with how the navigation between pages was set up and where I should go to change the title of each page, it took some time to look through different files and figure out which parts to change to get all my pages to have the correct names and get the navigation menu to show the right names for them.

Another stumbling block I faced was having to somewhat restructure how the application was set up, because after I had already coded a good portion of it, I realized that the way I had set up the class hierarchy with each account and sub-account being stored in a dictionary and certain of the object members being the keys, it wouldn't actually do what I wanted it to do. So then I had to rethink my conceptual model and map it all out until I figured out a way to do it that would work. The result of that was this diagram of the class hierarchy, which I am quite happy with.



Another problem I encountered was that I was getting unhandled exceptions thrown when I tried adding a bank more than once. I figured out it was because the bank key was the name of the bank, so I couldn't add it to the dictionary inside of the constructor like I tried to do, and thus it was throwing an exception because of that.

Problems I'm Currently Facing

I decided that I wanted to change the favicon next to the title of the webpage on the browser tab. But I haven't yet figured out where in the code I need to put the file path to make that work. So for now, it just has a blank favicon.

Other than that, surprisingly I haven't encountered many errors that weren't easy to fix.

;