Changes Made in this PR:

I have created a scoring system that allows the bank to assign a score to each user based on their utilization of the bank. The algorithm developed to calculate the score takes into consideration the percentage of transactions that are deposits, the number of cryptocurrencies the user has purchased through the bank, and any instances of the user overdrafting their account. More details regarding the specifics of the algorithm are included below.

`java/net/testudobank/MvcController.java`

- Created a get mapping and post mapping for the score for each user so that the score can be accessed elsewhere in the code without data leaks.
- Updated the interest history log on the front end to include the interest rate being applied to the users account. This is useful because the interest rate changes based on the users score range.

`src/main/java/net/testudobank/TestudoBankRepository.java`

- Modified the current interest feature such that the new interest rate is being applied as opposed to the old, static value (1.015).
- Added a score calculation function that takes in the users ID and calculates the score based on several factors. The score ranges from 0-10 with 10 being the highest possible.
    - 80% of the score is determined by the percentage of deposits out of total transactions that the user has made (I chose this because the bank benefits when users put more money into it, and this incentivizes the users to deposit more)
    - 20% of the score is determined by the user’s crypto balance in both ETH and SOL. If the user has purchased (and maintained a positive balance) in both, then the full 20% is received. If the user only has one, then only 10% is received. No points are added to the score if the user doesn’t have any crypto. I chose to use this as another contributing factor to the score because I believe that it

    will incentivize the users to purchase their crypto through the bank.

    - Finally, the score decreases based on the presence of an overdraft balance. If the user does have an active balance in the overdraft logs, then the user will lose 10% of their score.

- Added a interest calculation function that calculates the users score and then assigns them a corresponding interest rate. Below I've attached the interest rates by score range.
    -

Score Range	Interest Rate
> 1	.001
1 <= score <=3	.005
3 < score <= 6	.010
6 < score <= 8	.015
> 8	.02

`src/test/java/net/testudobank/User.java`

- Added a getter and setter for the score associated with the user

---

`src/test/java/net/testudobank/User.java`

- Updated the front end account_info page such that the score, as well as the users score breakdown is included. This allows the users to understand their score and the factors that are contributing.

`src/test/java/net/testudobank/tests/MvcControllerIntegTest.java`

- Added a simple score test that ensures that the basic functionality of the score calculation works when the customer makes deposits and is not in overdraft. This test then withdraws from the account such that the user goes into overdraft. The test then ensures that the score is decreased appropriately. Finally, the test guarantees that even with 5 deposits of over $20, interest is still not applied and the score does not improve while the user remains in overdraft.

- The next test I wrote tested the scoring when the user has purchased cryptocurrencies through testudobank. This test ensures that the score is improved by 10% (1 point out of 10) for each cryptocurrency they own (and have a positive balance with).

Future Changes to Make:

- Consider the length of time in overdraft in the score calculation such that being in overdraft for a longer period of time will effect one’s score more than a shorter period of time.
- Create a popup window that can explain the score breakdown as opposed to just having it displayed plainly on the front end. I tried to implement this but wasn’t able to complete it by the time of submission. In the future I think that this would be a good change to make to improve the overall UI experience.