Testudo Bank Overdraft Feature

Owner: Adithya Solai (adithyasolai7@gmail.com)

Problem Statement

Customers of Testudo Bank are currently limited to only withdrawing money that they have previously deposited. Customers would benefit from a Testudo Bank service that enables them to withdraw extra money on credit beyond their account balance (to pay off unexpected expenses, for example). This service would also create a new revenue stream for Testudo Bank because we can charge interest on loaned money.

Solution Requirements

- Customers should be able to withdraw up to **$1000** beyond their account balance on a credit loan from Testudo Bank.
- Customers should have a place on the TestudoBank application to pay back the credit loan.
- Customers should be charged a **2% interest fee** on any money loaned from Testudo Bank
- Testudo Bank should provide a way for both customers and the bank to audit a customer's re-payment history of all credit loans.

Proposed Solution

The proposed solution is an **overdraft** feature that allows customers to withdraw beyond their balance using the existing withdraw form and pay back the credit loan using the existing deposit form. Customers will also be provided a re-payment log with timestamps in the existing `account_info` page.

This solution was chosen because it re-uses existing pages in the application, and only requires slight changes to the underlying MySQL DB.

Pro/Con of all approaches considered:

Overdraft Approach

(Approach already described above.)

Pros:

- No new pages/forms are added. Existing pages and forms are minimally changed.
    - Keeps the customer experience the same as what they are used to. *(Some text needs to be added to existing pages to describe the overdraft feature, display the overdraft balance, and show a re-payment history.)*
    - Simplifies development since only the logic in existing Controller `Post` handlers needs to be modified.
- Only slight changes to the MySQL DB Schema need to be made (a new `OverdraftBalance` column in `Customers` table, and a new table to store re-payment logs).
- A **2-way door** for the Testudo Bank business. Since this approach mostly recycles existing components and keeps the customer experience largely the same, this change can easily be rolled back or re-factored in the future.

Cons:

- Might be unintuitive to some customers because most banks traditionally offer credit loans as part of a credit account separate from a debit account, and this approaches blends the credit loan feature with a debit account.
- This approach might be short-lived if TestudoBank moves to a credit/debit paradigm in the near future. This feature will conflict with the credit borrowing offered in a credit account, and might have to be deprecated.

Debit & Credit Account Approach

This approach re-classifies the existing customer accounts as **Debit** accounts (where customers can only withdraw as much money as they previously deposited). Customers will also be eligible to open a **Credit** account where the balance indicates how much the customer has borrowed from the bank and needs to pay back.

Pros:

- Follows the credit/debit paradigm used by most banks. More intuitive for customers familiar with other bank applications.
- Prepares the codebase for more robust credit account features in the future.
- Similar MySQL DB Schema changes.
    - A new table will be added to track credit account re-payment history (just like the overdraft approach).
    - A new `CreditBalance` column needs to be added to the `Customers` table to track the credit account's balance, and the existing `Balance` column should be re-named to `DebitBalance`.

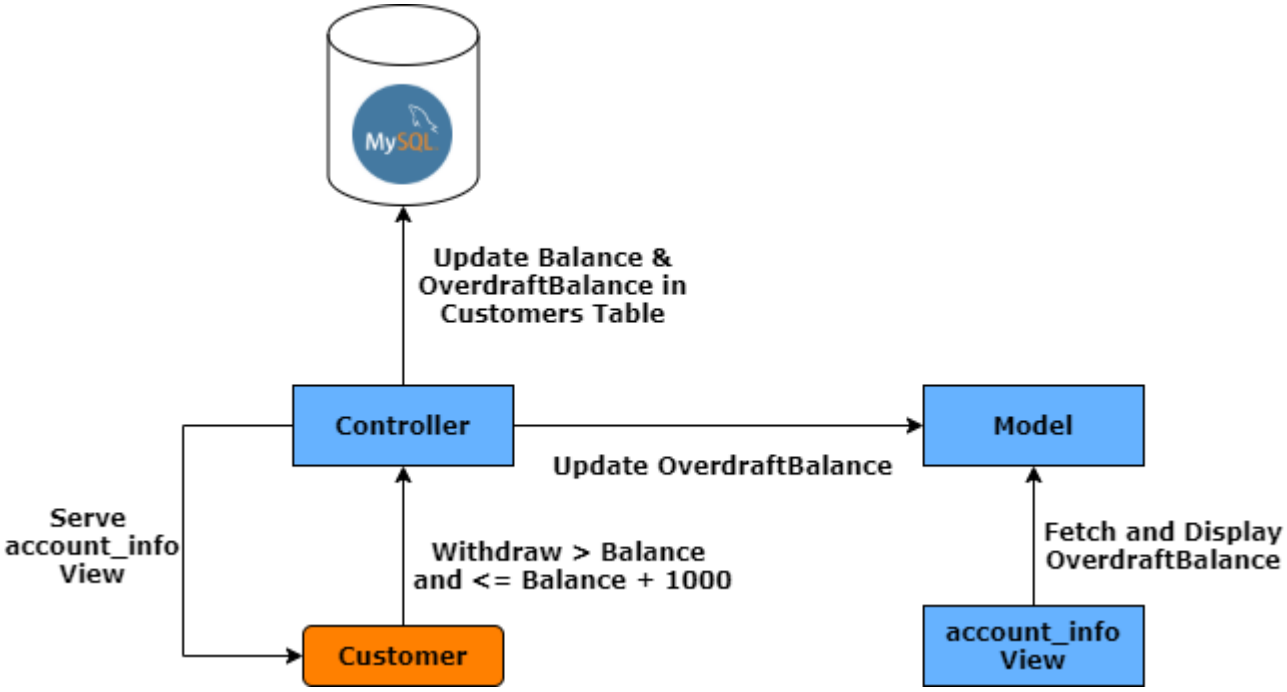
Cons:

- A credit/debit account separation is a **1-way door** for the TestudoBank business.
    - Once customers start using debit and credit accounts, we can't rollback that separation in the future.
    - The scope of a credit/debit account separation overshadows the original problem statement of trying to give customers a way to borrow money on credit. There should be more discussion and more customer pain-points that would be solved before going through this 1-way door.
- More complex to develop than the overdraft approach.
    - New pages may need to be made to show credit account info separately from the debit account.
    - A new form needs to be added to allow customers to transfer money from their debit account to pay off the credit account balance.

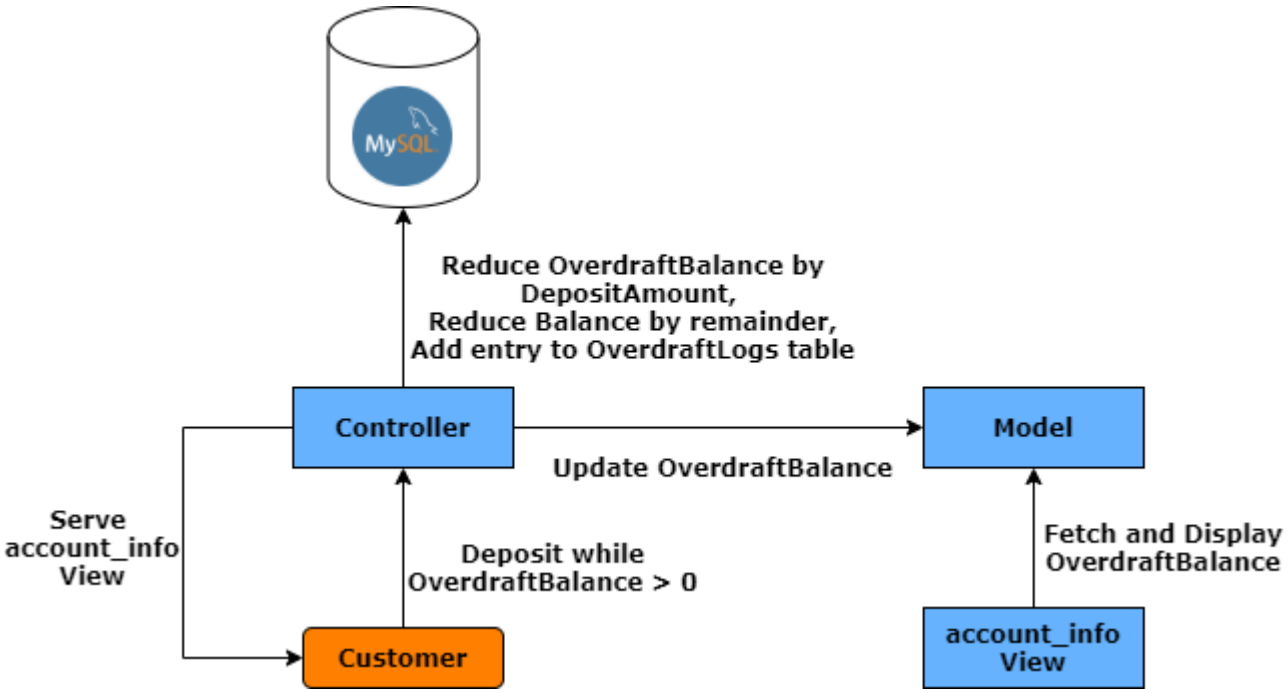
Technical Architecture (Overdraft Approach)

MVC Logic

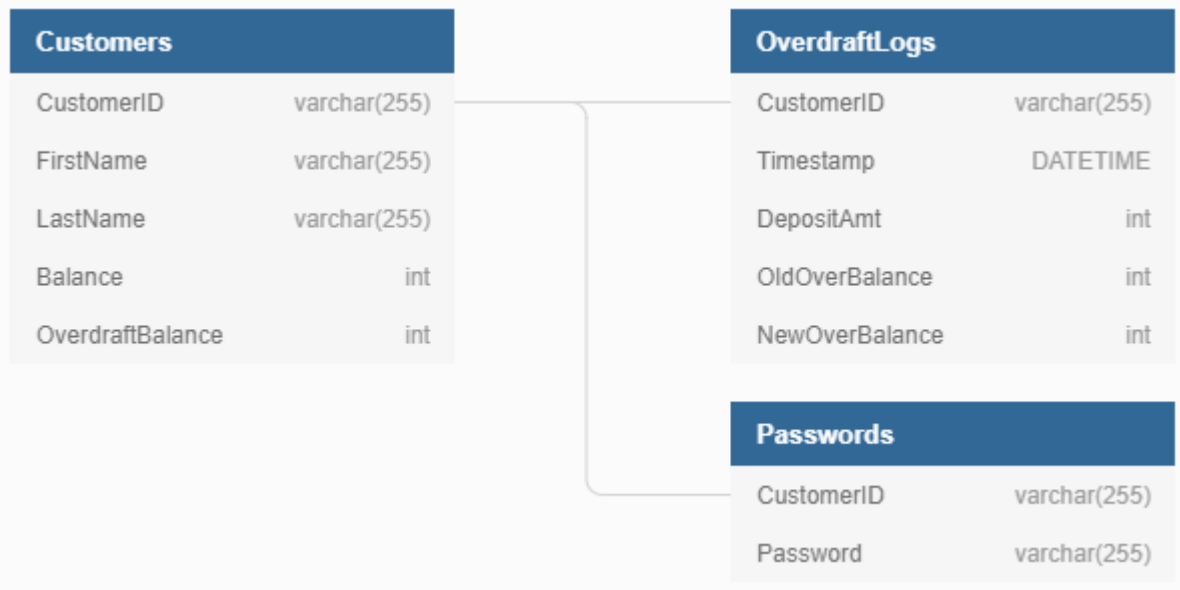
Customer Withdraws Beyond Balance



Customer Makes an Overdraft Repayment



MySQL DB Schema



- `CustomerID` is a primary key for the `Customers` table that can be used as a foreign key in the `OverdraftLogs` and `Passwords` table.
    - `CustomerID` is also a primary key in the `Passwords` table.
    - `CustomerID` is not a primary key in the `OverdraftLogs` table since each row in that table represents a single time a customer paid back some (or all) of their overdraft balance. The same `CustomerID` can be in numerous rows in the `OverdraftLogs` table since customers can pay back their overdraft balance over multiple payments.
- `OverdraftBalance` in the `Customers` table should only ever store values >= 0 and <= 1000. However, the customer actually owes `OverdraftBalance` * 1.02 due to the **2% interest rate**. This can be implemented by either storing all the interest rate logic in the Controller or adding another column in `Customers` called `OverBalInterest`. **Storing all interest rate logic in the Controller is preferred to save on DB memory.**