Testudo Bank Frontend Redesign and Reimplementation

Owners: Jason Cavanaugh

Problem Statement

Customers of Testudo Bank expect a clean and enjoyable user experience. The current implementation of Testudo Bank does not provide that. While the basic functionality is there, the user interface leaves much to be desired. We want customers of Testudo Bank to feel like they are using the banking software of the future, not the banking software of the early 2000s. If we are going to support such forward-looking ventures such as cryptocurrency, let our software also at least look the part.

This proposal is for an overhaul of the Testudo Bank frontend. The current frontend is implemented with `.jsp`. I propose that we redesign the frontend to use a modern UI framework. This will allow for cleaner, more maintainable frontend code and will thus contribute to the longevity of Testudo Bank.

Solution Requirements

- The frontend of the application should use a modern UI framework (such as React, Angular, Svelte, or Vue)
- The various pages of the application should be redesigned to be more visually appealing.
- Our new frontend code should strive to keep the state management of the frontend simple and straightforward. This will allow other engineers to onboard into our codebase more quickly and allow for changes to be made more quickly to our code.
- We should pick a framework that is well-established and has been used in other enterprise software.

- We should pick a framework that will allow us to hire developers more easily. For example, choosing React would mean that we have the largest pool of frontend developers to hire from, since React is so widely used. In contrast, choosing something like Yew, while perhaps bringing performance benefits, would not be easy to hire for and would not be as battle-tested.

Proposed Solution

The proposed solution would be to use some kind of modern frontend framework to reimplement the frontend of the application. Here are a few potential options.

React

Pros

- Our current team of developers is already familiar
- Many developers to hire from
- Widely used throughout the industry and in enterprise software
- Many, many, many libraries and packages to help us with whatever we need to build
- By far the most documentation and resources out of all the frameworks due to its popularity
- Fairly easy to pick up

Cons

- Less opinionated than other frameworks. It might be difficult to figure out what set of technologies is best suited to our particular use-case
- Less performant than other options
- Easier to build out bad patterns due to the fact that it is a library and not a framework
- Many bad React developers out there since it is so widely used
- useEffect can be difficult
- Will have to install many other packages

Angular

Pros

- More opinionated than React, meaning that it will try to steer us towards good design patterns
- Comes with many useful tools by default (e.g. TypeScript, a router, etc.)
- Also used widely in enterprise software

Cons

- More difficult to hire for than React
- Steep learning curve
- Fewer options about how to build due to its opinionated nature
- Many developers dislike it
- Our developers are unfamiliar with it

Svelte

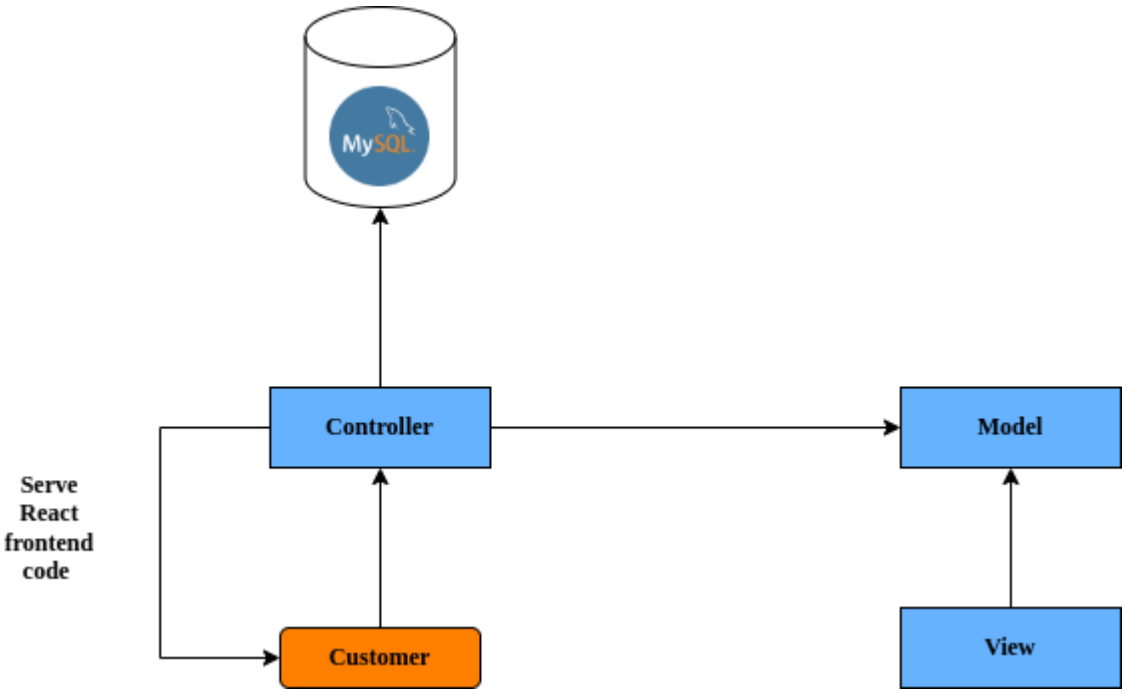
Pros

- The simplest option out of the three
- Developers love it
- Least amount of boilerplate
- Tiny bundle sizes
- Our developers are eager to use it on a project

Cons

- Not nearly as widely used as Angular and React, so it will be harder to hire for
- Not used in enterprise nearly as much as Angular and React
- Not as much OSS software built for it. Fewer libraries and packages to choose from
- Smaller community
- Our developers are unfamiliar with it

Flowchart Diagram


MySQL
Controller
Model
Serve
React
frontend
code
Customer
View

DB Schema


OverdraftLogs
CustomerID varchar(255)
Timestamp DATETIME
DepositAmt int
OldOverBalance int
NewOverBalance int
Passwords
CustomerID varchar(255)
Password varchar(255)
TransactionHistory
CustomerID varchar(255)
Timestamp DATETIME
Action varchar(255)
Amount int
Customers
CustomerID varchar(255)
FirstName varchar(255)
LastName varchar(255)
Balance int
OverdraftBalance int
NumFraudReversals int
TransferHistory
TransferFrom varchar(255)
TransferTo varchar(255)
Timestamp DATETIME
Amount int
CryptoHoldings
CustomerID varchar(255)
CryptoName varchar(255)
CryptoAmount float
CryptoHistory
CustomerID varchar(255)
Timestamp DATETIME
Action varchar(255)
CryptoName varchar(255)
CryptoAmount float
1
1
1
1
1
1
1
1

The db schema would not change for this particular feature implementation because this is a purely frontend feature.