**Video 1 – Front End Architecture**

Here is the diagram of the main components of my application. The front-end is developed with React components and coded using JavaScript and HTML. The database holding the user account info is hosted on MongoDB. The back end of this project uses node for development and Firebase for authentication.

My front-end is a basic professional look with mostly neutral colors and a little bit of imagery displayed on each card. The untrustworthy bit is a reference to it being a Bad Bank, in that it has no security. The navigation bar was created with the help of Bootstrap styling, but I also have my own CSS folder to achieve the overall look.

All the front-end components are housed in the public folder. Each menu item has its own JavaScript file for a better organization of the code. Not only is it organized, but it is also easy to navigate, and keeping components separate makes for easier troubleshooting.

Each page is displayed with a different card color that still manages to work well with the base look of the site. The withdraw page is displayed on a red card to represent decreasing funds while the deposit card is green to represent increasing funds.

The original bank image is displayed on the home page, but I also managed to incorporate it into the navigation bar as well as in the tab header.

I used Firebase for authentication because I found it is a tool I can easily use and incorporate into my application. It is used to log users into their account as well as create new accounts that are submitted in the create account form. Firebase made this process smooth since I have used it before during this course. Also, it can be incorporate into JavaScript code effortlessly. I don’t recall having any challenges to overcome with authentication. I think it’s the one section of this project that presented no issues.

**Video 2 – Database and API**

Mongo is the tool I used for the database in my application. It stores each user’s name, email, password, and balance. When someone creates an account, their information is automatically stored in this database along with a default balance of zero. Balance updates via deposit and withdrawals are automatically sent to the database. The DAL file I have acts as the communicator between the code and the database. The JavaScript code in the application page refers to the DAL code, which contains code that can be read by Mongo.

The API is used to communicate between the front-end and the database. The API is what is used to get data from and post data to the database. My application uses a RESTful API. GraphQL is a more advanced form of REST designed to make data retrieval more specific to allow get and post to occur without extra data that tends to come with REST. Because my app is small with very little data, REST is the better format.

**Video 3 – Deployment Additional Features, App Demonstration, and Reflection**