Cole Cassidy

Iota

CS 489 Thesis Project

Changes to the design plan

The last two weeks of development have yielded significant software design changes. The backend is entirely built using Node.JS, express, and the MongoDB driver. The actual Iota logic has been separated into a standalone library. This decision was made to escape the clutter of a single JS document and allow for the API and logic to be developed somewhat independently. MongoDB has been retained as the database as it has proven incredibly reliable and high speed. The design for the front-end has changed drastically. The original plan was to develop the front end using vanilla JavaScript. The issue with vanilla JavaScript is the overwhelming scope of developing from scratch.

State of the Back End

Progress on the back end has been slow but steady. The most tedious part of developing the back end has been integrating the logic with the database server. While the selected frameworks are incredibly useful, they leave much structural work to be done. However, as the essential structures get built development accelerates. The largest portion of the backend work tends to be supporting functions. Functions such as parsers, validators, authenticators, and database connections take a long time to build and test. Once these supporting functions have been built developing other functions becomes easier. Making a singular structured function for reading/writing to the database has made it easier to develop the actual API as large sections of code do not have to be repeated and tested each time.

Other than the challenge of developing the structural work, learning true asynchronous programming has been difficult. Asynchronous functions are functions that allow developers to create the illusion of multitasking in their applications. JavaScript is not truly multithreaded, but asynchronous programming allows developers to simulate this, by more directly controlling the execution order of their functions and their associated code.

In the example below “listAllUsers” is an asynchronous function. “Async” indicates that the function is of an asynchronous nature, and thus behaves differently than a standard function. Most importantly, it provides the developer with access to the “await” symbol. This stops the function from proceeding, until the following function call completes. In the example below the code will not continue to execute until “mongo.db(“iota_testing”)…… has completed execution, and assigned a value to “doc”. This is essential as otherwise the code would continue to execute instead of waiting for the database to respond and would return a null value from “doc”.

```
const listAllUsers = async () => {
    try {
        const doc = await mongodb.db("iota_testing").collection("users").find().toArray();
        return doc;
    } catch(e) {
        console.error(e);
        return ERRUnkown;
    }
}
```

A note on security

For the time being, implementing security has been put on hold. The intention behind the research project is to create a progressive web application to be used by organizations. Fortunately, by designing the application API first and by designing every API call as a function, it shall be easy to implement security in the future. Token validation can be implemented in a separate function that is called whenever an API function is called, and this will successfully secure the API.

State of the Front End

The front end has proven to be one of the more challenging portions of the overall project. The original intention was to design the front end completely from scratch using vanilla JavaScript. The decision to do so was influenced by the desire to reduce the size of the application and create a more efficient codebase. However, building the front end from scratch proved to be inefficient. There exist multiple libraries that can be used for designing front ends, and not utilizing them would cost development time and the stability of the application.

Multiple frameworks were considered before any selections were made. Vue.JS, Angular, and React are the most well-known JavaScript front end frameworks. React was developed by Facebook for the explicit purpose of creating web-apps, and while it has taken off in popularity, it appeared complex and rather bloated. Vue.JS is an open-source framework, and while it was the most minimal, its implementation seemed counter-intuitive to how the application had been designed so far. The decision was made to utilize Angular. Angular was designed by google, it is open-source, developer friendly, and well documented. It also features compilation features similar to React. This allows for the application to be developed in a distributed file format and compiled to a single site that can be cached by the browser, a feature essential to building a modern application.

Project State Summary

Iota has come far over the last two weeks. The API has been expanded on significantly and now allows for all of the essential database calls needed to create and manage users. The front end is seeing progress as Angular has been implemented. The next two weeks shall be dedicated to expanding the user interface and adding other features to the API.