Sophia Fortier

COMPSCI 326

Professor Richards

12 July 2023

Final Project Documentation

I. Key Features

There are three main components in the final web application. The majority of the screen shows a data display which features the three tables in our database: “Recipes”, “Ingredients”, “Tags”. In selecting a different table, said contents will be displayed for the user to scroll through.

The database editor panel, which is activated by clicking an edit button in the top right hand corner of the screen, serves to integrate CRUD functionality into the user interface. Here, a pop-up will display and allow users to pick from the three tables and three actions: create, update, and delete. The read action would have been more forefront with the detailed recipe view. However, it is still incorporated into the application by allowing users to scroll through the contents of each table. The editor is completely dynamic and will display a success or failure message depending on whether a user has entered in values for all the required variables and the fetch request processes smoothly.

Located to the left on a side panel is the generate-meals form. Here, in an attempt to rectify some of the difficulties associated with meal planning in college age populations, users can select a number of days from one to seven, and a few predefined tags (“Cheap”, "Quick", and "Bulk") and subsequently generate a random selection of meals from the recipe database which meet the tag criteria. Not selecting any of the tags will return recipes with any available tags and new tags created by the user will not show up under the tags options on the side panel. Upon pressing generate, "recipe cards" will appear at the bottom of the screen, underneath the side and main panels. They show a day number, recipe name, instructions, cook time, prep time, and tags (which help the user understand how tag selected affects their choices). Pressing clear will remove all the recipe cards. However, users can also press generate another time to get a new selection.

II. Project Architecture

The file system is split into two directories which contain the server and client files. There are also a few files located in the main directory which aid the server, such as node packages and the environment file where the database URL and port number is stored.

Inside the server folder, there is ***`index.js`*** and ***`recipesdb.js`***. ***`recipesdb`*** utilizes *node-postgres* to set up a connection between a PostgreSQL database called "Recipes," hosted by ElephantSQL. There is a variable called `RecipesDatabase` which will connect the user to a series of queries located in the `RecipeQuery` object upon passing in the predefined database URL. `RecipeQuery` contains an initializer, closing function, along with all the CRUD actions for each of the three tables in our database, and a few helper functions. ***`index.js`*** uses *Express* to define all of the routing for the application.

The client side is the most robust aspect of the project, with a single html file, ***`index.html`***, a style sheet, ***`style.css`***, and five client side files: ***`client.js`***, ***`datatable.js`***, ***`generate.js`***, ***`crud.js`***, and ***`schema.js`***. ***`schema`*** is a reference file, containing a json-formatted object with the header names for each table which is used to quickly grab header names as the user changes the main table view. ***`crud.js`*** is what connects the client to the server as it exports numerous functions with fetch requests for every user-accessible route in the program. ***`client.js`***, ***`datatable.js`***, and ***`generate.js`*** all contain functions which dynamically alter the display and give functionality to the html elements. ***`client.js`*** is the primary file since it is referenced by ***`index.html`***. However, it only contains the web page's initialization function and functions associated with the database editor pop-up. Upon refreshing the page, `init()` is called and any variables from localStorage are used to provide persistence to the application. Both datatable and generate are similar to ***`client.js`***, mixing in a series of event listeners and functions which are subsequently called and alter the user's view. ***`index.js`*** is a fairly simple file, since almost all of the elements are created using javascript. It defines main dividers and static buttons in the interface.

In terms of the "Recipes" database being frequently referenced, the architecture is as follows: there are three tables—"Recipes", "Ingredients", and "Tags"—which are all connected by a serialized integer called `rid`, standing for recipe id. It is the primary key in the recipe table and referenced as a part of the primary keys in the other two. Each table requires a unique `rid` and object name for each record while providing a few other fields for related information that default to null. Enums were originally considered for tags and ingredient units, but were avoided for their lack of flexibility.

III. Usage Instructions

In order to start up the website, you must start a terminal either in your preferred code editor or on your computer itself. Change the directory so that you are at the parent folder called *sfortier_finalproject*. From there, type and enter **`npm install`** to load all the dependencies. You can then type in npm start and should see a message saying a server has been started on port 5050. You can access this page via the link: https://localhost:5050. If **`npm start`** does not work, **`node server/index.js`** will also start the server. With the environment file being provided, there is no setup necessary to connect to the database. Though, in case of unforeseen errors, you can create a .env file in the parent directory, not contained in either the client or server folders, which contains the following code:

```
DATABASE_URL =
"postgres://jkpeastl:XrDTjic8eFbjb76tJ26aA_HnnCwSObkm@ruby.db.elephantsql.com/j
kpeastl"
PORT = 5050
```