Homework 4

Introduction

This week we further developed *Memento* into a workable prototype using Javascript to interface with our Firebase server. We added the essential CRUD(Create, Read, Update, Delete) functionality to our server using the Firebase API. In addition, we implemented the ability for users to upload their own photos instead of using our stock pictures. For this, we used the Firebase storage feature to upload the files that users selected in the File Picker. we feel that through this homework assignment, we learned how to navigate new libraries and incorporate them into our code.

New Functionality

Authentication

We created ways for the users to login and authenticate themselves through registering an email/password combination or authenticating through their Google accounts. To do this, we used the firebase API's for creating accounts and storing them in their users database and making a popup for the Google Authenticator. First, we tried to use the FirebaseUI library as it was basically pre-made UI elements with the functionality already included so we didn't have to write some of the JS ourselves, but we found that it wasn't very flexible and that some of the UI elements didn't respond to our own CSS rules in the way we expected, making it hard to resize and/or rename the text inside the login buttons. We decided instead to use our previous buttons made from Bootstrap and add onclick functions to them and wrote our own JS functions for logging in. What we learned here was that almost all Firebase API calls were asynchronous and we had to wait for them to finish before calling the next function, so we had to write callback functions with Promises rather than code sequentially as we were used to. We encountered several bugs before realizing that everything must be in a callback as we tried to use the user variable before Firebase completed the login.

Create

An essential part of our app is to be able to create new memories. We implemented this functionality through Firebase's set() function. We created a JSON tree schema that has the unique user ID's as the child of the root node, then the children of the user ID are the name of the locations we want to store, and the children of the name node are its elements such as rating, city, description, etc. We implemented some error-checking to ensure that the data is consistent in our database. Specifically, important data such as the name of the place can't be missing. What we learned here was how to structure the JSON tree so that it fits our needs. Since we had to be able to relate the data the current logged in user is storing to that user in the database, we used the unique user ID to separate the data stored by different users. Then, when searching the tree we path to that user ID and only search the locations stored for that user and not any other ones.

Read

When a user returns to our app, they must be able to sign into their account and view their previously created memories. To demonstrate this operation for the sake of this assignment, we have added a "Find" button in the modal that pops up after clicking the top leftmost picture of the "My Memories" panel. When you first click it, all the fields will be empty. This is a search by name, fill out the name field with the name of a location you have entered,

and we will populate the rest of the fields and picture with the data associated with that name. For the last phase of our app we will phase out this button and populate the panels with actual pictures of the locations, with a working modal for each. For reading, we simply use the user ID and concatenate that with the name they are searching to find the fields associated with that name in the JSON tree.

Update

We realize that users sometimes make mistakes. As such, we needed to make sure they would be able to update their memories. We were able to implement this functionality via using their Unique ID's to find their created memory and update the data via the set() function. Update can only be called from the modal that pops up after clicking the top leftmost picture of the "My Memories" panel, and must be called after "Find". We simply call set() again to overwrite the previous data in that path (we use the same path as the one we used to read the correct data).

Delete

Sometimes, users can change their minds about places they want to visit. So we implemented functionality that would allow them to delete a memory from their plans. This must also be called only after finding a location with "Find", and we simply use the same path again and call remove().

Upload Picture

We felt that users should be able to upload a picture of their own liking to improve their user experience. This way, they can associate their favorite memories with a personal picture to remind them of their experience. To implement this, we used the HTML input tag with the attribute of "file" to create a file picker. For image storage, we have to use the Firebase Storage instead of the Firebase Realtime Database as the image asset cannot fit into a JSON field. To link the stored image to the current logged in user and the current location the user is adding/editing, we use the same file path from the JSON tree in the Storage, with the user ID/{location_name} so that we search for pictures in the same way. What we had issues with at first was adding the picture to the Storage before the user added the location itself to the Realtime Database, since they were linked to two different buttons. We realized that "useless" images would get stored in the Storage if they clicked that button first but we exited the modal, so we had to linked both "add" operations to one button. We made it so that clicking the file picker only gave a preview of the picture they selected on the modal, and it does not get added to Storage until the user adds their fields to the Realtime Database.

Conclusion

In this assignment, we utilized the Firebase API to bring out idea to life. We were able to create the essential parts of Memento such as creating, editing, and deleting memories. Through our process, we learned how to link up a web app to a server. We initially felt that web programming was a mystery because of it was hard to find a cohesive source to learn from. However, we now realize that web technologies generally fit into a categories. As such, we feel that we are much better prepared to go out and learn new web technologies now.