Cloudie: Serverless demo

About

This PWA was created to demo development with a "serverless" backend.

Cloudie

Cloudie is an application that displays user uploaded images and also allows them to upload new images. Anonymous users may only view images.

Architecture

Cloudie was developed using *React* as the UI framework to create a Progressive Web Application. Details of the UI implementation won't be explained.

Firebase

For firebase, we required the `firebase` package, which brings everything we need to work with all of firebase functionalities.

- **Authentication** To work with authentication, we use the `firebase.auth()` function to build an authentication request using the Google login provider. To see the login implementation, please refer to the following file: `src/server/firebase.js` We have a React component called `LoginAware` that subscribes on mount to the `onChange` event triggered by Firebase for when a user logs in or out, this allows us to change the UI depending on the user authentication status.

```
await firebase
  .auth() // Get a reference to the authentication component
  .setPersistence(firebase.auth.Auth.Persistence.LOCAL) // Persist
session across app restarts.

// Configure authentication provider
var provider = new firebase.auth.GoogleAuthProvider();
provider.addScope("profile"); // Request user profile information
(name, etc)
provider.addScope("email"); // Request user email

firebase
  .auth()
  signInWithRedirect(provider) // Signin with an HTTP redirect to the
provider URL
  .then(result => { // Signin success
    // Access user information
    var token = result.credential.accessToken;
    var user = result.user;
  })
  .error(error => {console.log("Error :(")})) // Error signing in
```

- **Storage** Storage is handled inside the same file as Authentication. We get a storage handle calling `firebase.storage()` and afterwards we create some sub-directories to order images. Once that was done, we abstract the base storage reference to this sub-directory and create files as such:

```
const imageStore = () => {
  let ref = firebase
    .storage() // We get the storage service
    .ref() // Reference the root "/"
    .child("images"); // Reference "/images"
  if (auth.currentUser) { // If the user is logged in...
    ref = ref.child(auth.currentUser.uid); // Reference "/images/:uid"
  }
  return ref;
};

// ...

imageStore()
  .child(file.name) // Create a new child under the current path with the
given name
  .put(file); // Write and upload the file
  .then(() => { console.log("Success!"); }) // Upload was successful
  .error((error) => { console.log("Error :("); )}) // Upload failed for
some reason
```

- **Firestore** Firestore is the database provider. We consume Firestore in order to query and listen for remote changes on certain paths of our database:

```
firebase
  .firestore() // Get the firestore database service
  .collection("images") // Reference the "images" root collection
(/images)
  .orderBy("uploadTime", "desc") // Order the result by the uploadTime of
each item in the collection
  .limit(100) // Bring only the first 100 results
  .onSnapshot(snapshot => {
    // Listen for changes (add/remove/modify)
    snapshot.forEach(document => {
      // Iterate through each document in the collection
      console.log(document.data()); // Access the document raw data
    });
  });
```

Prerequisites

The project requires Yarn to handle package dependencies and project execution.

Getting started

First, we need to install the required dependencies for the project:

```
yarn install
```

Once those dependencies are installed, we can deploy the project with the following command:

```
yarn start
```

Deployment

To deploy the project, you'll need to execute the following command:

```
yarn run build
```

This will generate an optimized and production-ready build inside the `build` folder.

Dependencies

This project requires various libraries in order to work.

- React
  - Used to build the user interface
- Material-UI
  - Used to get pre-built react components following Material Design
- Moment.js
  - Used to pretty-format dates
- React Router
  - Used to handle application routing
- JavaScript Load Image
  - Used to fix EXIF orientation issues when capturing or loading images with a mobile device
- Recompose
  - Used to compose React High-Order Components
- React Webcam
  - Used for the `Webcam` react component
- Firebase
  - Used for everything related to Firebase (Functions, Storage, Database, Authentication)