o'hana Functional Requirements

Prepared By Team Go

Version 0.1

Webapp

Node Packages

- Firebase
    - URL: https://firebase.google.com/docs/reference/js/
    - Version: 5.12.0
    - Purpose: We will be using Firebase to store user data about tasks and also for user authentication for the webapp. Firebase offers a greater ease at retrieving information from the database by returning DataSnapshots. These data snapshots can be converted into JSON objects for further inspection. In addition, we decided to go with Firebase over other services such as Heroku because of Firebase's email / password authentication api.
- React:
    - URL: https://reactjs.org/
    - Version: 16.3.2
    - Purpose: React offers reusable components. This allows for a consistent style and efficiency. Components can be easily modified and can be reused in multiple pages.
- React-DOM
    - URL: https://getbootstrap.com/
    - Version: v4.1.0
    - Purpose: Connects React with the DOM.
- React-Router-DOM
    - URL: https://getbootstrap.com/
    - Version: v4.1.0
    - Purpose: Navigation component in React that allows users to navigate different sections of the web app.
- Bootstrap
    - URL: https://getbootstrap.com/

- Version: v4.1.0
- Purpose: Bootstrap offers a solid framework for us to build and style the page while simultaneously providing sufficient flexibility for us to include custom CSS components. We will be using Bootstrap CDN to avoid conflicts with React. In addition to the bootstrap styles, we will be developing our own CSS style library to customize the look and feel of the webapp.

Home view Component

A React JS component to render the UI for the home view of the app.

Name	Parameters	Returns	Description
componentDidMount()	-	-	Creates an observer to listen for firebase state changes. If there is a firebase state change and user is truthy, routes app to view component
componentWillUnmount()	-	-	Removes the observer for changes to the user's sign-in state when app is routed away from sign in screen
render()	-	HTML element with user input forms for signing in	Creates and renders sign in form

Signin View Component

A React JS component to render the UI for the sign in view of the app. The class includes Firebase functionality to authenticate users and navigate them to the right page.

Name	Parameters	Returns	Description
constructor()	`props`	-	Creates a state object initializing user email and password
componentDidMount()	-	-	Creates an observer to listen for firebase state changes. If there is a firebase state change and user is truthy, routes app to view component
componentWillUnmount()	-	-	Removes the observer for changes to the user's sign-in state when app is routed away from sign in screen
render()	-	HTML element with user input forms for signing in	Creates and renders sign in form
handleSubmit()	`JS event`	Async promise	Calls firebase signin with email and password based on state. Catches any errors and renders error message in user interface

Connectors

Once the user is authenticated in the sign in view, they are navigated to the tasks tab. So, so the user object is one of the props that is passed on to the tasks view.

Signup View Component A React JS component to render the UI for the sign up view of the app. The class includes

Firebase functionality to create a user in with the given email and password in our database.			
Name	Parameters	Returns	Description
constructor()	`props`	-	Creates a state object initializing device ID, roommates, chores, counts, user IDs, and lists to be empty to begin with.
componentDidMount()	-	-	Creates an observer to listen for firebase state changes. If there is a firebase state change and user is truthy, routes app to view component
componentWillUnmount()	-	-	Removes the observer for changes to the user's sign-in state when app is routed away from sign up screen. Also handles password validation.
handleSubmit()	`JS event`	Async promise	Performs input verification, ensuring that all fields are filled and valid. Returns error message if inputs are invalid or if passwords don't match. Calls firebase createUserAndPassword function otherwise. Catches

			any error returned by the server and renders them on the sign up page.
render()	–	HTML element with user input form with fields for email address, password, and display name.	Creates and renders multiple forms with validity checking for user sign up (Forms according to website sign up page mockup)

Connectors

Once the user is profile is created in the database, they are navigated to the roommates tab. The user object is passed on as one of the props and it will be used as the first element in the roommates list.

Roommates Component A React JS component to render the UI for one of the two main tabs in the webapp. The class includes functionality to read user data snapshots from Firebase database and renders it accordingly. The UI also allows for the user to add and remove roommates.			
Name	Parameters	Returns	Description
constructor()	`props`	-	Creates a state object with a fields for a list of roommates
componentDidMount()	–	-	Initialize the fields in the state with the data snapshots retrieved from Firebase whenever the page is reloaded
componentWillUnmount()	–	-	Stop listening for new data snapshots from Firebase.
componentWillReceiveProps()	`nextProps`	-	Used to pass in a

			new set of properties to the component. In this case, it's the new tab name.
storeName()	`JS event`	-	Inserts the new name from the user into the firebase database
editName()	`JS event`	-	Over-writes an existing roommate's name in the firebase database
deleteName()	`JS event`	-	Removes the given name from the firebase database
render()	–	An HTML element which consists of a list of users	Displays a list of roommates and also provides the option to add and remove users from the list. Provides for add, remove and edits by initializing listeners on each input to watch for changes or deletes. E

Tasks Component

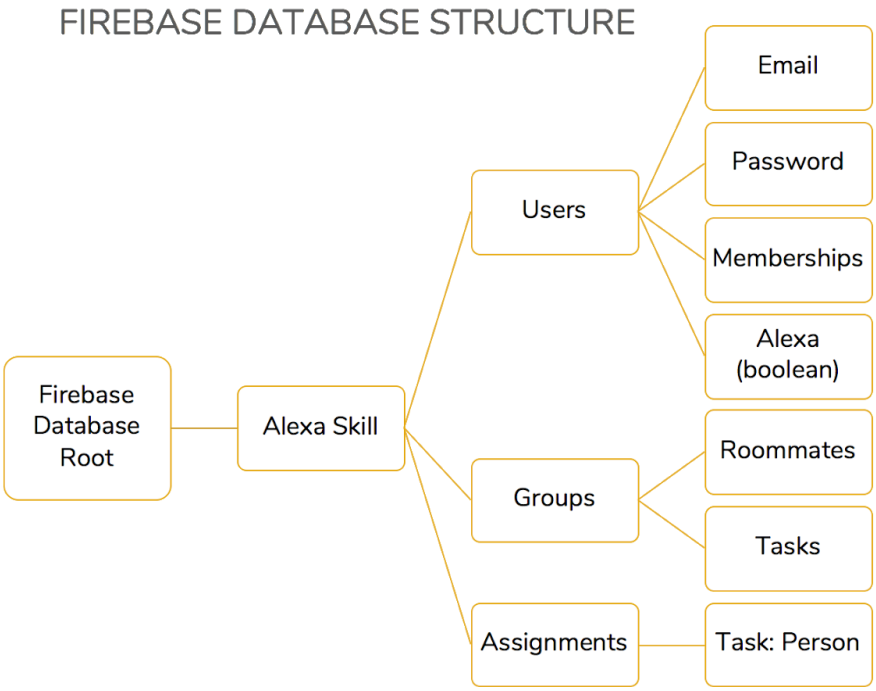
A React JS component to render the UI for one of the two main tabs in the webapp. The class includes functionality to read assignment data snapshots from Firebase database and renders it. The UI also allows for the user to re-assign the tasks among roommates.

Name	Parameters	Returns	Description
constructor()	`props`	-	Creates a state object with a fields for a list

			of tasks
componentDidMount()	–	-	Initialize the fields in the state with the data snapshots retrieved from Firebase whenever the page is reloaded
componentWillUnmount()	–	-	Stop listening for new data snapshots from Firebase.
componentWillReceiveProps()	`nextProps`	-	Used to pass in a new set of properties to the component. In this case, it's the new tab name.
storeTask()	-	-	Stores task to the firebase database
editTask()	-	-	Allows the user to edit the roommate assigned to the task.
deleteTask()	-	-	Removes task from the firebase database
render()	–	An HTML element which consists of a list of tasks	Displays a list of tasks and also provides the option to add and remove tasks from the list.

Firebase Database Structure

We have modularized the architecture of the web app and the alexa skill based on the database structure illustrated in the following diagram.



Alexa Skill

Node Packages

- Alexa-app
  - URL: https://www.npmjs.com/package/alexa-app
  - Version: 4.2.2
  - Purpose: A module to simplify the creation of Amazon echo skills.
- Firebase
  - URL: https://firebase.google.com/docs/reference/js/
  - Version: 5.12.0
  - Purpose: We will be using Firebase to store user data about tasks and also for user authentication for the webapp. Firebase offers a greater ease at retrieving information from the database by returning DataSnapshots. These data snapshots can be converted into JSON objects for further inspection. In addition, we decided to go with Firebase over other services such as Heroku because of Firebase's email / password authentication api.

DBConn A node module to interact with Firebase by reading and/or updating roommate and task related data on the Firebase database.

Name	Parameters	Returns	Description
getAssignments()	`string`	`string`	Queries the Firebase database for all the tasks associated with that device id. Filters the task with the person's name (input string) and returns it.
updateAssignments()	–	–	Takes the updated list and over writes Firebase database's assignments data snap shot

Ohana			
Name	Parameters	Returns	Description
markAsDone()	`string`	-	Extracts the roommate's name from the voice command. Queries the Firebase database with the person's name and updates to the person's next task.
getTasks()	`string`	-	Extracts the roommate's name from the voice command. Query Firebase database with person's name and read the task assigned to