
CS 23-332 TELEHEALTH APP FOR VETERANS

Quality of Life +

Kyle Perry

Yafet Zelleke

Kenneth Olivar

Setup instructions for future group

- Overview
- Getting the first app running
- Alexa Skill setup

Overview

Introduction

Hi, my name is Kyle, and I was in the group before you all for developing this app. I'm sure you know this, but you all will be the third installment of VCU engineers working on this app. My goal is to help you all get what has currently been developed up and running on your machines so you can continue to work on this.

The original group's task was to create the initial app, while our group's task was to link the app with a smart TV like Roku to improve communication between caretakers and patients. Unfortunately, the documentation from the original group is not laid out very well, so we will do our best to help explain it to you.

Getting the First App Running

The initial app created from the original group is a JavaScript web application created using the React library and Expo. If you have experience with React and JavaScript, then you should be able to run this no problem, if not, it would be helpful to watch a basic "Expo React startup" video. However, I will give you a brief overview of what each element means.

- JavaScript
 - o Programming language like Java, one of the most popular languages in the world and is primarily used in front and back-end web design.
- React
 - o Library for JavaScript that makes creating and editing the user interface for web applications super easy.
- Expo and Expo Go
 - o Library for JavaScript which works in tandem with React and allows you to run your web application on Android and Apple devices.

- Download the 'Expo Go' app from the app store, and it will allow you to see developments on your app in real time, which is neat.

All the information regarding the initial application is under the /src/ folder linked in the original team's GitHub, ask your mentor or advisor for help accessing their GitHub if you experience any difficulties.

The biggest issue that you will experience is that when you try to get their previous application running, it most likely won't load, or only partially load. This is because their application was developed a couple of years prior, and a lot of the dependencies they require are not up to date. They didn't provide any documentation on getting the web application running, so hopefully your job is to re-invent their app or something, because it was nearly impossible to get it running.

My biggest recommendation for you all (assuming you don't have any experience with front-end JavaScript design) would be to create and play with your own basic React Expo application, just so you can experience it and see how it works.

Another recommendation I would have, depending on what you are tasked with, is to create your own implementation of the app instead of trying to work off theirs. Their application was honestly low quality and is now incredibly outdated. I would try to convince whoever your mentor/advisor is to do this because it will help you in the long run.

Alexa Implementation

Contrary to what I said previously, our group did not end up using Roku in our implementation because Roku could not effectively accomplish our task at hand. Instead of this, we developed an implementation with Alexa which did accomplish what we needed. I'm not sure if you will need to get a version of this Alexa skill running, but I will explain how to get it running if you do need to:

Alexa Skill Setup

In simple terms, here is what the Alexa implementation accomplishes:

1. The patient will install the Alexa skill from the Alexa Skills store, provide permissions to the skill, and then say: "Alexa, open QL Plus" - *Alexa response* - Store User ID."
 - a. This one-time command will send the user's information to the database along with their Alexa User Skills ID.
2. The caretaker will add reminders on behalf of the patient (including a date, time, and message)
 - a. Ideally, the caretaker will add a lot of reminders on behalf of the user, to limit user interaction.
3. The caretaker will ask the patient to say "Alexa, open QL Plus – *Alexa response* - Add reminders."
 - a. Alexa will add reminders for the patient to their Alexa, along with 3 backup reminders.
4. The patient will say "Alexa, open QL Plus – *Alexa response* - I took my pills (or whatever action intended)"
 - a. Alexa will delete the backup reminders (if any remain) and it will not inform the caretaker.
 - b. If there is no acknowledgement, Alexa will inform the caretaker that their action was not completed.

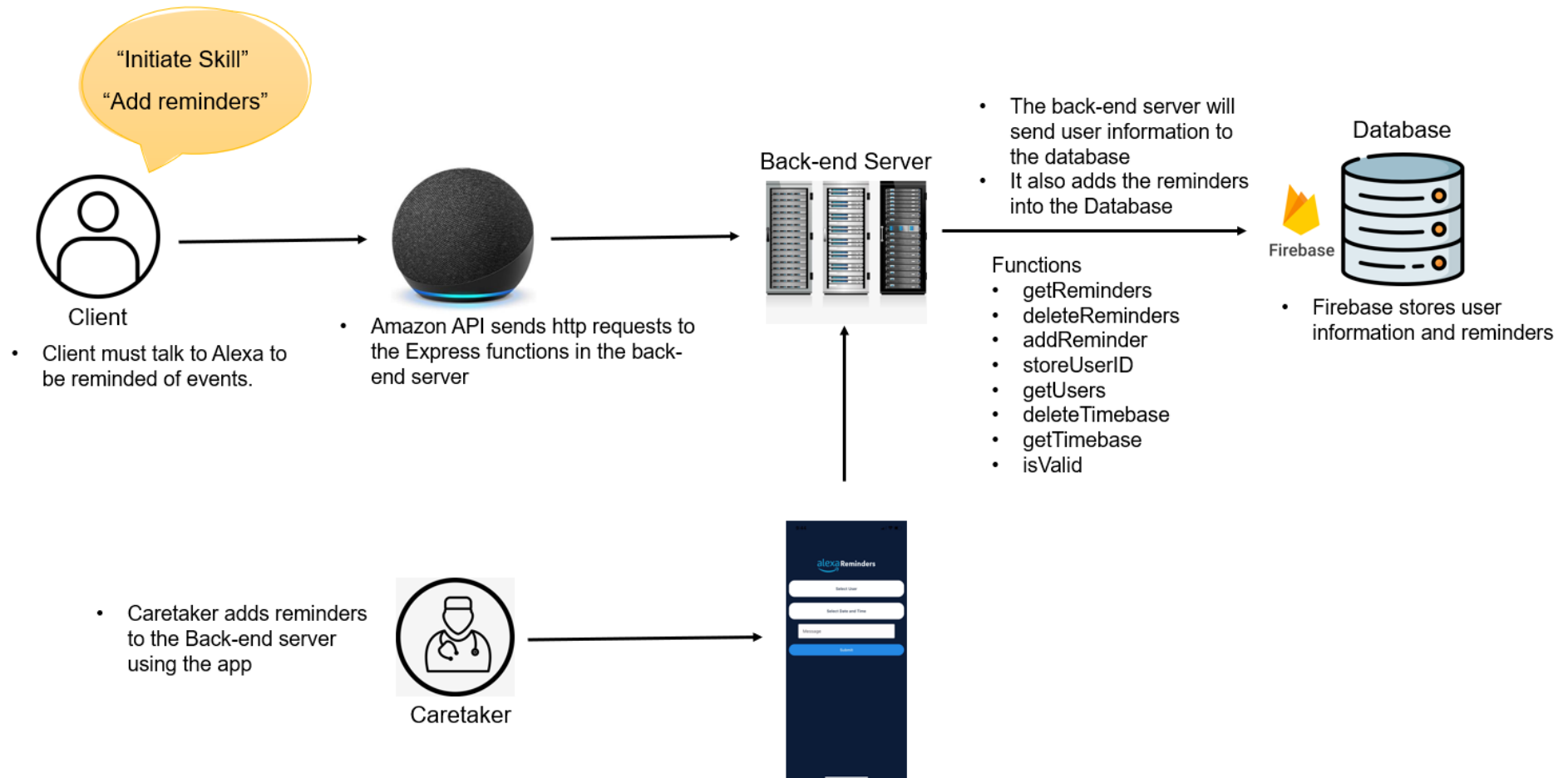
Thankfully, all the code developed for this shouldn't be going out of commission anytime soon. Unfortunately, you must set up everything for yourself to run this skill. This involves several steps which I will do my best to walk you through. Here are the concepts you should be familiar with:

- SAAS general design (software as a service)
 - o HTTP requests***
 - o API's
- Firebase
 - o Firebase functions
 - o Firebase firestore database
- Alexa Skills
 - o Alexa Skills Kit
 - o Alexa Skills Permissions
- Front-end web app design

- React Native
- JavaScript
- Expo

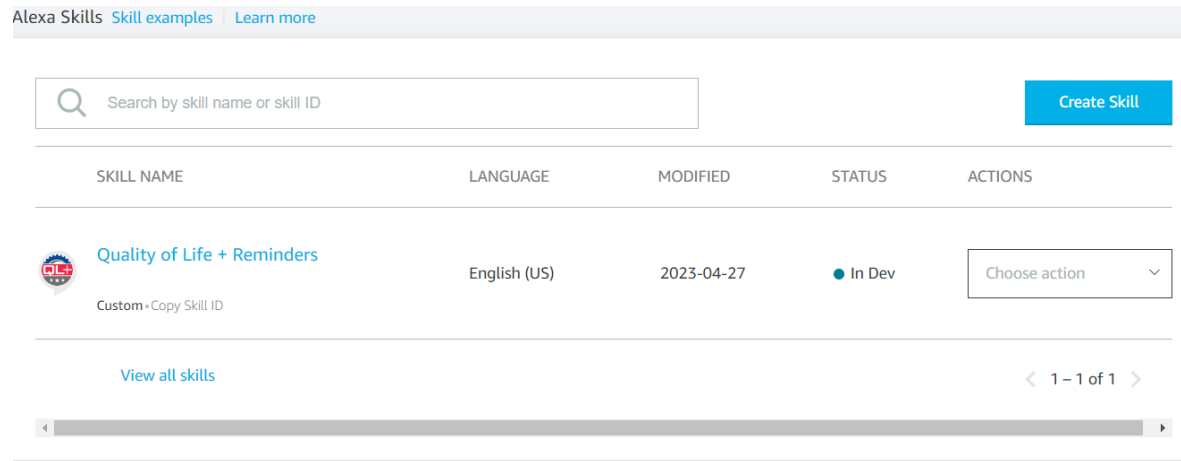
Once you have a general understanding of the following concepts, you should be able to set up all the provided code.

Conceptual Understanding

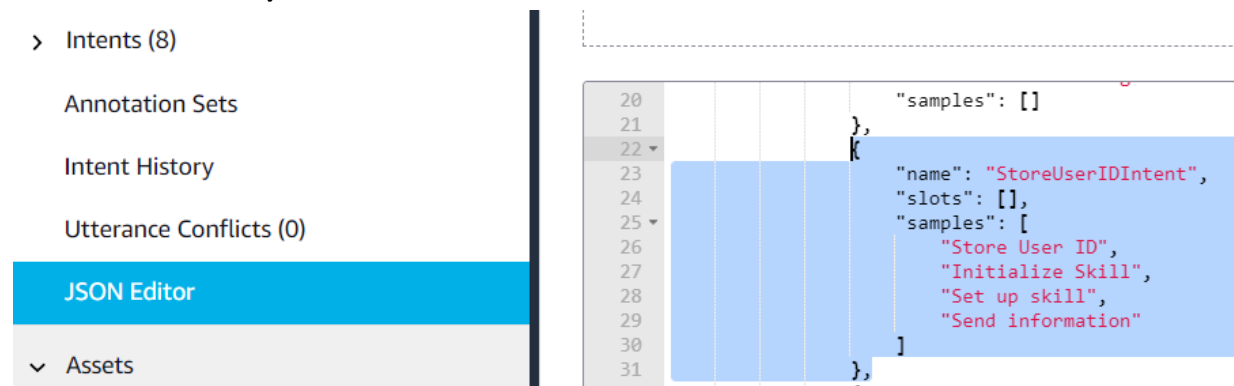


Here is the high-level overview of the Alexa app and its processes, I will explain everything as if you were familiar with the concepts listed above.

Alexa Skill



The Alexa skill is comprised of two main components, the “index.js” file which consists of the Intent Handlers, and the “interactionModel.json”. The interaction model creates intent names, and then includes the phrases that will invoke them.



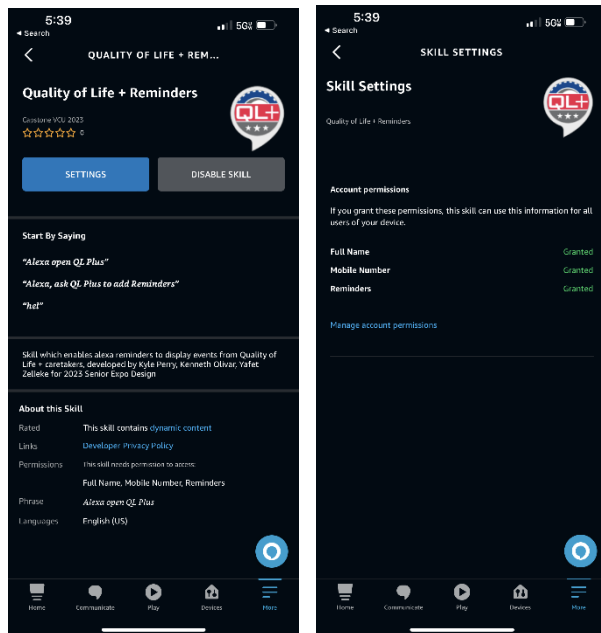
For example, to run “StoreUserIDIntent”, you can say “Store User ID”, “Initialize Skill”, etc.

That will call on this following intent handler located in the "index.js" file:

```
// Handler to call function to send user ID and full name to backend server
const StoreUserIDIntentHandler = {
  canHandle(handlerInput) {
    return (
      handlerInput.requestEnvelope.request.type === "IntentRequest" &&
      handlerInput.requestEnvelope.request.intent.name === "StoreUserIDIntent"
    );
  },
};
```

The Alexa skill is accessible in the Alexa Skills Developer console under the account:
expovideo2023@gmail.com

Make sure you have permissions enabled within the Alexa App:



Backend Server

The Alexa skill will make HTTP requests to and from the backend server, which you must set up on your own. The backend server is included as “index.js” in the edusourced folder.

The backend server is set up using Firebase functions. You need to create a Firebase account with a Gmail account (you can use the expovideo2023@gmail.com account if you want), and set up “Firebase pay as you go” account services, which requires a credit card.

Thankfully, firebase doesn't charge you until you reach a ridiculous amount of API calls, I set a limit for \$5.00 on my personal card but never got charged. I'm sure you can get a card number from your advisor.

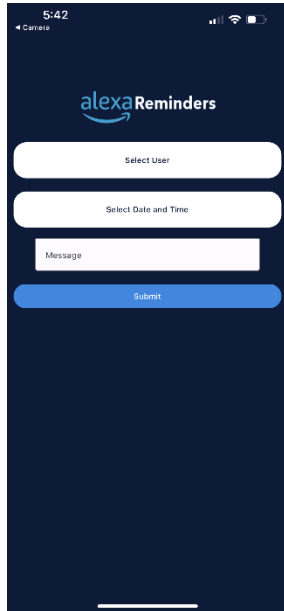
Once you create a firebase functions server, replace the index.js file it creates for you with the index.js file included in edusourced. This index.js is an API that gets called by the Alexa and the caretaker to add and remove data from the database. **This API is the only way Alexa can communicate with the database.**

Prior to testing the API, you should set up the Firestore database as well under the same account.

User Interface

To add, get, remove, delete reminders from the firestore database, you must interact with the API. The “App.js” file I have included in edusourced is a great example of creating reminders with the API. Keep in mind that you must have initialized a user on the Skill for the database to create reminders.

Here is what the test interface looks like:

A screenshot of a mobile application interface titled 'alexaReminders'. The interface is set against a dark blue background. At the top, there is a status bar showing the time '5:42' and various icons. Below the title, there are three white input fields stacked vertically. The first field is labeled 'Select User', the second 'Select Date and Time', and the third 'Message'. Below these fields is a blue button labeled 'Submit'. The bottom of the screen shows a white home indicator bar.

Keep in mind that adding and deleting reminders from the database is entirely created through HTTP requests through the API, so you can theoretically create any interface you want to add reminders, (e.g., Google Calendar, Mail, Desktop app, ETC.)

This was a lot of information, and it might be overwhelming.. If you have any questions, please definitely hesitate to ask. But if you absolutely cannot figure something out, email me at kyleperry2121@gmail.com. We got no help from the previous group, and I want to help you all get where you all need to be. But seriously like only worst-case scenario hit me.

Final Notes

- Creating a workspace in visual studio helped me work with the user interface and the backend functions files really easily. It made it easy to load up and put away.
- You should only be developing the alexa skills kit within the ASK developer console online.
- There is going to be a few functions in the backend server that might not make sense, for example "isValidTime" and "isTimeBase". These functions are for recognizing if the user is in a valid window to let the caretaker know they took their pills or called their doctor, etc.
- **You will need to change all of the URL's in the Alexa skills kit (and the App.js) to call to your API endpoint, I am deprecating my API endpoint and you will need to find all instances of my API URL and replace it with yours.**
- Don't forget to ensure that all of the permissions are enabled.
- **Alexa reminders can only be added with an "in-session access token", meaning that you cannot create reminders in the backend, which is why the user must say "add reminders". You can view and delete reminders out of session**
- You can view created reminders in the Alexa app under 'reminders'