**VOICE SEARCH**

**Requirements:**

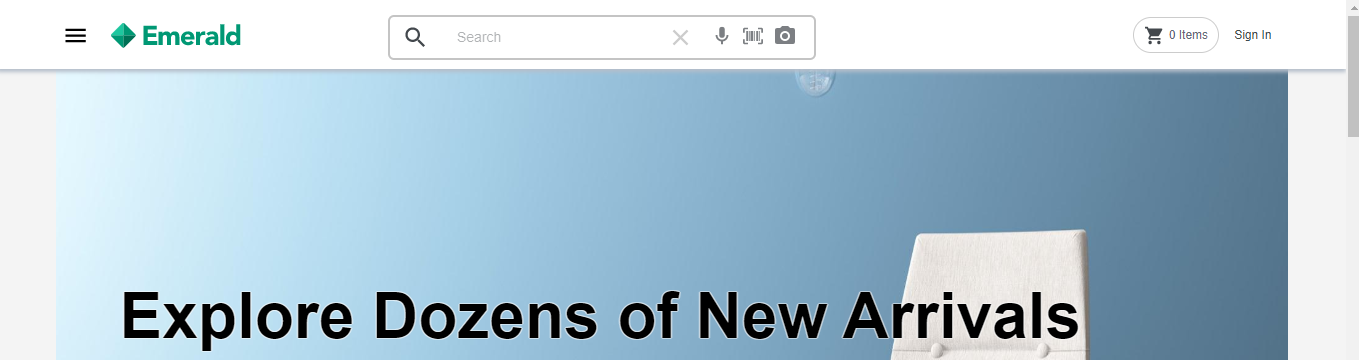
Provide the capability to search HCL Commerce Catalog using voice search on React Stores.

**User Flow:**

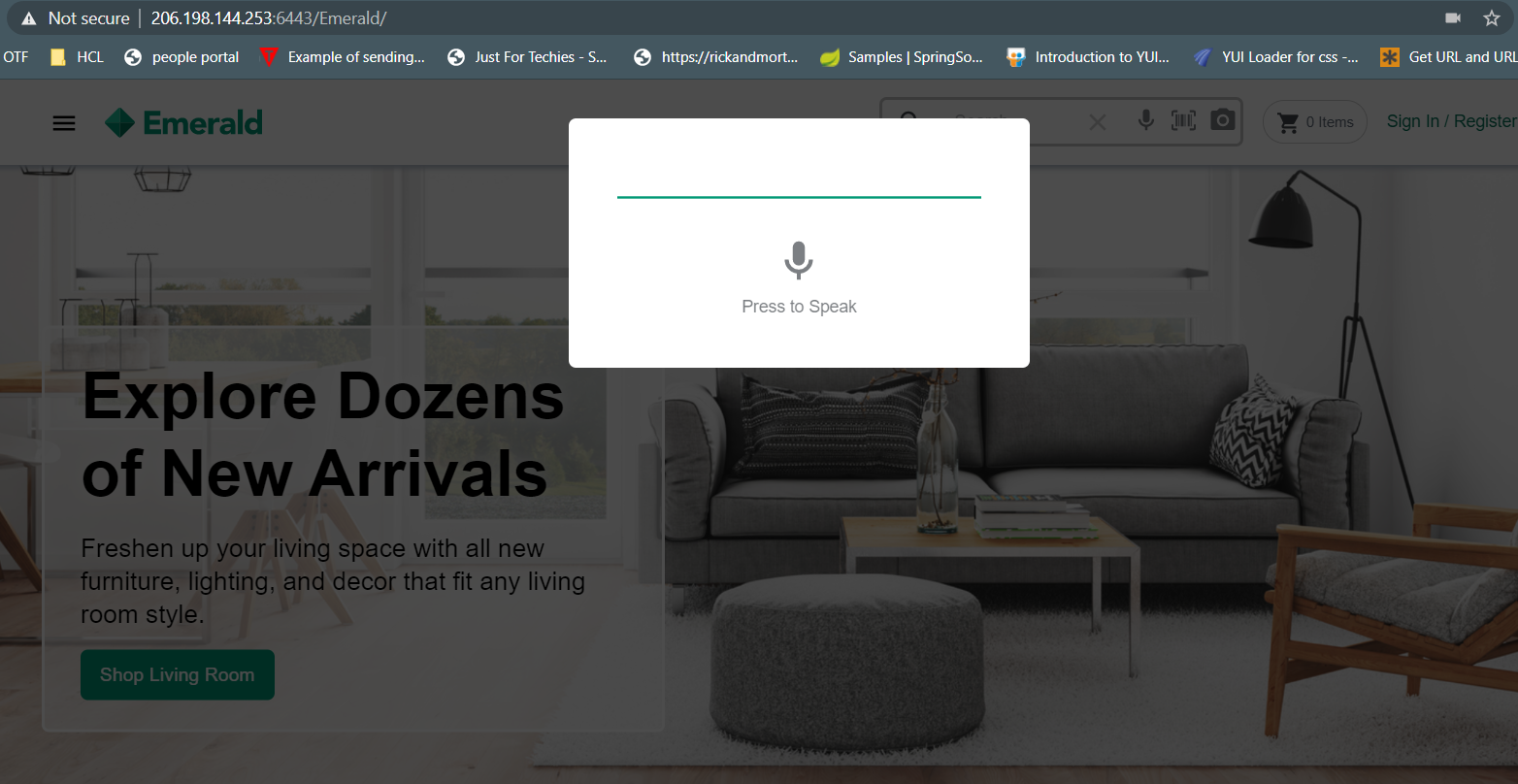
React Audio Analyser is used to recording audio and drawing the curve. It supports converting the audio to wav.

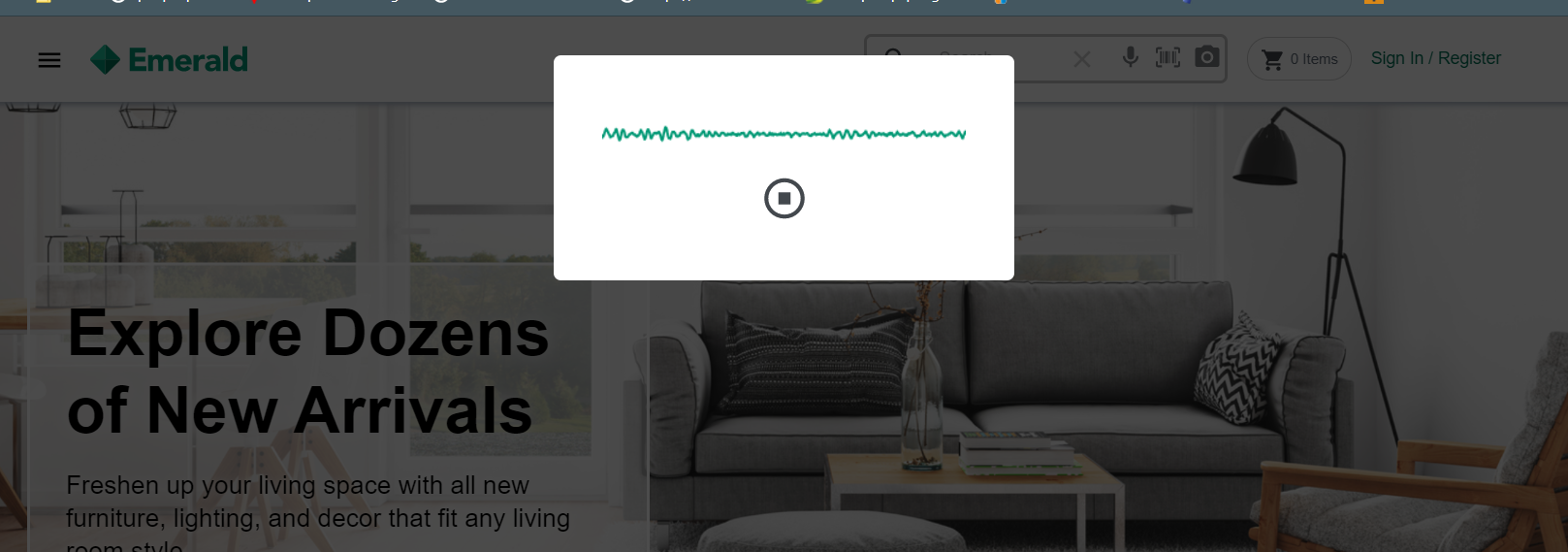
It’s an open source library at <https://www.npmjs.com/package/react-audio-analyser>.

The search bar appears as a below on load.

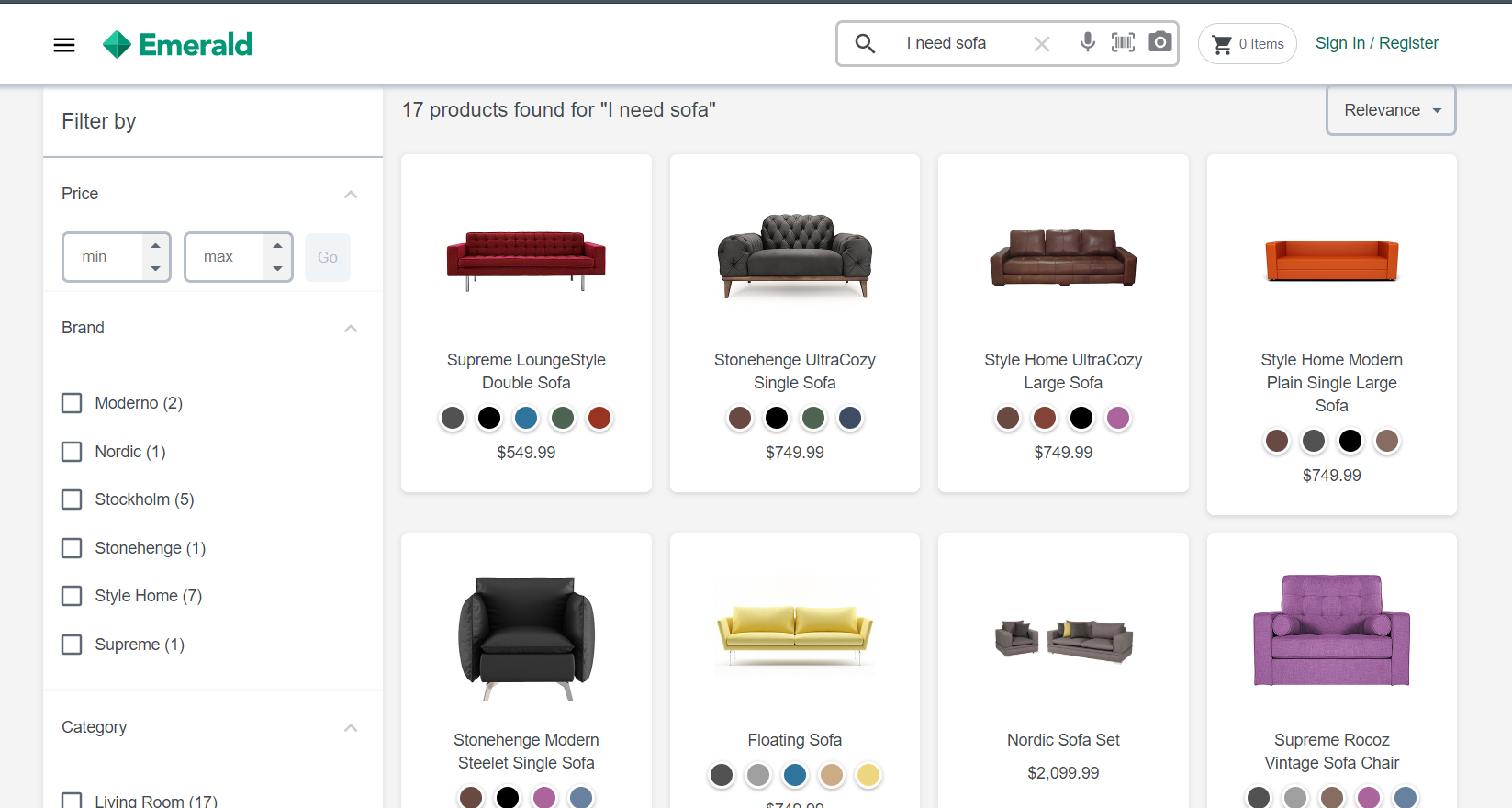


Onclick of mic icon it starts recording the audio along with the wave formation.





The text is rendered in the Input box and search result appears on page :



Note:

-The library “react-audio-analyser “used for recording audio doesn't support in IOS chrome and safari as this library uses HTML5 Media Recorder APIs “getUserMedia and MediaRecorder” which are not supported by chrome and safari in IOS.

**Steps to include the Voice search scanner in your project:**

1. You need to install the react audio anayser in your project as a dependency.

Complete usage is given at <https://www.npmjs.com/package/react-audio-analyser>

It can be installed using npm.

> npm install react-audio-analyser –save

Once installation is done. Verify the entry in your package.json file.

1. Below changes need to add in the SearchBar.tsx file present in src/components/widgets/search-bar folder

We have created a Search Type component to specify the search type for the user.

User can configure this according to his choice

Import this file in the SearchBar.tsx file.

import { SearchTypes } from "../Search-types/search-types";

This component can be used as below.

<SearchTypes

                    showBarcodeIcon={true}

                    showSpeechToText={true}

                    showImageTotext={true}

                    setSearchBoxVal={setInput}

                  />

showBarcode Iconà used for barcode scanning.

showSpeechToText à used for Voice transcribe.

showImageToText à used for Image search.

setSearchBoxValà used to set the vaue of the search result.

1. The Search Types component looks like below.

return (

    <>

      {props.showSpeechToText ? (

        <VoiceToText setSearchBox={props.setSearchBoxVal} />

      ) : null}

      {props.showBarcodeIcon ? (

        <BarcodeScan setSearchBox={props.setSearchBoxVal} />

      ) : null}

      {props.showImageTotext ? (

        <ImageToText setSearchBox={props.setSearchBoxVal} />

      ) : null}

    </>

In this document we will see only the VoiceToText Component.

1. The VoiceToText component looks like below:

Partial snippet



It has the below two icons imported from the iconify library

import microphoneIcon from "@iconify/icons-mdi/microphone";

import stopCircleOutline from "@iconify/icons-mdi/stop-circle-outline";

<div id="micWave">

                    <AudioAnalyser

                      audioType="audio/wav"

                      status={status}

                      startCallback={startCallback}

                      stopCallback={stopCallback}

                      width={300}

                      height={40}

                      strokeColor="#009874"

                      className={`canvas-box ${hideClass}`}

                      backgroundColor="#FFFFFF"

                    />

                  </div>

- Add the mic and stop button to start and stop the recording of the microphone.

{status === "" || status === "inactive" ? (

                    <span className="icon" onClick={startRecording}>

                      <Icon icon={microphoneIcon} width="22px" height="24px" />

                    </span>

                  ) : (

                    <span className="icon" onClick={stopRecording}>

                      <Icon

                        icon={stopCircleOutline}

                        width="22px"

                        height="24px"

                      />

                    </span>

                  )}

The display of the icon depend upon the status attribute and also the recording status.

- Add below code snippet

State variable for Recoding the audio.

const [status, setStatus] = React.useState("");

const startRecording = () => {

    setStatus("recording");

    setRecordingStarted(true);

    setHideClass("");

    setInput("");

  };

  const stopRecording = () => {

    setSpinner(true);

    setStatus("inactive");

    setHideClass("hidden");

    setRecordingStarted(false);

  };

For Styling the wave canvas, we need to make the below css changes.

-Add the below CSS in SearchTypes.scss file



**Google Speech API is used to translate the audio to text.**

It takes the base 64 encoded audio as an input and gives the translated text as a output.

The complete details can be read at <https://cloud.google.com/speech-to-text>.

Once the audio is recorded it is posted to the speech recognize API as below.

We have created the firebase api to call the google speech API.

The call to the firebase API is placed in the voiceImageTranscribeService.

const speechToText = async base64Audio => {

    try {

      const res = await voiceImageTranscibeService.getVoiceTranscribeText(

        base64Audio

      );

      console.log("RESPONSE RECEIVED: ", res);

      res.data.results

        ? props.setSearchBox(res.data.results[0].alternatives[0].transcript)

        : props.setSearchBox("No Data Found...");

      setSpinner(false);

    } catch (err) {

      console.log("ERROR: ", err);

      setSpinner(false);

    }

  };

The code in the service looks like below.

/\*\*local URLs (will delete once above will start working) \*/

const VOICE\_URL =

  "https://us-central1-emerald-a9fa8.cloudfunctions.net/app/voice-transcribe";

  getVoiceTranscribeText(audioBytes): AxiosPromise<any> {

    let requestOptions: AxiosRequestConfig = Object.assign({

      data: { audioBytes },

      url: VOICE\_URL,

      method: "post"

    });

    return Axios(requestOptions);

  },

The above service returns the transcribed word of the user’s speech.