VSCODE_PRINT_SCRIPT_TAGS

# Folder qalbi\src

## 18 printable files

qalbi\src\App.test.tsx
qalbi\src\App.tsx
qalbi\src\components\ExploreContainer.css
qalbi\src\components\ExploreContainer.tsx
qalbi\src\hooks\DataHook.tsx
qalbi\src\main.tsx
qalbi\src\pages\Advice.json
qalbi\src\pages\AdviceTab.css
qalbi\src\pages\AdviceTab.tsx
qalbi\src\pages\GraphTab.css
qalbi\src\pages\GraphTab.tsx
qalbi\src\pages\HomeScreen.css
qalbi\src\pages\HomeScreen.tsx
qalbi\src\pages\SettingsTab.css
qalbi\src\pages\SettingsTab.tsx
qalbi\src\setupTests.ts
qalbi\src\theme\variables.css
qalbi\src\vite-env.d.ts

## qalbi\src\App.test.tsx

```tsx
import React from 'react';
import { render } from '@testing-library/react';
import AppRoute from './App';

test('renders without crashing', () => {
  const { baseElement } = render(<AppRoute />);
  expect(baseElement).toBeDefined();
});

```

## qalbi\src\App.tsx

```tsx
import { Redirect, Route } from 'react-router-dom';
import {
  IonApp,
  IonButton,
  IonFab,
  IonFabButton,
  IonIcon,
  IonInput,
  IonRouterOutlet,
  IonTabBar,
  IonTabButton,
  IonTabs,
  IonText,
  IonToast,
  setupIonicReact,
  useIonLoading,
} from '@ionic/react';
import { IonReactRouter } from '@ionic/react-router';
import { analyticsOutline, bluetoothOutline, bookmarksOutline, homeOutline, lockOpenOutline, settingsOutline } from 'ionicons/icons';
import GraphTab from './pages/GraphTab';
import AdviceTab from './pages/AdviceTab';
import SettingsTab from './pages/SettingsTab';
import HomeScreen from './pages/HomeScreen';
import { useState, useEffect } from 'react';
import { dataHook, fetchRecords } from './hooks/DataHook';
import { Filesystem, Encoding, Directory } from '@capacitor/filesystem';
import { Preferences } from '@capacitor/preferences';
import { LocalNotifications, LocalNotificationSchema } from '@capacitor/local-notifications';

/* Core CSS required for Ionic components to work properly */
import '@ionic/react/css/core.css';

/* Basic CSS for apps built with Ionic */
import '@ionic/react/css/normalize.css';
import '@ionic/react/css/structure.css';
import '@ionic/react/css/typography.css';

/* Optional CSS utils that can be commented out */
import '@ionic/react/css/padding.css';
import '@ionic/react/css/float-elements.css';
import '@ionic/react/css/text-alignment.css';
import '@ionic/react/css/text-transformation.css';
import '@ionic/react/css/flex-utils.css';
import '@ionic/react/css/display.css';

/* Theme variables */
import './theme/variables.css';
setupIonicReact();

const LOWER_HRV = "lower_hrv";
const UPPER_HRV = "upper_hrv";
const PASSCODE = "passcode";
const NOTIFICATIONS = "notifications";

/*
 * React Functional Component responsible for setting up global states and creating the routing for the device android application.
 */
const AppRoute: React.FC = () => {
  /*
  // Test Code!
  const dumpHrv = async () => {
    const {files} = await Filesystem.readdir({
      path:"",
      directory:Directory.Data
    });

    console.log(files.map(file => file.name))

    files.forEach(async (file) => {
      if (file.name.includes("HRV")) {
        await Filesystem.deleteFile({
          path: file.name,
          directory: Directory.Data
        })
      }
    });
  }

  const testHrv = async () => {

    const buffer = new ArrayBuffer(8);
    const view = new DataView(buffer);
    const start = Date.now();

    for (var i = 0; i < 1000; i++) {
      const timestamp = (start - (Math.random()*(604800000-1000) + 1000))/1000
      const rmssd = Math.random()*50 + 70;

      view.setUint32(0, timestamp, true);
      view.setFloat32(4, rmssd, true);

      await storeRecord(view);
    }

    await determineUserState();
  };

  */

  // On startup, load the passcode
  useEffect(() => {
    loadPasscode();
  }, []);

  // userState is the user's current stress state
  // 0 : normal
  // -1: fatigued
  // 1: stressed
  const [stressState, setStressState] = useState<number>(0);
  const [connected, setConnected] = useState(false);

  /*
   * Callback to trigger whenever a new record is written to the HRV characteristic by the device wearable.
   * rawRecord is the received value from the HRV characteristic
   */
  const hrvCallback = async (rawRecord:DataView): Promise<void> => {
    await storeRecord(rawRecord);
    await determineUserState();
  };

  /*
   * Stores an HRV record given by the raw bits of rawRecord.
   * rawError is the received value from the HRV characteristic, following the format in design specification 4.5.1.
   */
  const storeRecord = async (rawRecord:DataView): Promise<void> => {
    /*
     * Parse the HRV Characteristic
     *
     * UUUUHHHH
     * U: Unix timestamp: uint32
     * H: HRV metric: float32
     *
     * Arduino is little endian, so read and write to the DataView with the little endian flag set for multibyte datatypes.
     */

    const timestamp = rawRecord.getUint32(0, true);
    const rmssd = rawRecord.getFloat32(4, true).toFixed(2).padStart(6, '0');

    console.log(timestamp);
    console.log(rmssd);

    // Format the record
    const record = `${timestamp} ${rmssd}\n`;

    // Get the record's year, month, and day to make the HRV day data file filename
    const currentDatetime = new Date(timestamp*1000);

    const year = currentDatetime.getUTCFullYear().toString().padStart(4, '0');
    const month = (currentDatetime.getUTCMonth()+1).toString().padStart(2, '0');
    const day = currentDatetime.getUTCDate().toString().padStart(2, '0');

    const filename = `HRV-${year}${month}${day}.txt`;

    // Attempt to read the day data file.
    try {
      const contents = await Filesystem.readFile({
        path: filename,
        directory: Directory.Data,
        encoding: Encoding.UTF8
      });

      // if the record is not already in the data file, append the record.
      if (!contents.data.includes(record)){
        await Filesystem.appendFile({
          path: filename,
          data: record,
          directory: Directory.Data,
          encoding: Encoding.UTF8
        });
      }

    } catch (error) {

      // Write to the day datafile to create a new file.
      await Filesystem.writeFile({
        path: filename,
        data: record,
        directory: Directory.Data,
        encoding: Encoding.UTF8
      });
    }
  }

  /*
   * Sets the userState global state based on HRV metrics (see requirements 3.2.2.1.4-3.2.2.1.8, 3.2.2.2.1)
   */
```

```
const determineUserState = async (): Promise<void> => {
  const baselineRecords = await fetchRecords(3 * 24 * 60 * 60); // Records from 3 days ago to now
  const sampleRecords = await fetchRecords(3 * 60 * 60); // Records from 3 hours ago to now

  // If there are no records, end method to avoid division by zero
  if (baselineRecords.length <= 0 || sampleRecords.length <= 0){
    return;
  }

  // Get the mean of each set of records to serve as the respective HRV metric.
  const baselineHRV = baselineRecords.map((record) => record[1]).reduce((acc, curr) => acc + curr, 0) / baselineRecords.length;
  const sampleHRV = sampleRecords.map((record) => record[1]).reduce((acc, curr) => acc + curr, 0) / sampleRecords.length;

  // Get the current user set thresholds
  const {value: rawUpperHrv} = await Preferences.get({key: UPPER_HRV});
  const upperHRV = Number(rawUpperHrv || "200");

  const {value: rawLowerHrv} = await Preferences.get({key: LOWER_HRV});
  const lowerHRV = Number(rawLowerHrv || "0");

  // If the user is "stressed", write 1 to the userState
  // Else if the user is "fatigued", write -1 to the userState
  // Else write 0 to the user state
  console.log(sampleHRV);
  console.log(baselineHRV);

  if (sampleHRV > 107 || sampleHRV > 1.15 * baselineHRV || sampleHRV > upperHRV)
    setStressState(1);
  else if (sampleHRV < 16 || sampleHRV < 0.85 * baselineHRV || sampleHRV < lowerHRV)
    setStressState(-1)
  else
    setStressState(0);
}

// Send notification on stress state change
useEffect(()=> {handleStressChange()}, [stressState]);
const handleStressChange = async () => {
  const notificationsOn = Boolean((await Preferences.get({key:NOTIFICATIONS})).value);

  if ((await LocalNotifications.checkPermissions()).display != 'granted'){
    await LocalNotifications.requestPermissions();
  }

  if (notificationsOn) {

    var header:string = "";
    var message:string = "";

    if (stressState == 0) {
      header = "Great job with your stress management!"
      message = "Tranquil+ detected that your stress levels are good! Keep it up!"
    }
    else if (stressState == -1) {
      header = "You might be a bit fatigued."
      message = "Try taking a break for a bit to catch some rest!"
    }
    else {
      header = "You might be a bit stress."
      message = "Try checking out the advice section in the Tranquil+ app to get some activities to destress!"
    }

    const notification:LocalNotificationSchema = {
      title: header,
      body: message,
      id:1,
    };
    LocalNotifications.schedule({notifications: [notification]});
  }
}

var readjustError = 0; // number of readjust Errors in the last hour
var timeoutID: NodeJS.Timeout|undefined = undefined; // timeout to handle clearing readjustError

/*
 * Handles error codes sent from the device wearable to the Error characteristic.
 * rawError is the received value from the characteristic, following the format in design specification 4.5.1.
 */
const errorCallback = async (rawError: DataView): Promise<void> => {

  /*
   * Parse the Error Characteristic
```

```
     *
     * Z
     * Z: Error code: uint8
     */

    const errorCode = rawError.getUint8(0);

    // If the errorCode is 1, a readjust error has occurred
    if (errorCode == 1) {
      readjustError += 1; // Increment readjust error counter

      // If 5 readjustErrors have occurred in the last hour, set a local notification and reset the timeout
      if (readjustError >= 5) {
        handleErrorMessage();
        clearTimeout(timeoutID);
      }

      // If this is the first readjust error, set a timeout to clear the reajustError value to 0 after an hour.
      if (readjustError == 1){
        timeoutID = setTimeout(
          () => {
            readjustError = 0;
            timeoutID = undefined
          },
          60*60
        )
      }
    }

    // Log the error code
    console.error('Error Code', errorCode);
  }

  // Send error message on fail to read
  const handleErrorMessage = async () => {
    const notificationsOn = Boolean((await Preferences.get({key:NOTIFICATIONS})).value);

    if ((await LocalNotifications.checkPermissions()).display != 'granted'){
      await LocalNotifications.requestPermissions();
    }

    if (notificationsOn) {
      const notification:LocalNotificationSchema = {
        title: "Device Misread",
        body:"Your Tranquil+ device is not properly reading your heart rate! Try readjusting the glove fit so that the sensor rests on the fingertip.",
        id:0,
      };
      LocalNotifications.schedule({notifications:[notification]});
    }
  }

  // Log in page
  const [loggedIn, setLoggedIn] = useState<boolean>(false); // Is user logged in?
  const [present, dismiss] = useIonLoading(); // Loading for getting user passcode
  const [passcode, setPasscode] = useState<string>(); // User passcode

  // Load the passcode
  const loadPasscode = async () => {
    present("Getting Creds");
    setPasscode((await Preferences.get({key:PASSCODE})).value || "");
    dismiss();
  }

  // On passcode load, log in if there is no passcode
  useEffect(() => {
    if (passcode == "") {
      setLoggedIn(true);
    }

    if (passcode != undefined)
      dismiss();
  }, [passcode])

  useEffect(() => {if (loggedIn) dataHook([hrvCallback, errorCallback], () => setConnected(true), () => setConnected(false))}, [loggedIn]); // Start dataHook on login

  const [failLogin, setFailLogin] = useState(false);

  // Handle login on password submission
  const handleLogin = () => {
    //@ts-ignore
    const pass:string = document.getElementById('passcode-input').focusedValue || "";
    setLoggedIn(pass==passcode);
    setFailLogin(pass!=passcode);
  }

  // Notifications on disconnect
  const handleDisconnect = async () => {
    const notification:LocalNotificationSchema = {
      title: "Device Disconnected",
      body:"Your Tranquil+ device disconnected! Start the Tranquil+ App to reconnect.",
      id:0,
    };

    // if the device disconnects, send a notification
    if (!connected) {
      const notificationsOn = Boolean((await Preferences.get({key:NOTIFICATIONS})).value);

      if ((await LocalNotifications.checkPermissions()).display != 'granted'){
        await LocalNotifications.requestPermissions();
      }

      if (notificationsOn) {
        LocalNotifications.schedule({notifications: [notification]});
      }
    }
    else {
      // Remove notification on reconnect
      LocalNotifications.removeDeliveredNotifications({notifications:[notification]})
    }
  }

  useEffect(() => {handleDisconnect()}, [connected]) // On connection status change, handle notifications

  return (
    <IonApp>
      {!loggedIn? passcode===undefined? undefined: passcode==""? undefined:
        <div className="react-lock-screen__ui">
          <IonText
          >
            <h1>Welcome</h1>
          </IonText>

          <IonToast
            isOpen={failLogin}
            duration={3000}
            message="Incorrect Password"
            onDidDismiss={() => setFailLogin(false)}
          >

          </IonToast>

          <IonInput
            placeholder="Enter Your Passcode."
            id='passcode-input'
            inputMode="numeric"
            type="password"
          />

          <IonButton onClick={handleLogin} shape='round' className='ion-padding'>
            <IonIcon icon={lockOpenOutline} size="large"/>
          </IonButton>
        </div>

        :

        <>
          <IonReactRouter>
            <IonTabs>
              <IonRouterOutlet>
                <Route exact path="/">
                  <Redirect to="/home" />
                </Route>
                <Route exact path="/tab1">
                  <GraphTab />
                </Route>
                <Route exact path="/tab2">
                  <AdviceTab />
                </Route>
                <Route exact path="/tab3">
                  <SettingsTab />
                </Route>
                <Route exact path="/home">
                  <HomeScreen stressState={stressState}/>
                </Route>
              </IonRouterOutlet>
              <IonTabBar slot="bottom">
                <IonTabButton tab="home" href="/home">
                  <IonIcon aria-hidden="true" icon={homeOutline}  size='large'/>
                </IonTabButton>
                <IonTabButton tab="tab1" href="/tab1">
                  <IonIcon aria-hidden="true" icon={analyticsOutline}  size='large'/>
                </IonTabButton>
                <IonTabButton tab="tab2" href="/tab2">
                  <IonIcon aria-hidden="true" icon={bookmarksOutline}  size='large'/>
                </IonTabButton>
                <IonTabButton tab="tab3" href="/tab3">
                  <IonIcon aria-hidden="true" icon={settingsOutline}  size='large'/>
                </IonTabButton>
              </IonTabBar>
            </IonTabs>
          </IonReactRouter>

          <IonFab vertical="top" horizontal="end" slot="fixed">
            <IonFabButton
              color={connected? "primary":"danger"}
              onClick={() => {console.log("Bluetooth Reconnect."); dataHook([hrvCallback, errorCallback], () => setConnected(true), () =>
setConnected(false))}}
            >
              <IonIcon aria-hidden="true" icon={bluetoothOutline} />
            </IonFabButton>
          </IonFab>
        </>
      }
    </IonApp>
  );
};

export default AppRoute;

```

## qalbi\src\components\ExploreContainer.css

```css
.container {
  text-align: center;
  position: absolute;
  left: 0;
  right: 0;
  top: 50%;
  transform: translateY(-50%);
}

.container strong {
  font-size: 20px;
  line-height: 26px;
}

.container p {
  font-size: 16px;
  line-height: 22px;
  color: ▨ #8c8c8c;
  margin: 0;
}

.container a {
  text-decoration: none;
}
```

## qalbi\src\components\ExploreContainer.tsx

```tsx
import './ExploreContainer.css';

interface ContainerProps {
  name: string;
}
```

```

const ExploreContainer: React.FC<ContainerProps> = ({ name }) => {
  return (
    <div className="container">
      <strong>{name}</strong>
      <p>Explore <a target="_blank" rel="noopener noreferrer" href="https://ionicframework.com/docs/components">UI Components</a></p>
    </div>
  );
};

export default ExploreContainer;

```

## qalbi\src\hooks\DataHook.tsx

```
import { BleClient, numberToUUID } from "@capacitor-community/bluetooth-le";
import { Preferences } from "@capacitor/preferences";
import { Directory, Filesystem, Encoding } from "@capacitor/filesystem";

const DEVICE_ID = "device_id" // Preference ID for Device Wearable ID.

// BLE Service and Characteristic UUIDs
const HRV_SERVICE = numberToUUID(0x180F); // Bluetooth Low Energy HRV Metric Service UUID
const HRV_CHARACTERISTIC = numberToUUID(0x2A19); // Bluetooth Low Energy Characteristic UUID (receive HRV records from device wearable)
const ERROR_CHARACTERISTIC = numberToUUID(0x2A1A); // Bluetooth Low Energy Characteristic UUID (receive error codes from device wearable)
const REQUEST_CHARACTERISTIC = numberToUUID(0x2A1B); // Bluetooth Low Energy Characteristic UUID (send data requests to device wearable)

/*
 * Hook responsible for handling and maintaining connections with the device wearable.
 * Takes in callbacks array, which assigns these functions to occur when their respective BLE characteristic is written to by the device wearable
 */
export const dataHook = async (callbacks:Array<(value:DataView) => void>, onConnect:()=>void, onDisconnect:()=>void) => {
    try {
        await BleClient.initialize(); // Start the BleClient at the beginning of the program

        // Disconnect from any previous connections
        const connections = await BleClient.getConnectedDevices([HRV_SERVICE]);

        connections.forEach(async (connection) => {
            await BleClient.disconnect(connection.deviceId);
        })


        var connected = false; // Bluetooth connection state
        var id: string; // BLE peripheral device id (the Device Wearable)

        // While not connected:
        do {

            // Check the saved preference for a Device Wearable ID
            const {value} = await Preferences.get({key: DEVICE_ID});

            if (value) {
                // If there is a Device Wearable ID stored in preferences, use the saved ID
                id = value;
            }
            else {
                // Else, Search for a device to connect to and save its ID to the preference.
                const device = await BleClient.requestDevice({
                    services: [HRV_SERVICE]
                })

                id = device.deviceId;
                await Preferences.set( {key: DEVICE_ID, value: id});
            }

            // Attempt to connect to the device wearable 5 times.
            var attempts = 0;
            while (!connected && attempts < 2) {
                try {
                    await BleClient.connect(id, () => onDisconnect());
                    connected = true;

                } catch (error) {
                    attempts++;
                }
            }

            // If the Device Wearable fails to connect, clear the saved ID preference
            if (!connected) {
                await Preferences.remove({key: DEVICE_ID});
            }

        } while (!connected);

        onConnect();

        // Start Notifications for the HRV and Error Characteristics
        // Updates to the HRV Characteristic should trigger the first callback
        // Updates to the Error Characteristic should trigger the second callback
        await BleClient.startNotifications(
            id,
            HRV_SERVICE,
            HRV_CHARACTERISTIC,
            callbacks[0]
        );

        await BleClient.startNotifications(
            id,
            HRV_SERVICE,
            ERROR_CHARACTERISTIC,
            callbacks[1]
        );

        // TODO: save the current last record read to effeciently recall records.
        // Get the time stamp for last week
        const lastWeekDate = new Date(Date.now() - 6.048e+8);

        /*
         * Pack the Request Characteristic into its format
         *
         * YYMD
         * Y: Year: uint16
         * M: Month: uint8
         * D: Day: uint8
         *
         * Arduino is little endian, so read and write to the DataView with the little endian flag set for multibyte datatypes.
         */

        // Create an ArrayBuffer
        const buffer = new ArrayBuffer(4);

        // Create a DataView for bit manipulation
        const view = new DataView(buffer);

        // Format the request data into the DataView
        view.setUint16(0, lastWeekDate.getUTCFullYear(), true);
        view.setUint8(2, lastWeekDate.getUTCMonth()+1);
        view.setUint8(3, lastWeekDate.getUTCDate());

        // Write a Records Request to the Request Characteristic
        await BleClient.writeWithoutResponse(
            id,
            HRV_SERVICE,
            REQUEST_CHARACTERISTIC,
            view
        )

        console.log("Done!")


    } catch (error) {
        // Log errors to console
        console.error(error);

        // TODO: Put Local notification

        // Reset the dataHook after 30 seconds
        //setTimeout(() => dataHook(callbacks), 30000);
    }
}

/*
 * Retrieves records from device application storage.
 * Takes in an amount of seconds, which denotes how long ago to look back.
 * Returns the found records in an array.
 */
export const fetchRecords = async(timePeriod:number): Promise<number[][]> => {
    var rawData = ""; // String holding the raw content of all the files the function reads.

    const endingTimestamp = Date.now(); // Timestamp to stop search, the current timestamp

    const startTimestamp = endingTimestamp - timePeriod*1000; // Starting timestamp is {timePeriod} seconds ago

    console.log(new Date(startTimestamp));
    console.log(new Date(endingTimestamp));

    const {files} = await Filesystem.readdir({
        path:"",
        directory:Directory.Data
    });

    console.log("Bruh:", files.map(file => file.name))

    var currentTimestamp = startTimestamp; // The current timestamp as the loop control variable.

    while (currentTimestamp <= endingTimestamp) {
        // Get the currentTimestamp's year, month, and day to make the HRV day data file filename
        const currentDatetime = new Date(currentTimestamp);

        const year = currentDatetime.getUTCFullYear().toString().padStart(4, '0');
        const month = (currentDatetime.getUTCMonth()+1).toString().padStart(2, '0');
        const day = currentDatetime.getUTCDate().toString().padStart(2, '0');

        const filename = `HRV-${year}${month}${day}.txt`;

        // Attempt to read from the current day data file
        // If it exist, append its content to the raw data string
        // Else, log FileNotFound error
        try {
            const contents = await Filesystem.readFile({
                path: filename,
                directory: Directory.Data,
                encoding: Encoding.UTF8,
            });
            rawData += contents.data
        } catch (error) {
            console.error(`${filename} doesn't exist`)
        }

        // Increment timestamp by 1 day
        currentTimestamp += 8.64e+7;
    }

    var records: number[][] = []; // Formatted Records array

    // If there is data, parse it.
    if (rawData != ""){

        // Split each record (separated by newline character) and split numbers within each record (separated by a single space)
        const allRecords = rawData.split("\n").map(element => element.split(' ').map(e => Number(e)));

        // Remove records with timestamps outside of the range [startTimestamp, endingTimestamp]
        records = allRecords.filter((row) => row[0] >= startTimestamp/1000 && row[0] <= endingTimestamp/1000);

        // Sort records for convenience.
        records.sort((record1, record2) => record1[0] - record2[1]);
    }

    console.log(records);
    return records;
}

```

## qalbi\src\main.tsx

```
import React from 'react';
import { createRoot } from 'react-dom/client';
import AppRoute from './App';

const container = document.getElementById('root');
const root = createRoot(container!);
root.render(
  <AppRoute />
);
```

## qalbi\src\pages\Advice.json

```json
{
  "scent": [
    "Lavender for tension release and sleep",
    "Jasmine for relaxing tension",
    "Vanilla for a sweeter relaxation",
    "Ylang ylang to release negative moods",
    "Peppermint for invigoration and focus",
    "Lemon for calming down"
  ],
  "water": [
    "Drink a glass of water slowly",
    "Stay hydrated!"
  ],
  "quote": [
    "Worry empties today of strength",
    "Give your stress wings and let it fly away",
    "Recognize that you are doing the best you can",
    "You will survive whatever is coming, and thrive after"
  ],
  "breath": [
    "4-7-8 technique:\n1. Place tip of tongue against tissue above upper front teth and hold it there\n2.Completely exhale through mouth\n3. Close mouth and inhale through nostrils for a count of 4\n4. Hold breath for count of 7\n5. Exhale through mouth for count of 8",
    "Diaphragmatic:\n1. Inhale slowly and deeply through nose (abdomen should expand and chest very little rise)\n2. Exhale through mouth, pursing lips and keeping jaw relaxed",
    "Resonance:\n1. Lie down and close eyes\n2. Mouth closed, gently breath in through nose for 6 seconds (do not fill up lungs all the way)\n3. Exhale for 6 seconds, allowing breath to leave slowly, not forced\n4. Repeat for up to 10 minutes."
  ]
}
```

## qalbi\src\pages\AdviceTab.css

```css
.popover {
  white-space: pre-wrap;
}
```

## qalbi\src\pages\AdviceTab.tsx

```tsx
import { IonContent, IonHeader, IonPage, IonTitle, IonToolbar, IonIcon, IonButton, IonPopover, IonText, IonCol, IonGrid, IonRow } from '@ionic/react';
import './AdviceTab.css';
import { bulbOutline, chatbubblesOutline, flameOutline, pauseCircleOutline, waterOutline } from 'ionicons/icons';
import { useEffect, useState } from 'react';
import data from './Advice.json';

/*
 * React Functional Component responsible for creating the front end of the advice tab for the user.
 */
const AdviceTab: React.FC = () => {

  const [type, setType] = useState<string>(""); // State for the type of advice selected
  const [text, setText] = useState<string>(""); // State for the toast message

  // On type selection, load a random piece of corresponding advice from the Advice json
  useEffect( () => {
    if (Object.keys(data).includes(type)){
      //@ts-ignore
      const strings = data[type]
      setText(strings[Math.floor(Math.random()*strings.length)]);
    }
  }
   , [type])

  useEffect( () => {
    if (text != "") {
      console.log(text)
    }
  }, [text])

  return (
    <IonPage>
      <IonHeader>
        <IonToolbar className='ion-text-center'>
          <IonTitle><h1>Advice</h1></IonTitle>
        </IonToolbar>
      </IonHeader>
      <IonContent fullscreen>

        <IonPopover
          isOpen={text != ""}
          onDidDismiss={() => {setText(""); setType("")}}
        >
          <div className='popover ion-text-center ion-padding'>
            <IonIcon icon={bulbOutline}/>
            <IonText>
              <h2>{toTitleCase(type) + ":\n\n" + text}</h2>
            </IonText>
          </div>
        </IonPopover>

        <IonGrid className="homepage ion-text-center">
          <h1>How would you like to relax?</h1>

          <IonRow>
            <IonCol>
              <IonButton onClick={() => setType("scent")} shape="round">
                <IonIcon icon={flameOutline} size='large'/>
                <h2> Scent</h2>
              </IonButton>
            </IonCol>
          </IonRow>

          <IonRow>
            <IonCol>
              <IonButton onClick={() => setType("water")} shape="round">
                <IonIcon icon={waterOutline} size='large'/>
                <h2> Water</h2>
              </IonButton>
            </IonCol>
          </IonRow>

          <IonRow>
            <IonCol>
              <IonButton onClick={() => setType("quote")} shape="round">
                <IonIcon icon={chatbubblesOutline} size='large'/>
                <h2> Quote</h2>
              </IonButton>
            </IonCol>
          </IonRow>

          <IonRow>
            <IonCol>
              <IonButton onClick={() => setType("breath")} shape="round">
                <IonIcon icon={pauseCircleOutline} size='large'/>
                <h2> Breath</h2>
              </IonButton>
            </IonCol>
          </IonRow>
        </IonGrid>
      </IonContent>

    </IonPage>
  );
};

// Make strings title case
function toTitleCase(str:string) {
  return str.replace(
    /\w\S*/g,
    function(txt) {
      return txt.charAt(0).toUpperCase() + txt.substr(1).toLowerCase();
    }
  );
}

export default AdviceTab;

```

## qalbi\src\pages\GraphTab.css

```css
.full {
    width: 100%;
    height: 100%;
    margin: 0;
}

.homepage {
    height: 100%;
    display: flex;
    align-items: center;
    flex-direction: column;
    justify-content: center;
}
```

## qalbi\src\pages\GraphTab.tsx

```tsx
import { IonButton, IonButtons, IonCard, IonCol, IonContent, IonGrid, IonHeader, IonPage, IonRow, IonTitle, IonToolbar, useIonLoading } from '@ionic/react';
import './GraphTab.css';
import { useState, useEffect } from 'react';

import { Scatter } from 'react-chartjs-2';
import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend,
} from 'chart.js';
import { fetchRecords } from '../hooks/DataHook';

import { Preferences } from '@capacitor/preferences';

// Preference IDs for user set HRV threshold values
const LOWER_HRV = "lower_hrv";
const UPPER_HRV = "upper_hrv";

ChartJS.register(
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend
);

ChartJS.defaults.font.size = 16;

/*
 * React Functional Component responsible for creating the front end of the graph tab for the user.
 * Takes in userSettings as a prop to read the HRV thresholds
 */
const GraphTab: React.FC = () => {

  // Stateful variable for the current chart.js data
  const [chartData, setChartData] = useState<any>({
    labels: [],
    datasets: []
  });

  // Stateful variable for the timeframe the user selects
  // 0: 1 hour
  // 1: 1 day
  // 2: 1 week

  const [timeframe, setTimeframe] = useState<number>(0);

  const [present, dismiss] = useIonLoading(); // Loading box when getting graph data
```

```typescript

  /*
   * Sets the graphData based on the HRV records that are within the current timeframe.
   */
  const getChartData = async (): Promise<void> => {
    present("Loading Chart Data"); // Show loading box at start of function

    var startTime = new Date(); // Start time to compare record timestamps to.
    var timePeriod: number; // Number of seconds to look back for records

    // Set timePeriod based on timeframe selection
    if (timeframe == 0) {
      timePeriod = 60 * 60; // 1 hour in seconds
    }
    else if (timeframe == 1) {
      timePeriod = 24 * 60 * 60; // 1 day in seconds
    }
    else if (timeframe == 2) {
      timePeriod = 7 * 24 * 60 * 60; // 1 week in seconds
    }
    else {
      // If the timeframe is not 0, 1, or 2, throw an error
      console.error("Invalid Timeframe Selection");
      dismiss();
      return;
    }

    const records = await fetchRecords(timePeriod); // Fetch the records for the corresponding time period

    // If no records exist, end execution to avoid division by zero
    if (records.length <= 0) {
      console.error("No records")
      setChartData({
        labels: [],
        datasets: []
      });
      dismiss();
      return;
    }

    var divisor: number;
    var multiplier: number = 1;
    var timeunit:string;

    // Split by 5 min intervals if past hour selected
    if (timeframe == 0) {
      divisor = 5*60*1000;
      multiplier = 5;
      timeunit = "minute";
    }
    // Split by even hour if past day selected
    else if (timeframe == 1){
      divisor = 2*60*60*1000;
      multiplier = 2;
      timeunit = "hour";
    }
    // Split by day if last week selected
    else {
      divisor = 24*60*60*1000;
      timeunit = "day";
    }

    // variable to hold the labels for each time point
    const labels = records.map((record) => {
      const time = new Date(record[0]*1000);
      //@ts-ignore
      return Math.floor((startTime - time)/divisor) * multiplier;
    })

    const unique_labels = [... new Set(labels)]; // Pull unique labels

    // Get friendly labels for each data point
    const friendly_labels: string[] = unique_labels.map((value) => `${value} ${value == 1? timeunit:timeunit+"s"} ago`);

    const values = new Array<number>(unique_labels.length); // Array to store aggregated values for each unique label

    // For each unique label, store the average of all record HRV with the same label
    unique_labels.forEach((label, i) => {
      const vals = records.filter((_, i) => labels[i] == label).map((record) => record[1]);

      values[i] = vals.reduce((acc, curr) => acc + curr, 0)/vals.length;
    })

    // Collect data points together in data object to pass as a data set
    const aggregatedRecords = unique_labels.map((label, i) => ({x: label, y: values[i]}));

    const colors = await colorRecords(values); // Get the colors according to their value.

    // If no colors, throw error
    if (colors.length <= 0) {
      console.error("No colors")
      setChartData({
        labels: [],
        datasets: []
      });
      dismiss();
      return;
    }

    // Set chartData to a chart.js data object
    const data = {
      labels: friendly_labels,
      datasets: [{
        label: 'HRV',
        data: aggregatedRecords,
        fill: false,
        borderColor: colors,
        backgroundColor: colors,
        tension: 0.1,
        pointRadius: 10,
        showLine: true
      }]
    };

    setChartData(data);
    dismiss();
    console.log(data);
  }

  /*
   * Assigns a color to each HRV record based on its HRV value.
   * Takes in an array of HRV values.
   * Returns an array of RGB values.
   */
  const colorRecords = async (values: number[]): Promise<String[]> => {
    const baselineRecords = await fetchRecords(3 * 24 * 60 * 60); // Records from 3 days ago to now

    // If there are no records, end method to avoid division by zero
    if (baselineRecords.length <= 0) {
      console.error("No records")
      return [];
    }

    // Get baselineHRV from the average of the record's HRV values.
    const baselineHRV = baselineRecords.map((record) => record[1]).reduce((acc, curr) => acc + curr, 0) / baselineRecords.length;

    console.log(baselineHRV);

    // Get the current user set thresholds
    const {value: rawUpperHrv} = await Preferences.get({key: UPPER_HRV});
    const userUpper = Number(rawUpperHrv || "200");

    const {value: rawLowerHrv} = await Preferences.get({key: LOWER_HRV});
    const userLower = Number(rawLowerHrv || "0");

    // Colors corresponsing to each record
    const colors = values.map( (HRV) => {
      // If the record is stressed or fatigued, map the record to a red color
      if (HRV > 107 || HRV > 1.15 * baselineHRV || HRV > userUpper || HRV < 16 || HRV < 0.85 * baselineHRV || HRV < userLower)
        return "#c46c7b";

      // Else If the records is close to stressed or close to fatigued, map the record to a yellow color
      if (HRV > 1.08 * baselineHRV || HRV < 0.92 * baselineHRV)
        return "#f0d973";

      // Else map the record to a green color
      return "#91f2a6";
    });

    return colors;
  }

  useEffect(() => {getChartData()}, [timeframe]); // getChartData on startup and every timeframe change

  return (
    <IonPage>
      <IonHeader>
        <IonToolbar className='ion-text-center'>
          <IonTitle><h1>HRV Readings</h1></IonTitle>
        </IonToolbar>
      </IonHeader>
      <IonContent fullscreen>

        <IonGrid className="homepage">

          <IonRow style={{"flexGrow":1, "width":"100%"}}>
            <IonCol className='full'>
                <IonCard className='full'>
                  <Scatter options={{
                    responsive: true,
                    maintainAspectRatio: false,
                    plugins: {
                      legend: {
                        display: false
                      },
                    },
                    scales : {
                      x : {
                        title: {
                          display: true,
                          text: timeframe == 0? "Minutes Ago": timeframe == 1? "Hours Ago" : "Days Ago"
                        },
                        max: timeframe == 0? 70: timeframe == 1? 25: 8,
                        min: -1,
                        reverse: true
                      },
                      y :{
                        title : {
                          display: true,
                          text: "HRV"
                        },
                     //   min: 40,
                     //   max: 140
                      }
                    }
                  }} data={chartData}/>
                </IonCard>
            </IonCol>
          </IonRow>

          <IonRow className='ion-text-center'>
            <IonButtons>
              <IonButton
                fill={timeframe==0? "solid":"outline"}
                onClick={() => setTimeframe(0)}
                shape="round"
                color="primary"
                >
                <h2>Past Hour</h2>
              </IonButton>

              <IonButton
                fill={timeframe==1? "solid":"outline"}
                onClick={() => setTimeframe(1)}
                shape="round"
                color="primary"
                >
                <h2>Today</h2>
              </IonButton>

              <IonButton
                fill={timeframe==2? "solid":"outline"}
                onClick={() => setTimeframe(2)}
                shape="round"
                color="primary"
                >
                <h2>This Week</h2>
              </IonButton>
            </IonButtons>
          </IonRow>

        </IonGrid>

      </IonContent>
    </IonPage>
  );
};

export default GraphTab;

```

## qalbi\src\pages\HomeScreen.css

```css
.homepage {
  height: 100%;
  display: flex;
  align-items: center;
  flex-direction: column;
  justify-content: center;
}

.react-lock-screen__ui {
  width: 100vw;
  height: 100vh;
  display: flex;
  align-items: center;
  flex-direction: column;
  justify-content: center;
  text-align: center;
}

.lock {
  filter: blur(100px);
  height: 100vh;
  overflow: hidden;
}

h1 {
  font-size: 24pt !important;
}

h2 {
  font-size: 12pt !important;
}
```

## qalbi\src\pages\HomeScreen.tsx

```tsx
import { IonCard, IonCol, IonContent, IonGrid, IonHeader, IonPage, IonRow, IonTitle, IonToolbar, useIonRouter } from "@ionic/react";
import './HomeScreen.css';

/*
 * React Functional Component responsible for creating the front end of the home screen for the user.
 * The content depends on the current userState.
 */
const HomeScreen: React.FC<{stressState: number}>  = ({stressState}) => {
  return (
    <IonPage>
      <IonHeader>
        <IonToolbar className='ion-text-center'>
          <IonTitle><h1>Tranquil+</h1></IonTitle>
        </IonToolbar>
      </IonHeader>
      <IonContent fullscreen>
        <IonGrid className="homepage">

          <IonRow style={{"flexGrow":1, "alignItems":"flex-end"}}>
            <IonCol>
              <h1>Your current stress level</h1>
            </IonCol>
          </IonRow>

          <IonRow style={{"flexGrow":2, "alignItems":"flex-start"}}>
            <IonCol>
              <IonCard className="ion-padding">
                <h1>{stressState == 1? "Stressed": stressState  == -1? "Fatigued":"Normal"}</h1>
              </IonCard>
            </IonCol>
          </IonRow>

        </IonGrid>
      </IonContent>
    </IonPage>
  );
}

export default HomeScreen;
```

## qalbi\src\pages\SettingsTab.css

```
.setting {
  width:85%;
  align-items:center;
  min-height: 66px;
}
```

## qalbi\src\pages\SettingsTab.tsx

```
import { IonButton, IonCol, IonContent, IonGrid, IonHeader, IonInput, IonPage, IonRow, IonTitle, IonToast, IonToggle, IonToolbar, useIonLoading } from
'@ionic/react';
import './SettingsTab.css';
import { Preferences } from '@capacitor/preferences';
import { useEffect, useState } from 'react';
import { LocalNotifications } from '@capacitor/local-notifications';

// Preference IDs for each setting
const LOWER_HRV = "lower_hrv";
const UPPER_HRV = "upper_hrv";
const PASSCODE = "passcode";
const NOTIFICATIONS = "notifications";

/*
 * React Functional Component responsible for creating the front end of the settings tab for the user.
 */
const SettingsTab: React.FC = () => {

  // Component states for each setting
  const [passcode, setPasscode] = useState<string>("");
  const [upperHRV, setUpperHRV] = useState<number>(107);
  const [lowerHRV, setLowerHRV] = useState<number>(16);
  const [notifications, setNotifications] = useState<boolean>(false);

  const [changes, setChanges] = useState(false); // State for if any changes are present

  const [ready, setReady] = useState(false); // State if settings tab is ready

  const [message, setMessage] = useState(""); // State holding toast message to confirm submission

  const [present, dismiss] = useIonLoading(); // Loading box when loading settings

  // Load all settings from preferences
  const getSettings = async () => {
    present({message:"Loading Settings"}); // Show Loading messafe

    // Get all settings from preferences
    setPasscode((await Preferences.get({key:PASSCODE})).value || "");
    setUpperHRV(Number((await Preferences.get({key:UPPER_HRV})).value  || "107" ));
    setLowerHRV(Number((await Preferences.get({key:LOWER_HRV})).value  || "16" ));
    setNotifications(Boolean((await Preferences.get({key:NOTIFICATIONS})).value  || ""));

    // Set ready state and dismiss loading
    setReady(true);
    dismiss();
  };

  // On render, load all settings
  useEffect (() => {getSettings()}, []);

  // For each setting, on local update, update the corresponding preference.
  useEffect( () => {
    if (ready) Preferences.set({key:PASSCODE, value:passcode});
  }, [passcode]);

  useEffect( () => {
    if (ready) Preferences.set({key:UPPER_HRV, value:(upperHRV).toString()});
  }, [upperHRV]);

  useEffect( () => {
    if (ready) Preferences.set({key:LOWER_HRV, value:(lowerHRV).toString()});
  }, [lowerHRV]);

  useEffect( () => {
    if (ready){
      Preferences.set({key:NOTIFICATIONS, value:(notifications).toString()});

      if (notifications) {
        LocalNotifications.requestPermissions();
      }

    }



  }, [notifications]);

  useEffect(
    () => console.log(passcode, upperHRV, lowerHRV, notifications), [passcode, upperHRV, lowerHRV, notifications]
  )

  // TODO: ensure valid HRV threshold
  // Save local changes to preferences
  const saveChanges = () => {
    // Pull values from HTML elements and store in each state
    //@ts-ignore
    setNotifications(document.getElementById('notif-toggle').checked);
    //@ts-ignore
    setLowerHRV(document.getElementById('lower-input').focusedValue || lowerHRV);
    //@ts-ignore
    setUpperHRV(document.getElementById('upper-input').focusedValue || upperHRV);

    // Unset the changes state and send confimation message
    setChanges(false);
    setMessage("Saved Changes");

    // Clear fields
    //@ts-ignore
    document.getElementById('upper-input').focusedValue = "";
    //@ts-ignore
    document.getElementById('lower-input').focusedValue = "";
  }

  // Handle passcode changes
  const changePasscode = () => {

    // Get new code
    //@ts-ignore
    const newcode:string = document.getElementById('new-passcode-input').focusedValue || ""

    // If there is a passcode already, get the inputted old pass code
    var oldcode = "";
    if (passcode != "") {
      //@ts-ignore
      oldcode = document.getElementById('old-passcode-input').focusedValue || "";
    }

    // Check if the new code is valid
    if (newcode != "" && (newcode.length > 6 || newcode.length < 4)) {
      setMessage("Invalid New Passcode");
      return;
    }

    // Check if the old code matches the new code
    if (passcode != "" && passcode != oldcode as string) {
      setMessage("Incorrect Old Passcode");
      return;
    }

    // Set the passcode
    setPasscode(newcode);
    setMessage("Set New Passcode");
  }

  return (
    <IonPage>
      <IonHeader>
        <IonToolbar className='ion-text-center'>
          <IonTitle><h1>Settings</h1></IonTitle>
        </IonToolbar>
      </IonHeader>
      <IonContent fullscreen>

        <IonToast
          isOpen={message!=""}
          onDidDismiss={() => setMessage("")}
          duration={3000}
          message={message}
          position='top'
        ></IonToast>

        <IonGrid className="homepage">
          <IonRow className='setting'>
            <IonCol size='10' className="ion-text-start">
              <h2>Enable Notifications</h2>
            </IonCol>

            <IonCol size='2' className="ion-text-end">
              <IonToggle
                id='notif-toggle'
                defaultChecked={notifications}
                onIonChange={(event) => setChanges(true)}
              />
            </IonCol>
          </IonRow>

          <IonRow className='setting'>
            <IonCol size='10' className="ion-text-start">
              <h2>Lower Threshold</h2>
            </IonCol>

            <IonCol size='2' className="ion-text-end">
              <IonInput
                id='lower-input'
                placeholder={lowerHRV.toString()}
                onIonChange={(event) => setChanges(true)}
                type='number'
              />
            </IonCol>
          </IonRow>

          <IonRow className='setting'>
            <IonCol size='10' className="ion-text-start">
              <h2>Upper Threshold</h2>
            </IonCol>

            <IonCol size='2' className="ion-text-end">
              <IonInput
                id='upper-input'
                placeholder={upperHRV.toString()}
                onIonChange={(event) => setChanges(true)}
                type='number'
              />
            </IonCol>
          </IonRow>

          <IonRow className='setting'>
              <IonCol className='ion-text-center'>
                <IonButton
                  disabled={!changes}
                  onClick={saveChanges}
                  shape='round'
                  fill='outline'
                >
                  <h2>Save Changes</h2>
                </IonButton>
              </IonCol>
          </IonRow>

          <IonRow className='setting'>
            <IonCol size='5' className="ion-text-start">
              {passcode == ""? undefined:<IonInput
```

```
                placeholder="Old Passcode"
                inputMode="numeric"
                type="password"
                id="old-passcode-input"
              />}
              <IonInput
                placeholder='New Passcode'
                inputMode="numeric"
                type="password"
                id="new-passcode-input"
              />
            </IonCol>

            <IonCol size='7' className="ion-text-end">
              <IonButton
                onClick={changePasscode}
                shape='round'
                fill='outline'
              >
                <h2>Change Code</h2>
              </IonButton>
            </IonCol>
          </IonRow>

        </IonGrid>
      </IonContent>
    </IonPage>
  );
};

export default SettingsTab;

```

## qalbi\src\setupTests.ts

```
// jest-dom adds custom jest matchers for asserting on DOM nodes.
// allows you to do things like:
// expect(element).toHaveTextContent(/react/i)
// learn more: https://github.com/testing-library/jest-dom
import '@testing-library/jest-dom/extend-expect';

// Mock matchmedia
window.matchMedia = window.matchMedia || function() {
  return {
      matches: false,
      addListener: function() {},
      removeListener: function() {}
  };
};

```

## qalbi\src\theme\variables.css

```
/* Ionic Variables and Theming. For more info, please see:
http://ionicframework.com/docs/theming/ */

/** Ionic CSS Variables **/
:root {
  /** primary **/
  --ion-color-primary: #3880ff;
  --ion-color-primary-rgb: 56, 128, 255;
  --ion-color-primary-contrast: #ffffff;
  --ion-color-primary-contrast-rgb: 255, 255, 255;
  --ion-color-primary-shade: #3171e0;
  --ion-color-primary-tint: #4c8dff;

  /** secondary **/
  --ion-color-secondary: #3dc2ff;
  --ion-color-secondary-rgb: 61, 194, 255;
  --ion-color-secondary-contrast: #ffffff;
```

```css
  --ion-color-secondary-contrast-rgb: 255, 255, 255;
  --ion-color-secondary-shade: #36abe0;
  --ion-color-secondary-tint: #50c8ff;

  /** tertiary **/
  --ion-color-tertiary: #5260ff;
  --ion-color-tertiary-rgb: 82, 96, 255;
  --ion-color-tertiary-contrast: #ffffff;
  --ion-color-tertiary-contrast-rgb: 255, 255, 255;
  --ion-color-tertiary-shade: #4854e0;
  --ion-color-tertiary-tint: #6370ff;

  /** success **/
  --ion-color-success: #2dd36f;
  --ion-color-success-rgb: 45, 211, 111;
  --ion-color-success-contrast: #ffffff;
  --ion-color-success-contrast-rgb: 255, 255, 255;
  --ion-color-success-shade: #28ba62;
  --ion-color-success-tint: #42d77d;

  /** warning **/
  --ion-color-warning: #ffc409;
  --ion-color-warning-rgb: 255, 196, 9;
  --ion-color-warning-contrast: #000000;
  --ion-color-warning-contrast-rgb: 0, 0, 0;
  --ion-color-warning-shade: #e0ac08;
  --ion-color-warning-tint: #ffca22;

  /** danger **/
  --ion-color-danger: #eb445a;
  --ion-color-danger-rgb: 235, 68, 90;
  --ion-color-danger-contrast: #ffffff;
  --ion-color-danger-contrast-rgb: 255, 255, 255;
  --ion-color-danger-shade: #cf3c4f;
  --ion-color-danger-tint: #ed576b;

  /** dark **/
  --ion-color-dark: #222428;
  --ion-color-dark-rgb: 34, 36, 40;
  --ion-color-dark-contrast: #ffffff;
  --ion-color-dark-contrast-rgb: 255, 255, 255;
  --ion-color-dark-shade: #1e2023;
  --ion-color-dark-tint: #383a3e;

  /** medium **/
  --ion-color-medium: #92949c;
  --ion-color-medium-rgb: 146, 148, 156;
  --ion-color-medium-contrast: #ffffff;
  --ion-color-medium-contrast-rgb: 255, 255, 255;
  --ion-color-medium-shade: #808289;
  --ion-color-medium-tint: #9d9fa6;

  /** light **/
  --ion-color-light: #f4f5f8;
  --ion-color-light-rgb: 244, 245, 248;
  --ion-color-light-contrast: #000000;
  --ion-color-light-contrast-rgb: 0, 0, 0;
  --ion-color-light-shade: #d7d8da;
  --ion-color-light-tint: #f5f6f9;
}

@media (prefers-color-scheme: dark) {
  /*
   * Dark Colors
   * -------------------------------------------
   */

  body {
    --ion-color-primary: #428cff;
    --ion-color-primary-rgb: 66,140,255;
    --ion-color-primary-contrast: #ffffff;
    --ion-color-primary-contrast-rgb: 255,255,255;
    --ion-color-primary-shade: #3a7be0;
    --ion-color-primary-tint: #5598ff;

    --ion-color-secondary: #50c8ff;
    --ion-color-secondary-rgb: 80,200,255;
    --ion-color-secondary-contrast: #ffffff;
    --ion-color-secondary-contrast-rgb: 255,255,255;
    --ion-color-secondary-shade: #46b0e0;
    --ion-color-secondary-tint: #62ceff;

    --ion-color-tertiary: #6a64ff;
    --ion-color-tertiary-rgb: 106,100,255;
    --ion-color-tertiary-contrast: #ffffff;
    --ion-color-tertiary-contrast-rgb: 255,255,255;
    --ion-color-tertiary-shade: #5d58e0;
    --ion-color-tertiary-tint: #7974ff;

    --ion-color-success: #2fdf75;
    --ion-color-success-rgb: 47,223,117;
    --ion-color-success-contrast: #000000;
    --ion-color-success-contrast-rgb: 0,0,0;
    --ion-color-success-shade: #29c467;
    --ion-color-success-tint: #44e283;

    --ion-color-warning: #ffd534;
    --ion-color-warning-rgb: 255,213,52;
    --ion-color-warning-contrast: #000000;
    --ion-color-warning-contrast-rgb: 0,0,0;
    --ion-color-warning-shade: #e0bb2e;
    --ion-color-warning-tint: #ffd948;

    --ion-color-danger: #ff4961;
    --ion-color-danger-rgb: 255,73,97;
    --ion-color-danger-contrast: #ffffff;
    --ion-color-danger-contrast-rgb: 255,255,255;
    --ion-color-danger-shade: #e04055;
    --ion-color-danger-tint: #ff5b71;

    --ion-color-dark: #f4f5f8;
    --ion-color-dark-rgb: 244,245,248;
    --ion-color-dark-contrast: #000000;
    --ion-color-dark-contrast-rgb: 0,0,0;
    --ion-color-dark-shade: #d7d8da;
    --ion-color-dark-tint: #f5f6f9;

    --ion-color-medium: #989aa2;
    --ion-color-medium-rgb: 152,154,162;
    --ion-color-medium-contrast: #000000;
    --ion-color-medium-contrast-rgb: 0,0,0;
    --ion-color-medium-shade: #86888f;
    --ion-color-medium-tint: #a2a4ab;

    --ion-color-light: #222428;
    --ion-color-light-rgb: 34,36,40;
    --ion-color-light-contrast: #ffffff;
    --ion-color-light-contrast-rgb: 255,255,255;
    --ion-color-light-shade: #1e2023;
    --ion-color-light-tint: #383a3e;
  }

  /*
   * iOS Dark Theme
   * -------------------------------------------
   */

  .ios body {
    --ion-background-color: #000000;
    --ion-background-color-rgb: 0,0,0;

    --ion-text-color: #ffffff;
    --ion-text-color-rgb: 255,255,255;

    --ion-color-step-50: #0d0d0d;
    --ion-color-step-100: #1a1a1a;
    --ion-color-step-150: #262626;
    --ion-color-step-200: #333333;
    --ion-color-step-250: #404040;
    --ion-color-step-300: #4d4d4d;
```

```css
    --ion-color-step-350: #595959;
    --ion-color-step-400: #666666;
    --ion-color-step-450: #737373;
    --ion-color-step-500: #808080;
    --ion-color-step-550: #8c8c8c;
    --ion-color-step-600: #999999;
    --ion-color-step-650: #a6a6a6;
    --ion-color-step-700: #b3b3b3;
    --ion-color-step-750: #bfbfbf;
    --ion-color-step-800: #cccccc;
    --ion-color-step-850: #d9d9d9;
    --ion-color-step-900: #e6e6e6;
    --ion-color-step-950: #f2f2f2;

    --ion-item-background: #000000;

    --ion-card-background: #1c1c1d;
  }

  .ios ion-modal {
    --ion-background-color: var(--ion-color-step-100);
    --ion-toolbar-background: var(--ion-color-step-150);
    --ion-toolbar-border-color: var(--ion-color-step-250);
  }


  /*
   * Material Design Dark Theme
   * -------------------------------------------
   */

  .md body {
    --ion-background-color: #121212;
    --ion-background-color-rgb: 18,18,18;

    --ion-text-color: #ffffff;
    --ion-text-color-rgb: 255,255,255;

    --ion-border-color: #222222;

    --ion-color-step-50: #1e1e1e;
    --ion-color-step-100: #2a2a2a;
    --ion-color-step-150: #363636;
    --ion-color-step-200: #414141;
    --ion-color-step-250: #4d4d4d;
    --ion-color-step-300: #595959;
    --ion-color-step-350: #656565;
    --ion-color-step-400: #717171;
    --ion-color-step-450: #7d7d7d;
    --ion-color-step-500: #898989;
    --ion-color-step-550: #949494;
    --ion-color-step-600: #a0a0a0;
    --ion-color-step-650: #acacac;
    --ion-color-step-700: #b8b8b8;
    --ion-color-step-750: #c4c4c4;
    --ion-color-step-800: #d0d0d0;
    --ion-color-step-850: #dbdbdb;
    --ion-color-step-900: #e7e7e7;
    --ion-color-step-950: #f3f3f3;

    --ion-item-background: #1e1e1e;

    --ion-toolbar-background: #1f1f1f;

    --ion-tab-bar-background: #1f1f1f;

    --ion-card-background: #1e1e1e;
  }
}

```

## qalbi\src\vite-env.d.ts

```
/// <reference types="vite/client" />

```