

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

```
1 import React from 'react';
2 import { render } from '@testing-library/react';
3 import AppRoute from './App';
4
5 test('renders without crashing', () => {
6   const { baseElement } = render(<AppRoute />);
7   expect(baseElement).toBeDefined();
8 });
9
```

qalbi\src\App.tsx

```
1 import { Redirect, Route } from 'react-router-dom';
2 import {
3   IonApp,
4   IonButton,
5   IonFab,
6   IonFabButton,
7   IonIcon,
8   IonInput,
9   IonRouterOutlet,
10  IonTabBar,
11  IonTabButton,
12  IonTabs,
13  IonText,
14  IonToast,
15  setupIonicReact,
16  useIonLoading,
17 } from '@ionic/react';
18 import { IonReactRouter } from '@ionic/react-router';
19 import { analyticsOutline, bluetoothOutline, bookmarksOutline, homeOutline, lockOpenOutline, settingsOutline } from 'ionicons/icons';
20 import GraphTab from './pages/GraphTab';
21 import AdviceTab from './pages/AdviceTab';
22 import SettingsTab from './pages/SettingsTab';
23 import HomeScreen from './pages/HomeScreen';
24 import { useState, useEffect } from 'react';
```

```

25 import { dataHook, fetchRecords } from './hooks/DataHook';
26 import { FileSystem, Encoding, Directory } from '@capacitor/filesystem';
27 import { Preferences } from '@capacitor/preferences';
28 import { LocalNotifications, LocalNotificationSchema } from '@capacitor/local-notifications';
29
30 /* Core CSS required for Ionic components to work properly */
31 import '@ionic/react/css/core.css';
32
33 /* Basic CSS for apps built with Ionic */
34 import '@ionic/react/css/normalize.css';
35 import '@ionic/react/css/structure.css';
36 import '@ionic/react/css/typography.css';
37
38 /* Optional CSS utils that can be commented out */
39 import '@ionic/react/css/padding.css';
40 import '@ionic/react/css/float-elements.css';
41 import '@ionic/react/css/text-alignment.css';
42 import '@ionic/react/css/text-transformation.css';
43 import '@ionic/react/css/flex-utils.css';
44 import '@ionic/react/css/display.css';
45
46 /* Theme variables */
47 import './theme/variables.css';
48 setupIonicReact();
49
50 const LOWER_HRV = "lower_hrv";
51 const UPPER_HRV = "upper_hrv";
52 const PASSCODE = "passcode";
53 const NOTIFICATIONS = "notifications";
54
55 /*
56 * React Functional Component responsible for setting up global states and creating the routing for the device android application.
57 */
58 const AppRoute: React.FC = () => {
59   /*
60    // Test Code!
61    const dumpHrv = async () => {
62      const {files} = await FileSystem.readdir({
63        path:"",
64        directory:Directory.Data
65      });
66
67      console.log(files.map(file => file.name))
68
69      files.forEach(async (file) => {
70        if (file.name.includes("HRV")) {
71          await FileSystem.deleteFile({
72            path: file.name,
73            directory: Directory.Data
74          })
75        }
76      });
77    }
78
79    const testHrv = async () => {
80
81      const buffer = new ArrayBuffer(8);
82      const view = new DataView(buffer);
83      const start = Date.now();
84
85      for (var i = 0; i < 1000; i++) {
86        const timestamp = (start - (Math.random()*(60480000-1000) + 1000))/1000
87        const rmssd = Math.random()*50 + 70;
88
89        view.setUint32(0, timestamp, true);
90        view.setFloat32(4, rmssd, true);
91
92        await storeRecord(view);
93      }
94
95      await determineUserState();
96    };
97
98    */
99
100    // On startup, load the passcode
101    useEffect(() => {
102      loadPasscode();
103    }, []);
104
105    // userState is the user's current stress state

```

```

106 // 0 : normal
107 // -1: fatigued
108 // 1: stressed
109 const [stressState, setStressState] = useState<number>(0);
110 const [connected, setConnected] = useState(false);
111
112 /*
113  * Callback to trigger whenever a new record is written to the HRV characteristic by the device wearable.
114  * rawRecord is the received value from the HRV characteristic
115  */
116 const hrvCallback = async (rawRecord: DataView): Promise<void> => {
117     await storeRecord(rawRecord);
118     await determineUserState();
119 };
120
121 /*
122  * Stores an HRV record given by the raw bits of rawRecord.
123  * rawError is the received value from the HRV characteristic, following the format in design specification 4.5.1.
124  */
125 const storeRecord = async (rawRecord: DataView): Promise<void> => {
126     /*
127      * Parse the HRV Characteristic
128      *
129      * UUUUHHHH
130      * U: Unix timestamp: uint32
131      * H: HRV metric: float32
132      *
133      * Arduino is little endian, so read and write to the DataView with the little endian flag set for multibyte datatypes.
134      */
135
136     const timestamp = rawRecord.getUint32(0, true);
137     const rmssd = rawRecord.getFloat32(4, true).toFixed(2).padStart(6, '0');
138
139     console.log(timestamp);
140     console.log(rmssd);
141
142     // Format the record
143     const record = `${timestamp} ${rmssd}\n`;
144
145     // Get the record's year, month, and day to make the HRV day data file filename
146     const currentDatetime = new Date(timestamp*1000);
147
148     const year = currentDatetime.getUTCFullYear().toString().padStart(4, '0');
149     const month = (currentDatetime.getUTCMonth()+1).toString().padStart(2, '0');
150     const day = currentDatetime.getUTCDate().toString().padStart(2, '0');
151
152     const filename = `HRV-${year}${month}${day}.txt`;
153
154     // Attempt to read the day data file.
155     try {
156         const contents = await FileSystem.readFile({
157             path: filename,
158             directory: Directory.Data,
159             encoding: Encoding.UTF8
160         });
161
162         // if the record is not already in the data file, append the record.
163         if (!contents.data.includes(record)){
164             await FileSystem.appendFile({
165                 path: filename,
166                 data: record,
167                 directory: Directory.Data,
168                 encoding: Encoding.UTF8
169             });
170         }
171     } catch (error) {
172
173         // Write to the day datafile to create a new file.
174         await FileSystem.writeFile({
175             path: filename,
176             data: record,
177             directory: Directory.Data,
178             encoding: Encoding.UTF8
179         });
180     }
181 }
182
183 /*
184  * 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)
185  */
186

```

```

187 const determineUserState = async (): Promise<void> => {
188     const baselineRecords = await fetchRecords(3 * 24 * 60 * 60); // Records from 3 days ago to now
189     const sampleRecords = await fetchRecords(3 * 60 * 60); // Records from 3 hours ago to now
190
191     // If there are no records, end method to avoid division by zero
192     if (baselineRecords.length <= 0 || sampleRecords.length <= 0){
193         return;
194     }
195
196     // Get the mean of each set of records to serve as the respective HRV metric.
197     const baselineHRV = baselineRecords.map((record) => record[1]).reduce((acc, curr) => acc + curr, 0) / baselineRecords.length;
198     const sampleHRV = sampleRecords.map((record) => record[1]).reduce((acc, curr) => acc + curr, 0) / sampleRecords.length;
199
200     // Get the current user set thresholds
201     const {value: rawUpperHrv} = await Preferences.get({key: UPPER_HRV});
202     const upperHRV = Number(rawUpperHrv || "200");
203
204     const {value: rawLowerHrv} = await Preferences.get({key: LOWER_HRV});
205     const lowerHRV = Number(rawLowerHrv || "0");
206
207     // If the user is "stressed", write 1 to the userState
208     // Else if the user is "fatigued", write -1 to the userState
209     // Else write 0 to the user state
210     console.log(sampleHRV);
211     console.log(baselineHRV);
212
213     if (sampleHRV > 107 || sampleHRV > 1.15 * baselineHRV || sampleHRV > upperHRV)
214         setStressState(1);
215     else if (sampleHRV < 16 || sampleHRV < 0.85 * baselineHRV || sampleHRV < lowerHRV)
216         setStressState(-1)
217     else
218         setStressState(0);
219 }
220
221 // Send notification on stress state change
222 useEffect(()=> {handleStressChange()}, [stressState]);
223 const handleStressChange = async () => {
224     const notificationsOn = Boolean(await Preferences.get({key:NOTIFICATIONS})).value);
225
226     if ((await LocalNotifications.checkPermissions()).display != 'granted'){
227         await LocalNotifications.requestPermissions();
228     }
229
230     if (notificationsOn) {
231
232         var header:string = "";
233         var message:string = "";
234
235         if (stressState == 0) {
236             header = "Great job with your stress management!"
237             message = "Tranquil+ detected that your stress levels are good! Keep it up!"
238         }
239         else if (stressState == -1) {
240             header = "You might be a bit fatigued."
241             message = "Try taking a break for a bit to catch some rest!"
242         }
243         else {
244             header = "You might be a bit stress."
245             message = "Try checking out the advice section in the Tranquil+ app to get some activities to destress!"
246         }
247
248         const notification:LocalNotificationSchema = {
249             title: header,
250             body: message,
251             id:1,
252         };
253         LocalNotifications.schedule({notifications: [notification]});
254     }
255 }
256
257 var readjustError = 0; // number of readjust Errors in the last hour
258 var timeoutID: NodeJS.Timeout|undefined = undefined; // timeout to handle clearing readjustError
259
260 /*
261  * Handles error codes sent from the device wearable to the Error characteristic.
262  * rawError is the received value from the characteristic, following the format in design specification 4.5.1.
263  */
264 const errorCallback = async (rawError: DataView): Promise<void> => {
265
266     /*
267     * Parse the Error Characteristic

```

```

268 *
269 * Z
270 * Z: Error code: uint8
271 */
272
273 const errorCode = rawError.getUint8(0);
274
275 // If the errorCode is 1, a readjust error has occurred
276 if (errorCode == 1) {
277     readjustError += 1; // Increment readjust error counter
278
279     // If 5 readjustErrors have occurred in the last hour, set a local notification and reset the timeout
280     if (readjustError >= 5) {
281         handleErrorMessage();
282         clearTimeout(timeoutID);
283     }
284
285     // If this is the first readjust error, set a timeout to clear the readjustError value to 0 after an hour.
286     if (readjustError == 1){
287         timeoutID = setTimeout(
288             () => {
289                 readjustError = 0;
290                 timeoutID = undefined
291             },
292             60*60
293         )
294     }
295 }
296
297 // Log the error code
298 console.error('Error Code', errorCode);
299 }
300
301 // Send error message on fail to read
302 const handleErrorMessage = async () => {
303     const notificationsOn = Boolean((await Preferences.get({key:NOTIFICATIONS})).value);
304
305     if ((await LocalNotifications.checkPermissions()).display != 'granted'){
306         await LocalNotifications.requestPermissions();
307     }
308
309     if (notificationsOn) {
310         const notification:LocalNotificationSchema = {
311             title: "Device Misread",
312             body:"Your Tranquil+ device is not properly reading your heart rate! Try readjusting the glove fit so that the sensor rests on the fingertip.",
313             id:0,
314         };
315         LocalNotifications.schedule({notifications:[notification]});
316     }
317 }
318
319 // Log in page
320 const [loggedIn, setLoggedIn] = useState<boolean>(false); // Is user logged in?
321 const [present, dismiss] = useIonLoading(); // Loading for getting user passcode
322 const [passcode, setPasscode] = useState<string>(); // User passcode
323
324 // Load the passcode
325 const loadPasscode = async () => {
326     present("Getting Creds");
327     setPasscode((await Preferences.get({key:PASSCODE})).value || "");
328     dismiss();
329 }
330
331 // On passcode load, log in if there is no passcode
332 useEffect(() => {
333     if (passcode == "") {
334         setLoggedIn(true);
335     }
336
337     if (passcode != undefined)
338         dismiss();
339 }, [passcode])
340
341 useEffect(() => {if (loggedIn) dataHook([hrvCallback, errorCallback], () => setConnected(true), () => setConnected(false)), [loggedIn]}; // Start dataHook on login
342
343 const [failLogin, setFailLogin] = useState(false);
344
345 // Handle login on password submission
346 const handleLogin = () => {
347     //@ts-ignore
348     const pass:string = document.getElementById('passcode-input').focusedValue || "";

```

```

349 setLoggedIn(pass==passcode);
350 setFailLogin(pass!=passcode);
351 }
352
353 // Notifications on disconnect
354 const handleDisconnect = async () => {
355   const notification: LocalNotificationSchema = {
356     title: "Device Disconnected",
357     body: "Your Tranquil+ device disconnected! Start the Tranquil+ App to reconnect.",
358     id: 0,
359   };
360
361   // if the device disconnects, send a notification
362   if (!connected) {
363     const notificationsOn = Boolean((await Preferences.get({key: NOTIFICATIONS})).value);
364
365     if ((await LocalNotifications.checkPermissions()).display !== 'granted'){
366       await LocalNotifications.requestPermissions();
367     }
368
369     if (notificationsOn) {
370       LocalNotifications.schedule({notifications: [notification]});
371     }
372   }
373   else {
374     // Remove notification on reconnect
375     LocalNotifications.removeDeliveredNotifications({notifications: [notification]})
376   }
377 }
378
379 useEffect(() => {handleDisconnect()}, [connected]) // On connection status change, handle notifications
380
381 return (
382   <IonApp>
383     {!loggedIn? passcode===undefined? undefined: passcode===""? undefined:
384       <div className="react-lock-screen_ui">
385         <IonText>
386           >
387           <h1>Welcome</h1>
388         </IonText>
389
390         <IonToast>
391           isOpen={failLogin}
392           duration={3000}
393           message="Incorrect Password"
394           onDidDismiss={() => setFailLogin(false)}
395         >
396
397         </IonToast>
398
399         <IonInput>
400           placeholder="Enter Your Passcode."
401           id='passcode-input'
402           inputMode="numeric"
403           type="password"
404         />
405
406         <IonButton onClick={handleLogin} shape='round' className='ion-padding'>
407           <IonIcon icon={lockOpenOutline} size="large"/>
408         </IonButton>
409       </div>
410
411       :
412
413     <>
414       <IonReactRouter>
415         <IonTabs>
416           <IonRouterOutlet>
417             <Route exact path="/">
418               <Redirect to="/home" />
419             </Route>
420             <Route exact path="/tab1">
421               <GraphTab />
422             </Route>
423             <Route exact path="/tab2">
424               <AdviceTab />
425             </Route>
426             <Route exact path="/tab3">
427               <SettingsTab />
428             </Route>
429             <Route exact path="/home">

```

```

430         <HomeScreen stressState={stressState}/>
431     </Route>
432 </IonRouterOutlet>
433 <IonTabBar slot="bottom">
434     <IonTabButton tab="home" href="/home">
435         <IonIcon aria-hidden="true" icon={homeOutline} size='large' />
436     </IonTabButton>
437     <IonTabButton tab="tab1" href="/tab1">
438         <IonIcon aria-hidden="true" icon={analyticsOutline} size='large' />
439     </IonTabButton>
440     <IonTabButton tab="tab2" href="/tab2">
441         <IonIcon aria-hidden="true" icon={bookmarksOutline} size='large' />
442     </IonTabButton>
443     <IonTabButton tab="tab3" href="/tab3">
444         <IonIcon aria-hidden="true" icon={settingsOutline} size='large' />
445     </IonTabButton>
446 </IonTabBar>
447 </IonTabs>
448 </IonReactRouter>
449
450     <IonFab vertical="top" horizontal="end" slot="fixed">
451         <IonFabButton
452             color={connected? "primary": "danger"}
453             onClick={() => {console.log("Bluetooth Reconnect."); dataHook([hrvCallback, errorCallback], () => setConnected(true), () =>
setConnected(false))}}
454         >
455             <IonIcon aria-hidden="true" icon={bluetoothOutline} />
456         </IonFabButton>
457     </IonFab>
458 </>
459 }
460 </IonApp>
461 );
462 };
463
464 export default AppRoute;
465

```

galbi\src\components\ExploreContainer.css

```

1  .container {
2      text-align: center;
3      position: absolute;
4      left: 0;
5      right: 0;
6      top: 50%;
7      transform: translateY(-50%);
8  }
9
10 .container strong {
11     font-size: 20px;
12     line-height: 26px;
13 }
14
15 .container p {
16     font-size: 16px;
17     line-height: 22px;
18     color: #8c8c8c;
19     margin: 0;
20 }
21
22 .container a {
23     text-decoration: none;
24 }

```

galbi\src\components\ExploreContainer.tsx

```

1  import './ExploreContainer.css';
2
3  interface ContainerProps {
4      name: string;
5  }

```

```

6
7 const ExploreContainer: React.FC<ContainerProps> = ({ name }) => {
8   return (
9     <div className="container">
10      <strong>{name}</strong>
11      <p>Explore <a target="_blank" rel="noopener noreferrer" href="https://ionicframework.com/docs/components">UI Components</a></p>
12    </div>
13  );
14 };
15
16 export default ExploreContainer;
17

```

galbi\src\hooks\DataHook.tsx

```

1 import { BleClient, numberToUUID } from "@capacitor-community/bluetooth-le";
2 import { Preferences } from "@capacitor/preferences";
3 import { Directory, Filesystem, Encoding } from "@capacitor/filesystem";
4
5 const DEVICE_ID = "device_id" // Preference ID for Device Wearable ID.
6
7 // BLE Service and Characteristic UUIDs
8 const HRV_SERVICE = numberToUUID(0x180F); // Bluetooth Low Energy HRV Metric Service UUID
9 const HRV_CHARACTERISTIC = numberToUUID(0x2A19); // Bluetooth Low Energy Characteristic UUID (receive HRV records from device wearable)
10 const ERROR_CHARACTERISTIC = numberToUUID(0x2A1A); // Bluetooth Low Energy Characteristic UUID (receive error codes from device wearable)
11 const REQUEST_CHARACTERISTIC = numberToUUID(0x2A1B); // Bluetooth Low Energy Characteristic UUID (send data requests to device wearable)
12
13 /*
14  * Hook responsible for handling and maintaining connections with the device wearable.
15  * Takes in callbacks array, which assigns these functions to occur when their respective BLE characteristic is written to by the device wearable
16  */
17 export const dataHook = async (callbacks:Array<(value:DataView) => void>, onConnect:()=>void, onDisconnect:()=>void) => {
18   try {
19     await BleClient.initialize(); // Start the BleClient at the beginning of the program
20
21     // Disconnect from any previous connections
22     const connections = await BleClient.getConnectedDevices([HRV_SERVICE]);
23
24     connections.forEach(async (connection) => {
25       await BleClient.disconnect(connection.deviceId);
26     })
27
28     var connected = false; // Bluetooth connection state
29     var id: string; // BLE peripheral device id (the Device Wearable)
30
31     // While not connected:
32     do {
33
34       // Check the saved preference for a Device Wearable ID
35       const {value} = await Preferences.get({key: DEVICE_ID});
36
37       if (value) {
38         // If there is a Device Wearable ID stored in preferences, use the saved ID
39         id = value;
40       }
41       else {
42         // Else, Search for a device to connect to and save its ID to the preference.
43         const device = await BleClient.requestDevice({
44           services: [HRV_SERVICE]
45         })
46
47         id = device.deviceId;
48         await Preferences.set( {key: DEVICE_ID, value: id});
49       }
50     }
51
52     // Attempt to connect to the device wearable 5 times.
53     var attempts = 0;
54     while (!connected && attempts < 2) {
55       try {
56         await BleClient.connect(id, () => onDisconnect());
57         connected = true;
58
59       } catch (error) {
60         attempts++;
61       }
62     }
63

```



```

63
64     // If the Device Wearable fails to connect, clear the saved ID preference
65     if (!connected) {
66         await Preferences.remove({key: DEVICE_ID});
67     }
68
69     } while (!connected);
70
71     onConnect();
72
73     // Start Notifications for the HRV and Error Characteristics
74     // Updates to the HRV Characteristic should trigger the first callback
75     // Updates to the Error Characteristic should trigger the second callback
76     await BleClient.startNotifications(
77         id,
78         HRV_SERVICE,
79         HRV_CHARACTERISTIC,
80         callbacks[0]
81     );
82
83     await BleClient.startNotifications(
84         id,
85         HRV_SERVICE,
86         ERROR_CHARACTERISTIC,
87         callbacks[1]
88     );
89
90     // TODO: save the current last record read to effeciently recall records.
91     // Get the time stamp for last week
92     const lastWeekDate = new Date(Date.now() - 6.048e+8);
93
94     /*
95     * Pack the Request Characteristic into its format
96     *
97     * YYMD
98     * Y: Year: uint16
99     * M: Month: uint8
100    * D: Day: uint8
101    *
102    * Arduino is little endian, so read and write to the DataView with the little endian flag set for multibyte datatypes.
103    */
104
105    // Create an ArrayBuffer
106    const buffer = new ArrayBuffer(4);
107
108    // Create a DataView for bit manipulation
109    const view = new DataView(buffer);
110
111    // Format the request data into the DataView
112    view.setUint16(0, lastWeekDate.getUTCFullYear(), true);
113    view.setUint8(2, lastWeekDate.getUTCMonth()+1);
114    view.setUint8(3, lastWeekDate.getUTCDate());
115
116    // Write a Records Request to the Request Characteristic
117    await BleClient.writeWithoutResponse(
118        id,
119        HRV_SERVICE,
120        REQUEST_CHARACTERISTIC,
121        view
122    )
123
124    console.log("Done!")
125
126
127    } catch (error) {
128        // Log errors to console
129        console.error(error);
130
131        // TODO: Put Local notification
132
133        // Reset the dataHook after 30 seconds
134        //setTimeout(() => dataHook(callbacks), 30000);
135    }
136 }
137
138 /*
139 * Retrieves records from device application storage.
140 * Takes in an amount of seconds, which denotes how long ago to look back.
141 * Returns the found records in an array.
142 */
143 export const fetchRecords = async(timePeriod:number): Promise<number[][]> => {

```

```

144 var rawData = ""; // String holding the raw content of all the files the function reads.
145
146 const endingTimestamp = Date.now(); // Timestamp to stop search, the current timestamp
147
148 const startTimestamp = endingTimestamp - timePeriod*1000; // Starting timestamp is {timePeriod} seconds ago
149
150 console.log(new Date(startTimestamp));
151 console.log(new Date(endingTimestamp));
152
153 const {files} = await Filesystem.readdir({
154   path:"",
155   directory:Directory.Data
156 });
157
158 console.log("Bruh:", files.map(file => file.name))
159
160 var currentTimestamp = startTimestamp; // The current timestamp as the loop control variable.
161
162 while (currentTimestamp <= endingTimestamp) {
163   // Get the currentTimestamp's year, month, and day to make the HRV day data file filename
164   const currentDatetime = new Date(currentTimestamp);
165
166   const year = currentDatetime.getUTCFullYear().toString().padStart(4, '0');
167   const month = (currentDatetime.getUTCMonth()+1).toString().padStart(2, '0');
168   const day = currentDatetime.getUTCDate().toString().padStart(2, '0');
169
170   const filename = `HRV-${year}${month}${day}.txt`;
171
172   // Attempt to read from the current day data file
173   // If it exist, append its content to the raw data string
174   // Else, log FileNotFound error
175   try {
176     const contents = await Filesystem.readFile({
177       path: filename,
178       directory: Directory.Data,
179       encoding: Encoding.UTF8,
180     });
181     rawData += contents.data
182   } catch (error) {
183     console.error(`${filename} doesn't exist`)
184   }
185
186   // Increment timestamp by 1 day
187   currentTimestamp += 8.64e+7;
188 }
189
190 var records: number[][] = []; // Formatted Records array
191
192 // If there is data, parse it.
193 if (rawData != ""){
194
195   // Split each record (separated by newline character) and split numbers within each record (separated by a single space)
196   const allRecords = rawData.split("\n").map(element => element.split(' ').map(e => Number(e)));
197
198   // Remove records with timestamps outside of the range [startTimestamp, endingTimestamp]
199   records = allRecords.filter((row) => row[0] >= startTimestamp/1000 && row[0] <= endingTimestamp/1000);
200
201   // Sort records for convenience.
202   records.sort((record1, record2) => record1[0] - record2[1]);
203 }
204
205 console.log(records);
206 return records;
207 }
208

```

galbi\src\main.tsx

```

1 import React from 'react';
2 import { createRoot } from 'react-dom/client';
3 import AppRoute from './App';
4
5 const container = document.getElementById('root');
6 const root = createRoot(container!);
7 root.render(
8   <AppRoute />

```

```
9 | );
```

qalbi\src\pages\Advice.json

```
1 | {
2 |   "scent": [
3 |     "Lavender for tension release and sleep",
4 |     "Jasmine for relaxing tension",
5 |     "Vanilla for a sweeter relaxation",
6 |     "Ylang ylang to release negative moods",
7 |     "Peppermint for invigoration and focus",
8 |     "Lemon for calming down"
9 |   ],
10 |   "water": [
11 |     "Drink a glass of water slowly",
12 |     "Stay hydrated!"
13 |   ],
14 |   "quote": [
15 |     "Worry empties today of strength",
16 |     "Give your stress wings and let it fly away",
17 |     "Recognize that you are doing the best you can",
18 |     "You will survive whatever is coming, and thrive after"
19 |   ],
20 |   "breath": [
21 |     "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",
22 |     "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",
23 |     "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."
24 |   ]
25 | }
```

qalbi\src\pages\AdviceTab.css

```
1 | .popover {
2 |   white-space: pre-wrap;
3 | }
```

qalbi\src\pages\AdviceTab.tsx

```
1 | import { IonContent, IonHeader, IonPage, IonTitle, IonToolbar, IonIcon, IonButton, IonPopover, IonText, IonCol, IonGrid, IonRow } from '@ionic/react';
2 | import './AdviceTab.css';
3 | import { bulbOutline, chatbubblesOutline, flameOutline, pauseCircleOutline, waterOutline } from 'ionicons/icons';
4 | import { useEffect, useState } from 'react';
5 | import data from './Advice.json';
6 |
7 | /*
8 |  * React Functional Component responsible for creating the front end of the advice tab for the user.
9 |  */
10 | const AdviceTab: React.FC = () => {
11 |
12 |   const [type, setType] = useState<string>(""); // State for the type of advice selected
13 |   const [text, setText] = useState<string>(""); // State for the toast message
14 |
15 |   // On type selection, load a random piece of corresponding advice from the Advice json
16 |   useEffect( () => {
17 |     if (Object.keys(data).includes(type)){
18 |       //@ts-ignore
19 |       const strings = data[type]
20 |       setText(strings[Math.floor(Math.random()*strings.length)]);
21 |     }
22 |   }
23 |   , [type])
24 |
25 |   useEffect( () => {
26 |     if (text != "") {
27 |       console.log(text)
```

```

28     }
29   }, [text])
30
31   return (
32     <IonPage>
33       <IonHeader>
34         <IonToolbar className='ion-text-center'>
35           <IonTitle><h1>Advice</h1></IonTitle>
36         </IonToolbar>
37       </IonHeader>
38       <IonContent fullscreen>
39
40         <IonPopover
41           isOpen={text != ""}
42           onDidDismiss={() => {setText(""); setType("")}}
43         >
44           <div className='popover ion-text-center ion-padding'>
45             <IonIcon icon={bulbOutline}/>
46             <IonText>
47               <h2>{toTitleCase(type) + ":\n\n" + text}</h2>
48             </IonText>
49           </div>
50         </IonPopover>
51
52         <IonGrid className="homepage ion-text-center">
53           <h1>How would you like to relax?</h1>
54
55           <IonRow>
56             <IonCol>
57               <IonButton onClick={() => setType("scent")} shape="round">
58                 <IonIcon icon={flameOutline} size='large' />
59                 <h2> Scent</h2>
60               </IonButton>
61             </IonCol>
62           </IonRow>
63
64           <IonRow>
65             <IonCol>
66               <IonButton onClick={() => setType("water")} shape="round">
67                 <IonIcon icon={waterOutline} size='large' />
68                 <h2> Water</h2>
69               </IonButton>
70             </IonCol>
71           </IonRow>
72
73           <IonRow>
74             <IonCol>
75               <IonButton onClick={() => setType("quote")} shape="round">
76                 <IonIcon icon={chatbubblesOutline} size='large' />
77                 <h2> Quote</h2>
78               </IonButton>
79             </IonCol>
80           </IonRow>
81
82           <IonRow>
83             <IonCol>
84               <IonButton onClick={() => setType("breath")} shape="round">
85                 <IonIcon icon={pauseCircleOutline} size='large' />
86                 <h2> Breath</h2>
87               </IonButton>
88             </IonCol>
89           </IonRow>
90         </IonGrid>
91       </IonContent>
92     </IonPage>
93   );
94 };
95
96 // Make strings title case
97 function toTitleCase(str:string) {
98   return str.replace(
99     /\w\S*/g,
100     function(txt) {
101       return txt.charAt(0).toUpperCase() + txt.substr(1).toLowerCase();
102     }
103   );
104 };
105 }
106
107 export default AdviceTab;
108

```

qalbi\src\pages\GraphTab.css

```
1  .full {
2    width: 100%;
3    height: 100%;
4    margin: 0;
5  }
6
7  .homepage {
8    height: 100%;
9    display: flex;
10   align-items: center;
11   flex-direction: column;
12   justify-content: center;
13 }
```

qalbi\src\pages\GraphTab.tsx

```
1  import { IonButton, IonButtons, IonCard, IonCol, IonContent, IonGrid, IonHeader, IonPage, IonRow, IonTitle, IonToolbar, useIonLoading } from '@ionic/react';
2  import './GraphTab.css';
3  import { useState, useEffect } from 'react';
4
5  import { Scatter } from 'react-chartjs-2';
6  import {
7    Chart as ChartJS,
8    CategoryScale,
9    LinearScale,
10   PointElement,
11   LineElement,
12   Title,
13   Tooltip,
14   Legend,
15 } from 'chart.js';
16 import { fetchRecords } from '../hooks/DataHook';
17
18 import { Preferences } from '@capacitor/preferences';
19
20 // Preference IDs for user set HRV threshold values
21 const LOWER_HRV = "lower_hrv";
22 const UPPER_HRV = "upper_hrv";
23
24 ChartJS.register(
25   CategoryScale,
26   LinearScale,
27   PointElement,
28   LineElement,
29   Title,
30   Tooltip,
31   Legend
32 );
33
34 ChartJS.defaults.font.size = 16;
35
36 /*
37  * React Functional Component responsible for creating the front end of the graph tab for the user.
38  * Takes in userSettings as a prop to read the HRV thresholds
39  */
40 const GraphTab: React.FC = () => {
41
42   // Stateful variable for the current chart.js data
43   const [chartData, setChartData] = useState<any>({
44     labels: [],
45     datasets: []
46   });
47
48   // Stateful variable for the timeframe the user selects
49   // 0: 1 hour
50   // 1: 1 day
51   // 2: 1 week
52
53   const [timeframe, setTimeframe] = useState<number>(0);
54
55   const [present, dismiss] = useIonLoading(); // Loading box when getting graph data
```

```

56
57 /*
58  * Sets the graphData based on the HRV records that are within the current timeframe.
59  */
60 const getChartData = async (): Promise<void> => {
61     present("Loading Chart Data"); // Show loading box at start of function
62
63     var startTime = new Date(); // Start time to compare record timestamps to.
64     var timePeriod: number; // Number of seconds to look back for records
65
66     // Set timePeriod based on timeframe selection
67     if (timeframe == 0) {
68         timePeriod = 60 * 60; // 1 hour in seconds
69     }
70     else if (timeframe == 1) {
71         timePeriod = 24 * 60 * 60; // 1 day in seconds
72     }
73     else if (timeframe == 2) {
74         timePeriod = 7 * 24 * 60 * 60; // 1 week in seconds
75     }
76     else {
77         // If the timeframe is not 0, 1, or 2, throw an error
78         console.error("Invalid Timeframe Selection");
79         dismiss();
80         return;
81     }
82
83     const records = await fetchRecords(timePeriod); // Fetch the records for the corresponding time period
84
85     // If no records exist, end execution to avoid division by zero
86     if (records.length <= 0) {
87         console.error("No records")
88         setChartData({
89             labels: [],
90             datasets: []
91         });
92         dismiss();
93         return;
94     }
95
96     var divisor: number;
97     var multiplier: number = 1;
98     var timeunit:string;
99
100     // Split by 5 min intervals if past hour selected
101     if (timeframe == 0) {
102         divisor = 5*60*1000;
103         multiplier = 5;
104         timeunit = "minute";
105     }
106     // Split by even hour if past day selected
107     else if (timeframe == 1){
108         divisor = 2*60*60*1000;
109         multiplier = 2;
110         timeunit = "hour";
111     }
112     // Split by day if last week selected
113     else {
114         divisor = 24*60*60*1000;
115         timeunit = "day";
116     }
117
118     // variable to hold the labels for each time point
119     const labels = records.map((record) => {
120         const time = new Date(record[0]*1000);
121         //@ts-ignore
122         return Math.floor((startTime - time)/divisor) * multiplier;
123     })
124
125     const unique_labels = [... new Set(labels)]; // Pull unique labels
126
127     // Get friendly labels for each data point
128     const friendly_labels: string[] = unique_labels.map((value) => `${value} ${value == 1? timeunit:timeunit+"s"} ago`);
129
130     const values = new Array<number>(unique_labels.length); // Array to store aggregated values for each unique label
131
132     // For each unique label, store the average of all record HRV with the same label
133     unique_labels.forEach((label, i) => {
134         const vals = records.filter((_, i) => labels[i] == label).map((record) => record[1]);
135
136         values[i] = vals.reduce((acc, curr) => acc + curr, 0)/vals.length;

```

```

137 })
138
139 // Collect data points together in data object to pass as a data set
140 const aggregatedRecords = unique_labels.map((label, i) => ({x: label, y: values[i]}));
141
142 const colors = await colorRecords(values); // Get the colors according to their value.
143
144 // If no colors, throw error
145 if (colors.length <= 0) {
146   console.error("No colors")
147   setChartData({
148     labels: [],
149     datasets: []
150   });
151   dismiss();
152   return;
153 }
154
155 // Set chartData to a chart.js data object
156 const data = {
157   labels: friendly_labels,
158   datasets: [{
159     label: 'HRV',
160     data: aggregatedRecords,
161     fill: false,
162     borderColor: colors,
163     backgroundColor: colors,
164     tension: 0.1,
165     pointRadius: 10,
166     showLine: true
167   }]
168 };
169
170 setChartData(data);
171 dismiss();
172 console.log(data);
173 }
174
175 /*
176  * Assigns a color to each HRV record based on its HRV value.
177  * Takes in an array of HRV values.
178  * Returns an array of RGB values.
179  */
180 const colorRecords = async (values: number[]): Promise<String[]> => {
181   const baselineRecords = await fetchRecords(3 * 24 * 60 * 60); // Records from 3 days ago to now
182
183   // If there are no records, end method to avoid division by zero
184   if (baselineRecords.length <= 0) {
185     console.error("No records")
186     return [];
187   }
188
189   // Get baselineHRV from the average of the record's HRV values.
190   const baselineHRV = baselineRecords.map((record) => record[1]).reduce((acc, curr) => acc + curr, 0) / baselineRecords.length;
191
192   console.log(baselineHRV);
193
194   // Get the current user set thresholds
195   const {value: rawUpperHrv} = await Preferences.get({key: UPPER_HRV});
196   const userUpper = Number(rawUpperHrv || "200");
197
198   const {value: rawLowerHrv} = await Preferences.get({key: LOWER_HRV});
199   const userLower = Number(rawLowerHrv || "0");
200
201   // Colors corresponding to each record
202   const colors = values.map( (HRV) => {
203     // If the record is stressed or fatigued, map the record to a red color
204     if (HRV > 107 || HRV > 1.15 * baselineHRV || HRV > userUpper || HRV < 16 || HRV < 0.85 * baselineHRV || HRV < userLower)
205       return "#c46c7b";
206
207     // Else If the records is close to stressed or close to fatigued, map the record to a yellow color
208     if (HRV > 1.08 * baselineHRV || HRV < 0.92 * baselineHRV)
209       return "#f0d973";
210
211     // Else map the record to a green color
212     return "#91f2a6";
213   });
214
215   return colors;
216 }
217

```

```

218 useEffect(() => {getChartData()}, [timeframe]); // getChartData on startup and every timeframe change
219
220 return (
221   <IonPage>
222     <IonHeader>
223       <IonToolbar className='ion-text-center'>
224         <IonTitle><h1>HRV Readings</h1></IonTitle>
225       </IonToolbar>
226     </IonHeader>
227     <IonContent fullscreen>
228
229       <IonGrid className="homepage">
230
231         <IonRow style={{"flexGrow":1, "width":"100%}}>
232           <IonCol className='full'>
233             <IonCard className='full'>
234               <Scatter options={{
235                 responsive: true,
236                 maintainAspectRatio: false,
237                 plugins: {
238                   legend: {
239                     display: false
240                   },
241                 },
242                 scales : {
243                   x : {
244                     title: {
245                       display: true,
246                       text: timeframe == 0? "Minutes Ago": timeframe == 1? "Hours Ago" : "Days Ago"
247                     },
248                     max: timeframe == 0? 70: timeframe == 1? 25: 8,
249                     min: -1,
250                     reverse: true
251                   },
252                   y :{
253                     title : {
254                       display: true,
255                       text: "HRV"
256                     },
257                     // min: 40,
258                     // max: 140
259                   }
260                 }} data={chartData}/>
261             </IonCard>
262           </IonCol>
263         </IonRow>
264
265         <IonRow className='ion-text-center'>
266           <IonButtons>
267             <IonButton
268               fill={timeframe==0? "solid":"outline"}
269               onClick={() => setTimeframe(0)}
270               shape="round"
271               color="primary"
272             >
273               <h2>Past Hour</h2>
274             </IonButton>
275
276             <IonButton
277               fill={timeframe==1? "solid":"outline"}
278               onClick={() => setTimeframe(1)}
279               shape="round"
280               color="primary"
281             >
282               <h2>Today</h2>
283             </IonButton>
284
285             <IonButton
286               fill={timeframe==2? "solid":"outline"}
287               onClick={() => setTimeframe(2)}
288               shape="round"
289               color="primary"
290             >
291               <h2>This Week</h2>
292             </IonButton>
293           </IonButtons>
294         </IonRow>
295       </IonGrid>
296
297     </IonContent>
298   </IonPage>

```



```

299         </IonContent>
300     </IonPage>
301 );
302 };
303
304 export default GraphTab;
305

```

qalbi\src\pages\HomeScreen.css

```

1  .homepage {
2      height: 100%;
3      display: flex;
4      align-items: center;
5      flex-direction: column;
6      justify-content: center;
7  }
8
9  .react-lock-screen__ui {
10     width: 100vw;
11     height: 100vh;
12     display: flex;
13     align-items: center;
14     flex-direction: column;
15     justify-content: center;
16     text-align: center;
17 }
18
19 .lock {
20     filter: blur(100px);
21     height: 100vh;
22     overflow: hidden;
23 }
24
25 h1 {
26     font-size: 24pt !important;
27 }
28
29 h2 {
30     font-size: 12pt !important;
31 }

```

qalbi\src\pages\HomeScreen.tsx

```

1  import { IonCard, IonCol, IonContent, IonGrid, IonHeader, IonPage, IonRow, IonTitle, IonToolbar, useIonRouter } from "@ionic/react";
2  import './HomeScreen.css';
3
4  /*
5   * React Functional Component responsible for creating the front end of the home screen for the user.
6   * The content depends on the current userState.
7   */
8  const HomeScreen: React.FC<{stressState: number}> = ({stressState}) => {
9      return (
10         <IonPage>
11             <IonHeader>
12                 <IonToolbar className='ion-text-center'>
13                     <IonTitle><h1>Tranquil+</h1></IonTitle>
14                 </IonToolbar>
15             </IonHeader>
16             <IonContent fullscreen>
17                 <IonGrid className="homepage">
18
19                     <IonRow style={{ "flexGrow":1, "alignItems":"flex-end"}}>
20                         <IonCol>
21                             <h1>Your current stress level</h1>
22                         </IonCol>
23                     </IonRow>
24
25                     <IonRow style={{ "flexGrow":2, "alignItems":"flex-start"}}>
26                         <IonCol>
27                             <IonCard className="ion-padding">
28                                 <h1>{stressState == 1? "Stressed": stressState == -1? "Fatigued":"Normal"}</h1>

```

```

29         </IonCard>
30     </IonCol>
31 </IonRow>
32
33     </IonGrid>
34 </IonContent>
35 </IonPage>
36 );
37 }
38
39 export default HomeScreen;

```

qalbi\src\pages\SettingsTab.css

```

1 .setting {
2     width:85%;
3     align-items:center;
4     min-height: 66px;
5 }

```

qalbi\src\pages\SettingsTab.tsx

```

1 import { IonButton, IonCol, IonContent, IonGrid, IonHeader, IonInput, IonPage, IonRow, IonTitle, IonToast, IonToggle, IonToolbar, useIonLoading } from
    '@ionic/react';
2 import './SettingsTab.css';
3 import { Preferences } from '@capacitor/preferences';
4 import { useEffect, useState } from 'react';
5 import { LocalNotifications } from '@capacitor/local-notifications';
6
7 // Preference IDs for each setting
8 const LOWER_HRV = "lower_hrv";
9 const UPPER_HRV = "upper_hrv";
10 const PASSCODE = "passcode";
11 const NOTIFICATIONS = "notifications";
12
13 /*
14  * React Functional Component responsible for creating the front end of the settings tab for the user.
15  */
16 const SettingsTab: React.FC = () => {
17
18     // Component states for each setting
19     const [passcode, setPasscode] = useState<string>("");
20     const [upperHRV, setUpperHRV] = useState<number>(107);
21     const [lowerHRV, setLowerHRV] = useState<number>(16);
22     const [notifications, setNotifications] = useState<boolean>(false);
23
24     const [changes, setChanges] = useState(false); // State for if any changes are present
25
26     const [ready, setReady] = useState(false); // State if settings tab is ready
27
28     const [message, setMessage] = useState(""); // State holding toast message to confirm submission
29
30     const [present, dismiss] = useIonLoading(); // Loading box when loading settings
31
32     // Load all settings from preferences
33     const getSettings = async () => {
34         present({message:"Loading Settings"}); // Show Loading messafe
35
36         // Get all settings from preferences
37         setPasscode((await Preferences.get({key:PASSCODE})).value || "");
38         setUpperHRV(Number((await Preferences.get({key:UPPER_HRV})).value || "107" ));
39         setLowerHRV(Number((await Preferences.get({key:LOWER_HRV})).value || "16" ));
40         setNotifications(Boolean((await Preferences.get({key:NOTIFICATIONS})).value || ""));
41
42         // Set ready state and dismiss loading
43         setReady(true);
44         dismiss();
45     };
46
47     // On render, load all settings
48     useEffect (() => {getSettings()}, []);
49

```

```

50 // For each setting, on local update, update the corresponding preference.
51 useEffect( () => {
52   if (ready) Preferences.set({key:PASSCODE, value:passcode});
53 }, [passcode]);
54
55 useEffect( () => {
56   if (ready) Preferences.set({key:UPPER_HRV, value:(upperHRV).toString()});
57 }, [upperHRV]);
58
59 useEffect( () => {
60   if (ready) Preferences.set({key:LOWER_HRV, value:(lowerHRV).toString()});
61 }, [lowerHRV]);
62
63 useEffect( () => {
64   if (ready){
65     Preferences.set({key:NOTIFICATIONS, value:(notifications).toString()});
66
67     if (notifications) {
68       LocalNotifications.requestPermissions();
69     }
70
71   }
72
73
74
75 }, [notifications]);
76
77 useEffect(
78   () => console.log(passcode, upperHRV, lowerHRV, notifications), [passcode, upperHRV, lowerHRV, notifications]
79 )
80
81 // TODO: ensure valid HRV threshold
82 // Save local changes to preferences
83 const saveChanges = () => {
84   // Pull values from HTML elements and store in each state
85   //@ts-ignore
86   setNotifications(document.getElementById('notif-toggle').checked);
87   //@ts-ignore
88   setLowerHRV(document.getElementById('lower-input').focusedValue || lowerHRV);
89   //@ts-ignore
90   setUpperHRV(document.getElementById('upper-input').focusedValue || upperHRV);
91
92   // Unset the changes state and send confirmation message
93   setChanges(false);
94   setMessage("Saved Changes");
95
96   // Clear fields
97   //@ts-ignore
98   document.getElementById('upper-input').focusedValue = "";
99   //@ts-ignore
100  document.getElementById('lower-input').focusedValue = "";
101 }
102
103 // Handle passcode changes
104 const changePasscode = () => {
105
106   // Get new code
107   //@ts-ignore
108   const newcode:string = document.getElementById('new-passcode-input').focusedValue || ""
109
110   // If there is a passcode already, get the inputted old pass code
111   var oldcode = "";
112   if (passcode !== "") {
113     //@ts-ignore
114     oldcode = document.getElementById('old-passcode-input').focusedValue || "";
115   }
116
117   // Check if the new code is valid
118   if (newcode !== "" && (newcode.length > 6 || newcode.length < 4)) {
119     setMessage("Invalid New Passcode");
120     return;
121   }
122
123   // Check if the old code matches the new code
124   if (passcode !== "" && passcode !== oldcode as string) {
125     setMessage("Incorrect Old Passcode");
126     return;
127   }
128
129   // Set the passcode
130   setPasscode(newcode);

```

```

131     setMessage("Set New Passcode");
132 }
133
134 return (
135   <IonPage>
136     <IonHeader>
137       <IonToolbar className='ion-text-center'>
138         <IonTitle><h1>Settings</h1></IonTitle>
139       </IonToolbar>
140     </IonHeader>
141     <IonContent fullscreen>
142
143       <IonToast
144         isOpen={message!=""}
145         onDidDismiss={() => setMessage("")}
146         duration={3000}
147         message={message}
148         position='top'
149       ></IonToast>
150
151       <IonGrid className="homepage">
152         <IonRow className='setting'>
153           <IonCol size='10' className="ion-text-start">
154             <h2>Enable Notifications</h2>
155           </IonCol>
156
157           <IonCol size='2' className="ion-text-end">
158             <IonToggle
159               id='notif-toggle'
160               defaultChecked={notifications}
161               onChange={(event) => setChanges(true)}
162             />
163           </IonCol>
164         </IonRow>
165
166         <IonRow className='setting'>
167           <IonCol size='10' className="ion-text-start">
168             <h2>Lower Threshold</h2>
169           </IonCol>
170
171           <IonCol size='2' className="ion-text-end">
172             <IonInput
173               id='lower-input'
174               placeholder={lowerHRV.toString()}
175               onChange={(event) => setChanges(true)}
176               type='number'
177             />
178           </IonCol>
179         </IonRow>
180
181         <IonRow className='setting'>
182           <IonCol size='10' className="ion-text-start">
183             <h2>Upper Threshold</h2>
184           </IonCol>
185
186           <IonCol size='2' className="ion-text-end">
187             <IonInput
188               id='upper-input'
189               placeholder={upperHRV.toString()}
190               onChange={(event) => setChanges(true)}
191               type='number'
192             />
193           </IonCol>
194         </IonRow>
195
196         <IonRow className='setting'>
197           <IonCol className='ion-text-center'>
198             <IonButton
199               disabled={!changes}
200               onClick={saveChanges}
201               shape='round'
202               fill='outline'
203             >
204               <h2>Save Changes</h2>
205             </IonButton>
206           </IonCol>
207         </IonRow>
208
209         <IonRow className='setting'>
210           <IonCol size='5' className="ion-text-start">
211             {passcode == ""? undefined:<IonInput

```

```

212         placeholder="Old Passcode"
213         inputMode="numeric"
214         type="password"
215         id="old-passcode-input"
216     />
217     <IonInput
218         placeholder='New Passcode'
219         inputMode="numeric"
220         type="password"
221         id="new-passcode-input"
222     />
223 </IonCol>
224
225 <IonCol size='7' className="ion-text-end">
226     <IonButton
227         onClick={changePasscode}
228         shape='round'
229         fill='outline'
230     >
231         <h2>Change Code</h2>
232     </IonButton>
233 </IonCol>
234 </IonRow>
235
236 </IonGrid>
237 </IonContent>
238 </IonPage>
239 );
240 };
241
242 export default SettingsTab;
243
244

```

qalbi\src\setupTests.ts

```

1 // jest-dom adds custom jest matchers for asserting on DOM nodes.
2 // allows you to do things like:
3 // expect(element).toHaveTextContent(/react/i)
4 // learn more: https://github.com/testing-library/jest-dom
5 import '@testing-library/jest-dom/extend-expect';
6
7 // Mock matchmedia
8 window.matchMedia = window.matchMedia || function() {
9     return {
10         matches: false,
11         addListener: function() {},
12         removeListener: function() {}
13     };
14 };
15

```

qalbi\src\theme\variables.css

```

1 /* Ionic Variables and Theming. For more info, please see:
2 http://ionicframework.com/docs/theming/ */
3
4 /** Ionic CSS Variables **/
5 :root {
6     /** primary **/
7     --ion-color-primary: #3880ff;
8     --ion-color-primary-rgb: 56, 128, 255;
9     --ion-color-primary-contrast: #ffffff;
10    --ion-color-primary-contrast-rgb: 255, 255, 255;
11    --ion-color-primary-shade: #3171e0;
12    --ion-color-primary-tint: #4c8dff;
13
14    /** secondary **/
15    --ion-color-secondary: #3dc2ff;
16    --ion-color-secondary-rgb: 61, 194, 255;
17    --ion-color-secondary-contrast: #ffffff;








































```

```

18 --ion-color-secondary-contrast-rgb: 255, 255, 255;
19 --ion-color-secondary-shade: #36abe0;
20 --ion-color-secondary-tint: #50c8ff;
21
22 /** tertiary */
23 --ion-color-tertiary: #5260ff;
24 --ion-color-tertiary-rgb: 82, 96, 255;
25 --ion-color-tertiary-contrast: #ffffff;
26 --ion-color-tertiary-contrast-rgb: 255, 255, 255;
27 --ion-color-tertiary-shade: #4854e0;
28 --ion-color-tertiary-tint: #6370ff;
29
30 /** success */
31 --ion-color-success: #2dd36f;
32 --ion-color-success-rgb: 45, 211, 111;
33 --ion-color-success-contrast: #ffffff;
34 --ion-color-success-contrast-rgb: 255, 255, 255;
35 --ion-color-success-shade: #28ba62;
36 --ion-color-success-tint: #42d77d;
37
38 /** warning */
39 --ion-color-warning: #ffc409;
40 --ion-color-warning-rgb: 255, 196, 9;
41 --ion-color-warning-contrast: #000000;
42 --ion-color-warning-contrast-rgb: 0, 0, 0;
43 --ion-color-warning-shade: #e0ac08;
44 --ion-color-warning-tint: #ffc222;
45
46 /** danger */
47 --ion-color-danger: #eb445a;
48 --ion-color-danger-rgb: 235, 68, 90;
49 --ion-color-danger-contrast: #ffffff;
50 --ion-color-danger-contrast-rgb: 255, 255, 255;
51 --ion-color-danger-shade: #cf3c4f;
52 --ion-color-danger-tint: #ed576b;
53
54 /** dark */
55 --ion-color-dark: #222428;
56 --ion-color-dark-rgb: 34, 36, 40;
57 --ion-color-dark-contrast: #ffffff;
58 --ion-color-dark-contrast-rgb: 255, 255, 255;
59 --ion-color-dark-shade: #1e2023;
60 --ion-color-dark-tint: #383a3e;
61
62 /** medium */
63 --ion-color-medium: #92949c;
64 --ion-color-medium-rgb: 146, 148, 156;
65 --ion-color-medium-contrast: #ffffff;
66 --ion-color-medium-contrast-rgb: 255, 255, 255;
67 --ion-color-medium-shade: #808289;
68 --ion-color-medium-tint: #9d9fa6;
69
70 /** light */
71 --ion-color-light: #f4f5f8;
72 --ion-color-light-rgb: 244, 245, 248;
73 --ion-color-light-contrast: #000000;
74 --ion-color-light-contrast-rgb: 0, 0, 0;
75 --ion-color-light-shade: #d7d8da;
76 --ion-color-light-tint: #f5f6f9;
77 }
78
79 @media (prefers-color-scheme: dark) {
80 /**
81  * Dark Colors
82  * -----
83  */
84
85  body {
86    --ion-color-primary: #428cff;
87    --ion-color-primary-rgb: 66,140,255;
88    --ion-color-primary-contrast: #ffffff;
89    --ion-color-primary-contrast-rgb: 255,255,255;
90    --ion-color-primary-shade: #3a7be0;
91    --ion-color-primary-tint: #5598ff;
92
93    --ion-color-secondary: #50c8ff;

```

```

94  --ion-color-secondary-rgb: 80,200,255;
95  --ion-color-secondary-contrast:  #ffffff;
96  --ion-color-secondary-contrast-rgb: 255,255,255;
97  --ion-color-secondary-shade:  #46b0e0;
98  --ion-color-secondary-tint:  #62ceff;
99
100 --ion-color-tertiary:  #6a64ff;
101 --ion-color-tertiary-rgb: 106,100,255;
102 --ion-color-tertiary-contrast:  #ffffff;
103 --ion-color-tertiary-contrast-rgb: 255,255,255;
104 --ion-color-tertiary-shade:  #5d58e0;
105 --ion-color-tertiary-tint:  #7974ff;
106
107 --ion-color-success:  #2dfd75;
108 --ion-color-success-rgb: 47,223,117;
109 --ion-color-success-contrast:  #000000;
110 --ion-color-success-contrast-rgb: 0,0,0;
111 --ion-color-success-shade:  #29c467;
112 --ion-color-success-tint:  #44e283;
113
114 --ion-color-warning:  #ffd534;
115 --ion-color-warning-rgb: 255,213,52;
116 --ion-color-warning-contrast:  #000000;
117 --ion-color-warning-contrast-rgb: 0,0,0;
118 --ion-color-warning-shade:  #e0bb2e;
119 --ion-color-warning-tint:  #ffd948;
120
121 --ion-color-danger:  #ff4961;
122 --ion-color-danger-rgb: 255,73,97;
123 --ion-color-danger-contrast:  #ffffff;
124 --ion-color-danger-contrast-rgb: 255,255,255;
125 --ion-color-danger-shade:  #e04055;
126 --ion-color-danger-tint:  #ff5b71;
127
128 --ion-color-dark:  #f4f5f8;
129 --ion-color-dark-rgb: 244,245,248;
130 --ion-color-dark-contrast:  #000000;
131 --ion-color-dark-contrast-rgb: 0,0,0;
132 --ion-color-dark-shade:  #d7d8da;
133 --ion-color-dark-tint:  #f5f6f9;
134
135 --ion-color-medium:  #989aa2;
136 --ion-color-medium-rgb: 152,154,162;
137 --ion-color-medium-contrast:  #000000;
138 --ion-color-medium-contrast-rgb: 0,0,0;
139 --ion-color-medium-shade:  #86888f;
140 --ion-color-medium-tint:  #a2a4ab;
141
142 --ion-color-light:  #222428;
143 --ion-color-light-rgb: 34,36,40;
144 --ion-color-light-contrast:  #ffffff;
145 --ion-color-light-contrast-rgb: 255,255,255;
146 --ion-color-light-shade:  #1e2023;
147 --ion-color-light-tint:  #383a3e;
148 }
149
150 /*
151  * iOS Dark Theme
152  * -----
153  */
154
155 .ios body {
156   --ion-background-color:  #000000;
157   --ion-background-color-rgb: 0,0,0;
158
159   --ion-text-color:  #ffffff;
160   --ion-text-color-rgb: 255,255,255;
161
162   --ion-color-step-50:  #0d0d0d;
163   --ion-color-step-100:  #1a1a1a;
164   --ion-color-step-150:  #262626;
165   --ion-color-step-200:  #333333;
166   --ion-color-step-250:  #404040;
167   --ion-color-step-300:  #4d4d4d;

```

```

168 --ion-color-step-350: #595959;
169 --ion-color-step-400: #666666;
170 --ion-color-step-450: #737373;
171 --ion-color-step-500: #808080;
172 --ion-color-step-550: #8c8c8c;
173 --ion-color-step-600: #999999;
174 --ion-color-step-650: #a6a6a6;
175 --ion-color-step-700: #b3b3b3;
176 --ion-color-step-750: #bfbfbf;
177 --ion-color-step-800: #cccccc;
178 --ion-color-step-850: #d9d9d9;
179 --ion-color-step-900: #e6e6e6;
180 --ion-color-step-950: #f2f2f2;
181
182 --ion-item-background: #000000;
183
184 --ion-card-background: #1c1c1d;
185 }
186
187 .ios ion-modal {
188   --ion-background-color: var(--ion-color-step-100);
189   --ion-toolbar-background: var(--ion-color-step-150);
190   --ion-toolbar-border-color: var(--ion-color-step-250);
191 }
192
193
194 /*
195  * Material Design Dark Theme
196  * -----
197  */
198
199 .md body {
200   --ion-background-color: #121212;
201   --ion-background-color-rgb: 18,18,18;
202
203   --ion-text-color: #ffffff;
204   --ion-text-color-rgb: 255,255,255;
205
206   --ion-border-color: #222222;
207
208   --ion-color-step-50: #1e1e1e;
209   --ion-color-step-100: #2a2a2a;
210   --ion-color-step-150: #363636;
211   --ion-color-step-200: #414141;
212   --ion-color-step-250: #4d4d4d;
213   --ion-color-step-300: #595959;
214   --ion-color-step-350: #656565;
215   --ion-color-step-400: #717171;
216   --ion-color-step-450: #7d7d7d;
217   --ion-color-step-500: #898989;
218   --ion-color-step-550: #949494;
219   --ion-color-step-600: #a0a0a0;
220   --ion-color-step-650: #acacac;
221   --ion-color-step-700: #b8b8b8;
222   --ion-color-step-750: #c4c4c4;
223   --ion-color-step-800: #d0d0d0;
224   --ion-color-step-850: #dbdbdb;
225   --ion-color-step-900: #e7e7e7;
226   --ion-color-step-950: #f3f3f3;
227
228   --ion-item-background: #1e1e1e;
229
230   --ion-toolbar-background: #1f1f1f;
231
232   --ion-tab-bar-background: #1f1f1f;
233
234   --ion-card-background: #1e1e1e;
235 }
236 }
237

```

galbi\src\vite-env.d.ts


```
1 | /// <reference types="vite/client" />
2 |
```