Patient Health-Centred Social Network
Technical Document

Author: Adam Coakley

Supervisor: Hisain El Shaafi

Date: April 29th, 2022

Table of Contents

Table of Figures

1 Introduction

The purpose of this technical document is to initally outline the HealthSpace's application's implementation details. Initially, the technologies used to develop HealthSpace will be discussed. The project structure and the reason for the folder structure will be addressed shortly after that.

The third section of this manual will display both the admin site's and the mobile application's source code. As a result of potentially poor formatting, the source code can also be viewed on my GitHub account by following the link below.

GitHub Link:
https://github.com/Adamcoakley/HealthSpace

2 Technologies Used

The HealthSpace application was developed in Visual Studio Code using Javascript and the React Framework. The mobile application's user interface was built using React Native and the admin site's user interface was developed using ReactJS. The backend and most of the security was handled by Firebase. In the research report, all of these technologies were researched and documented and they will be breifly listed in the section below.

2.1 React Native

React Native was discovered to be an effective library for generating cross-platform applications at a faster rate than native development after extensive research. The productivity of the development increased when using React Native due to its reusable components and The HealthSpace application has a significant number of components. Another reason React Native was chosen was because of its great documentation and community.

2.2 React

For the same reasons as above React was chosen to develop the admin site. React is very similar to React Native, and using separate technologies to construct the admin site would have merely added to the project's complexity.

2.3 Firebase

In comparison to the other databases researched, Firebase was chosen because of the tools and services it offers.

- **Firebase Authentication -** Firebase Authentication is a Google Authentication functionality designed specifically for Firebase-based projects. It allows you to register users using custom credentials, emails, or any social account including but not limited to Google, Facebook and Apple. It also ensures that both the registration and login are extremely secure.
- **Firebase Storage –** This service was used to store all of the application's images. I would have to use another third-party service to store images if it weren't for Firebase Storage.

Not only that but Firebase also suited HealthSpace as a social network application by providing a real time database that stores data quickly and easily. This is necessary for majority of the features within a typical social network application, including but not limited to, sending messages and creating posts / updating the timeline.

3 Project Structure

There is no recommended way to structure React and React Native projects. However, having a clean file structure and codebase within your project can help you greatly. Being able to reference and even reuse components of your work can vastly speed up development and help you keep ahead of your deadline. Structuring the application's folders was one of the first steps of the development phase for HealthSpace.

3.1 Mobile Application

As HealthSpace was going to become a relatively large project with multiple reusable components and screens, the style sheets were kept internal for each file. One disadvantage of this is that some of the styling may have been unnecessarily reproduced but in return it almost halved the amount of potential files. As all of the core React Native components can make use of the property style, sometimes inline styling was used to reduce the amount of style objects within the stylesheets. This is demonstrated by figure 1 below.

```
1.  <TouchableOpacity onPress={submitPost}>
2.        <Text style={{color:'#000', fontWeight: 'bold'}}>Submit Post</Text>
3.  </TouchableOpacity>
```

Figure 1. Inline Styles.

Outside of that, the project followed a common structure by placing each of the applications screens, assets, configuration files, data files and helper files within their own folder. The contents of each folder is as follows:

- **Assets** – Contains all of the applications icons and images including the logo and loading screen.
- **Components** – Contains all of the applications reusable chunks of code.
- **Config** – Contains the application's firebase configuration file.
- **Data** – Contains all of the health conditions, including a short description, the symptoms you may be feeling and the possible treatments for each.
- **Screens** – Contains all of the user interface screens visible when using the application.
- **Utils** – Contains a dimensions helper file, making it easier to create a responsive application.

All of these folders are visible by figure 2 below.

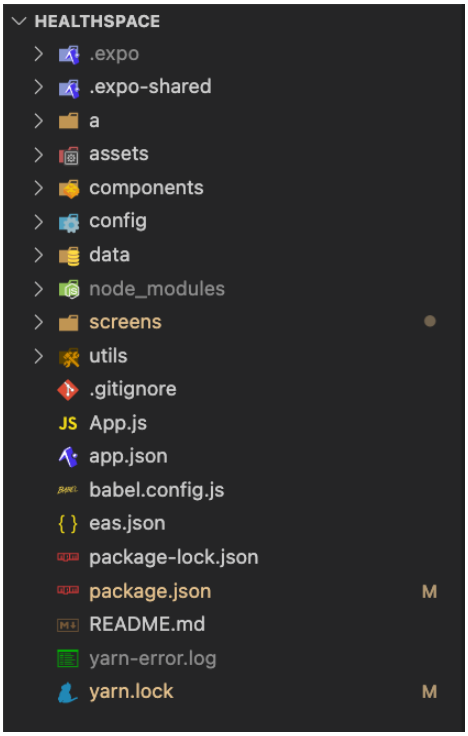

Figure 2. Mobile App - Project Structure.

3.2 Admin Site

The admin site, being a React application, followed a similar project structure as the mobile application. Despite being a smaller application overall, it is still a good idea to organise the project in case there are any future updates that make it larger. The contens of each folder are as follows:

- **Components** – Contains all of the applications reusable chunks of code.
- **Config** – Contains the application's firebase configuration file.
- **Pages** – Contains all of the user interface screens visible when using the application.

All of these folders are visible by figure 3 below

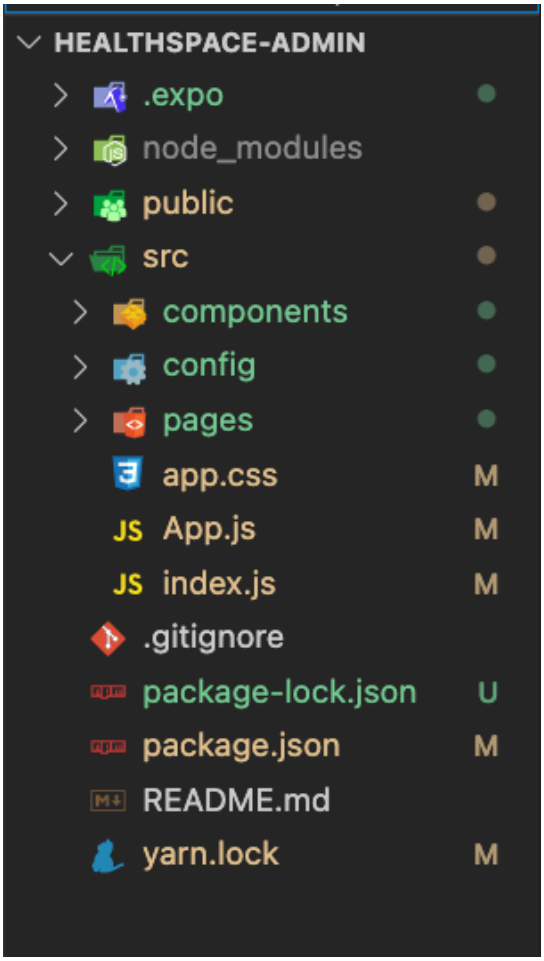

Figure 3. Admin Site - Project Structure.

4 Source Code

4.1 Mobile Application

4.1.1 Components

4.1.1.1 Post.js

```
import { Alert, View, Text, Image, TouchableOpacity, TextInput, StyleSheet} from
'react-native';
// divider between posts
import { Divider } from 'react-native-elements';
// icons
import Feather from 'react-native-vector-icons/Feather';
import AntDesign from 'react-native-vector-icons/AntDesign';
import Ionic from 'react-native-vector-icons/Ionicons';
import FontAwesome from 'react-native-vector-icons/FontAwesome';
// navigation to comment screen
import { useNavigation } from '@react-navigation/native';
import { Menu, MenuOptions, MenuOption, MenuTrigger } from 'react-native-
popup-menu';
// notifications
import Constants from 'expo-constants';
import * as Notifications from 'expo-notifications';
import * as Device from 'expo-device';
// firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const PostHeader = ({ post }) => {

  // function to update the post's report count when the report button is pressed
  const reportPost = () => {
    db.collection('posts').doc(post.id).update({
      report_count: post.report_count + 1,
    }).then(() => {
      // then alert the user that the report has been submitted
      Alert.alert(
        "Report Submitted",
        "We will review it shortly.",
        [{ text: "OK" }],
        { cancelable: false }
      );
    }).catch((error) => {
      console.log('Error updating report count: ', error);
    }
  );
```

```
  }

  // function navigate to another user's profile
  const navigateToAnotherUser = () => {
    if (post.user_id === auth.currentUser.uid) {
      // do nothing if the user is viewing their own profile
    } else{
      // pass the params to the other user's screen
      navigation.navigate('Profile', {profile_picture: post.profile_picture, name:
post.name, user_id: post.user_id});
    }
  }

  // function to delete a post
  const deletePost = () => {
    db.collection('posts').doc(post.id).delete().then(() => {
      // then alert the user that the post has been deleted
      Alert.alert(
        "Success!",
        "Your post has been deleted.",
        [{ text: "OK" }],
        { cancelable: false }
      );
    }).catch((error) => {
      console.log('Error deleting post: ', error);
    }
  );
  }

  // check if the user is the creator of the post
  const isCreator = () => {
    // if they are they can report and delete the post
    if (post.user_id === auth.currentUser.uid) {
      return (
        <MenuOptions customStyles={optionsStyles}>
          <MenuOption text='Report' customStyles={optionStyles}
onSelect={reportPost} />
          <MenuOption text='Delete' customStyles={optionStyles}
onSelect={deletePost} />
        </MenuOptions>
      )
    } else {
      // if they are not they can only report the post
```

```
    return (
      <MenuOptions customStyles={optionsStyles}>
        <MenuOption text='Report' customStyles={optionStyles}
onSelect={reportPost} />
      </MenuOptions>
    )
  }
 }

 const navigation = useNavigation();
 return(
 <View style={styles.header}>
  <View style={{flexDirection: 'row', alignItems: 'center'}}>
   {/* Navigate to another user's screen on profile picture or name press */}
   <TouchableOpacity onPress={navigateToAnotherUser}>
    <Image source={{url: post.profile_picture}} style={styles.profile_picture}/>
   </TouchableOpacity>
   {/* Padding to create space between the profile picture and the user's name */}
   <View style={{paddingLeft: 10}}>
    <TouchableOpacity onPress={navigateToAnotherUser}>
     <Text style={{fontSize: 15, fontWeight: '500',}}>{post.name}</Text>
    </TouchableOpacity>
   </View>
  </View>
  {/* Allow users to report posts */}
  <Menu>
  <MenuTrigger>
   <Feather name="more-vertical" style={{fontSize: 20}}/>
  </MenuTrigger>
  {isCreator()}
  </Menu>
 </View>
)}

const Post = ({ post }) => {
 const navigation = useNavigation();
 // state variables
 const [like, setLike] = useState(false);
 const [numOfLikes, setNumOfLikes] = useState(0);
 const [user, setUser] = useState('');

 // get the currrent user
 const getUser = async () => {
```

```
  db.collection('users')
  .doc(firebase.auth().currentUser.uid)
  .get()
  .then((documentSnapshot) => {
    if( documentSnapshot.exists ) {
      setUser(documentSnapshot.data());
    }
  })
}

// check to see if the current user has liked the post
const checkLike = () => {
  db.collection('likes')
    .doc(post.id)
    .collection('likedBy')
    .doc(auth.currentUser.uid)
    .get()
    .then((doc) => {
      if (doc.exists) {
        setLike(true);
      } else {
        setLike(false);
      }
    })
    .catch((error) => {
      console.log('Error getting document: ', error);
    });
}

// function to handle like button press
const handleLike = () => {
  // if the user has liked the post, unlike the post
  if (like) {
    db.collection('likes')
    .doc(post.id)
    .collection('likedBy')
    .doc(auth.currentUser.uid)
    .delete()
    .then(() => {
      setLike(false);
      setNumOfLikes(numOfLikes - 1);
    }).catch((error) => {
      console.log('Error removing like: ', error);
```

```
      });
    } else {
      // if the user has not liked the post, like the post
      db.collection('likes')
      .doc(post.id)
      .collection('likedBy')
      .doc(auth.currentUser.uid)
      .set({})
      .then(() => {
        setLike(true);
        setNumOfLikes(numOfLikes + 1);
      }).catch((error) => {
        console.log('Error adding like: ', error);
      });

      // if a token exists, send a notification to the user
      if(post.token) {
        sendPushNotification(post.token);
        console.log('Notification sent to: ' + post.token);
      }

  // store the notification in the database for ui purposes
  // including the document id so that the notification can be deleted later if needed
const newDocRef =
db.collection('notifications').doc(post.user_id).collection('allNotifications').doc();
      newDocRef.set({
        profile_picture: user.profile_picture,
        name: user.name,
        text: 'liked your post',
        time: firebase.firestore.FieldValue.serverTimestamp(),
        id: newDocRef.id
      })
    }
  }

  // function to get the posts number of likes
  const getNumOfLikes = () => {
    db.collection('likes')
    .doc(post.id)
    .collection('likedBy')
    .get()
    .then((querySnapshot) => {
      setNumOfLikes(querySnapshot.size);
```

```
  }).catch((error) => {
    console.log('Error getting number of likes: ', error);
  });
 }

 // send a push notification when the post is liked
 const sendPushNotification = async (token) => {
  const message = {
    to: token,
    sound: 'default',
    title: 'HealthSpace',
    body: user.name + ' liked your post.',
    data: { someData: 'goes here' },
  };

  await fetch('https://exp.host/--/api/v2/push/send', {
    method: 'POST',
    headers: {
      Accept: 'application/json',
      'Accept-encoding': 'gzip, deflate',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(message),
  });
 }

 useEffect(async () => {
  await getUser();
  await getNumOfLikes();
  await checkLike();
 }, []);

 return (
  <View>
    <View style={styles.container}>
      <PostHeader post={post}/>
      {/* User's text from the post */}
      <View style={styles.textContainer}>
      <Text>{post.text}</Text>
      </View>
      {/* If there is no image posted, there is no need to create space for an image
*/}
      <View>
```

```
        {post.image != null ? (
          <Image source={{uri: post.image}} style={{width: '100%', height: 200,
marginTop: 15,}}/>
          ) : ( <View /> )}
        </View>
        {/* Icon and likes container */}
        <View
          style={{
            flexDirection: 'row',
            justifyContent: 'space-between',
            alignItems: 'center',
            paddingHorizontal: 15,
            paddingVertical: 15,
          }}>
          {/* Icon and likes count -  */}
          <View style={{flexDirection: 'row', alignItems: 'center'}}>
          <AntDesign name='like2' style={{fontSize: 15, paddingRight: 2, color:
'#0C4BAE'}}/>
            <Text style={{fontSize: 12}}>
            Liked by{like ? ' you and' : ''}{' '}
            {like ? numOfLikes -1: numOfLikes} others {/* -1 to remove your own like
once liked */}
            </Text>
          </View>
        </View>
        {/* Like and comment section */}
        <Divider width={0.3} style={{marginBottom: 8}}/>
        <View style={styles.likeAndComment}>
          <View style={{ flex: 1, alignItems: 'center', flexDirection: "row",
justifyContent: 'center'}}>
          <TouchableOpacity onPress={handleLike}>
            <AntDesign  name={like ? 'like1' : 'like2'}
            style={{
                paddingRight: 2,
                fontSize: 15,
                color: like ? '#0C4BAE' : 'black', }}/>
            </TouchableOpacity>
            <TouchableOpacity onPress={handleLike}>
              <Text style={{ color: like ? '#0C4BAE' : '#000', fontWeight: '400'}}> Like
</Text>
            </TouchableOpacity>
          </View>
```

```
        <View style={{ flex: 1, alignItems: 'center', flexDirection: "row",
justifyContent: 'center'}}>
          {/* on press navigate to the comment section, passing the posts details  */}
            <TouchableOpacity onPress={() => navigation.navigate('Comments',
{profile_picture: post.profile_picture, name: post.name, text: post.text, like: like,
likes: numOfLikes, image: post.image, id: post.id, user_id: post.user_id, token:
post.token})}>
              <FontAwesome name='comment-o' style={{fontSize: 15, paddingRight:
2}}/>
            </TouchableOpacity>
            <TouchableOpacity onPress={() => navigation.navigate('Comments',
{profile_picture: post.profile_picture, name: post.name, text: post.text, like: like,
likes: numOfLikes, image: post.image, id: post.id, user_id: post.user_id, token:
post.token})}>
              <Text style={{fontWeight: '400'}}> Comment </Text>
            </TouchableOpacity>
        </View>
       </View>
       <Divider width={8} style={{paddingTop: 8}}/>
     </View>
   </View>
  );
};

export default Post;

const optionsStyles = {
  optionsContainer: {
    backgroundColor: '#fff',
    padding: 5,
    width: '20%',
  }
};

const optionStyles = {
  optionWrapper: {
    backgroundColor: '#fff',
  },
  optionText: {
    fontSize: 15,
    color: '#000',
    fontWeight: '600',
  }
```

```
};

const styles = StyleSheet.create({
   container: {
     paddingBottom: 10,
     borderBottomColor: 'gray',
     borderBottomWidth: 0.1,
   },
   header: {
     flexDirection: 'row',
     alignItems: 'center',
     justifyContent: 'space-between',
     padding: 15,
   },
   profile_picture:{
      width: 40,
      height: 40,
      borderRadius: 100
   },
   textContainer:{
     marginHorizontal: 15,
   },
   likeAndComment:{
     flex: 1,
     flexDirection: "row",
   }
});
```

Comment.js

```
import {View, Text, Image, TouchableOpacity, TextInput, StyleSheet} from 'react-
native';
// divider between comments
import { Divider } from 'react-native-elements';
// icons
import Feather from 'react-native-vector-icons/Feather';
import AntDesign from 'react-native-vector-icons/AntDesign';
import Ionic from 'react-native-vector-icons/Ionicons';
import FontAwesome from 'react-native-vector-icons/FontAwesome';
//navigation to comment screen
import { useNavigation } from '@react-navigation/native';

const Comment = ({ comment }) => {
  const navigation = useNavigation();
```

```
  return (
    <View style={styles.container}>
    <View style={styles.header}>
     <View style={{flexDirection: 'row', alignItems: 'center'}}>
     {/* Navigate to another user's screen on profile picture or name press */}
     <TouchableOpacity onPress={() => navigation.navigate('Profile',
{profile_picture: comment.profile_picture, name: comment.name})}>
        <Image source={{url: comment.profile_picture}}
style={styles.profile_picture}/>
     </TouchableOpacity>
     {/* Padding to create space between the profile picture and the user's name
*/}
     <View style={{paddingLeft: 10}}>
        <TouchableOpacity>
         <Text style={{fontSize: 15, fontWeight: '500',}}>{comment.name}</Text>
        </TouchableOpacity>
        <View style={{width: '87%'}}>
         <Text style={{paddingTop: 5}}>{comment.text}</Text>
        </View>
     </View>
     </View>
     <Feather name="more-vertical" style={{fontSize: 20}} />
    </View>
    </View>
  );
};

export default Comment;

const styles = StyleSheet.create({
   container: {
    paddingBottom: 0,
    borderBottomColor: 'gray',
    borderBottomWidth: 0.1,
   },
   header: {
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    padding: 15,
   },
   profile_picture:{
     width: 40,
```

```
      height: 40,
      borderRadius: 100,
      marginTop: 0,
    },
    textContainer:{
     marginHorizontal: 15,
    },
    likeAndComment:{
     flex: 1,
     flexDirection: "row",
    }
});
```

4.1.1.2 FormButton.js

```
import { View, Text, TouchableOpacity, StyleSheet } from 'react-native';
// mobile window height and width from Dimensions
import {windowHeight, windowWidth} from '../utils/Dimensions';

// form button for login and register screens
const FormButton = ({ navigation, title, ...rest }) => {
   return(
   <TouchableOpacity style={styles.container} {...rest}>
      <Text style={styles.text}>{title}</Text>
   </TouchableOpacity>
   );
};

export default FormButton;

const styles = StyleSheet.create({
   container:{
      marginTop: 15,
      justifyContent: 'center',
      alignItems: 'center',
      borderRadius: 3,
      backgroundColor: '#0C4BAE',
      width: '80%',
      height: windowHeight / 18,
   },
   text:{
      fontSize: 14,
      color: '#FFF',
      fontFamily: 'Verdana',
```

```
    }
});
```

4.1.1.3 FormInput.js

```
import {View, TextInput, StyleSheet} from 'react-native';
// mobile window height and width from Dimensions
import {windowHeight, windowWidth} from '../utils/Dimensions';
// icon
import { Feather } from '@expo/vector-icons';

const FormInput = ({labelValue, placeholderText, iconType, ...rest}) => {
  return (
    <View style={styles.inputContainer}>
      <View style={styles.iconStyle}>
        <Feather name={ iconType } size={18} color="#000" />
      </View>
      <TextInput
        value={labelValue}
        style={styles.input}
        numberOfLines={1}
        placeholder={placeholderText}
        placeholderTextColor="#666"
        {...rest}
      />
    </View>
  );
};

export default FormInput;

const styles = StyleSheet.create({
  inputContainer: {
    marginTop: 15,
    width: '80%',
    height: windowHeight / 18,
    borderColor: '#ccc',
    borderRadius: 3,
    borderWidth: 1,
    flexDirection: 'row',
    alignItems: 'center',
    backgroundColor: '#fff',
  },
  iconStyle: {
    padding: 10,
```

```
    height: '100%',
    justifyContent: 'center',
    alignItems: 'center',
    borderRightColor: '#ccc',
    borderRightWidth: 1,
    width: 50,
  },
  input: {
    padding: 10,
    flex: 1,
    fontSize: 14,
    fontFamily: 'Verdana',
    color: '#333',
    justifyContent: 'center',
    alignItems: 'center',
  },
});
```

4.1.1.4 SocialButton.js

```
import {Text, TouchableOpacity, View, StyleSheet} from 'react-native';
import {windowHeight, windowWidth} from '../utils/Dimensions';
import FontAwesome from 'react-native-vector-icons/FontAwesome';

// component used for facebook and google login buttons
const SocialButton = ({ title, type, color, backgroundColor, ...rest}) => {
  let buttonColor = backgroundColor;
  return (
    <TouchableOpacity style={[styles.buttonContainer, {backgroundColor:
buttonColor}]} {...rest}>
      <View style={styles.iconWrapper}>
        <FontAwesome name={type} style={styles.icon} size={22} color={color} />
      </View>
      <View style={styles.btnTxtWrapper}>
        <Text style={[styles.buttonText, {color: color}]}>{title}</Text>
      </View>
    </TouchableOpacity>
  );
};

export default SocialButton;

const styles = StyleSheet.create({
  buttonContainer: {
```

```
    width: '80%',
    height: windowHeight / 18,
    padding: 10,
    flexDirection: 'row',
    borderRadius: 3,
    marginTop: 15,
  },
  iconWrapper: {
    width: 30,
    justifyContent: 'center',
    alignItems: 'center',
  },
  icon: {
    fontWeight: 'bold',
  },
  btnTxtWrapper: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  buttonText: {
    fontSize: 14,
    fontFamily: 'Verdana',
  },
});
```

4.1.1.5 Stories.js

```
import { View, Text, StyleSheet, TouchableOpacity, Image, ScrollView } from 'react-
native';
import Entypo from 'react-native-vector-icons/Entypo';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const Stories = () => {
  // state variables to store the user's profile picture
  const [profilePicture, setProfilePicture] = useState();

  // get the current users profile picture
  const getUser = async() => {
    db.collection('users')
    .doc(firebase.auth().currentUser.uid)
    .get()
```

```
   .then((documentSnapshot) => {
    if( documentSnapshot.exists ) {
     setProfilePicture(documentSnapshot.data().profile_picture);
    }
   })
  }

  // get the current user's profile picture when the screen is loaded
  useEffect(() => {
   getUser();
  }, []);

  return (
   <View style={styles.container}>
    <ScrollView horizontal showsHorizontalScrollIndicator={false}>
    <View style={{paddingHorizontal: 5,}}>
     <View style={styles.border}>
      <View style={styles.innerBorder}>
       <Image source={{uri: profilePicture}} style={styles.image}/>
        <View style={{ position: 'absolute', bottom: 0, right: 0, borderRadius: 50,
backgroundColor: '#FFF', alignItems: 'center', justifyContent: 'center'}}>
          <Entypo name="circle-with-plus" style={{ fontSize: 20, color: '#0C4BAE'}}
/>
        </View>
      </View>
     </View>
     <Text style={styles.username}>Your Story</Text>
    </View>
    {/*{USERS.map((story, index) => (
    <View key={index} style={{paddingHorizontal: 5,}}>
     <View style={styles.border}>
      <View style={styles.innerBorder}>
       <Image source={story.image} style={styles.image}/>
      </View>
     </View>
     <Text style={styles.username}>{
          story.user.length > 11 ? story.user.slice(0,10) + '...' : story.user
     }</Text>
    </View>))} */}
   </ScrollView>
  </View>
  );
}
```

```
export default Stories;

const styles = StyleSheet.create({
    container:{
        marginVertical: 5,
    },
    border:{
        backgroundColor: '#0C4BAE',
        height: 68,
        width: 68,
        borderRadius: 50,
        alignItems: 'center',
        justifyContent: 'center',
    },
    innerBorder: {
        backgroundColor: '#fff',
        height: 64,
        width: 64,
        borderRadius: 50,
        justifyContent: 'center',
        alignItems: 'center',
    },
    image:{
        width: 60,
        height: 60,
        borderRadius: 50,
    },
    users:{
        alignItems: 'center',
        marginRight: 10,
    },
    username:{
        marginTop: 5,
        fontSize: 12,
    }
});
```

4.1.2 Config Folder

4.1.2.1 Firebase.js

```
import 'firebase/compat/auth';
import 'firebase/compat/firestore';
import { getStorage } from "firebase/storage";
```

```
const firebaseConfig = {
  apiKey: "AIzaSyC1bnf2ZjNR4YYxHrXH7ftCMqquS656Fmg",
  authDomain: "healthspace-12f87.firebaseapp.com",
  projectId: "healthspace-12f87",
  storageBucket: "healthspace-12f87.appspot.com",
  //storageBucket: "gs://healthspace-12f87.appspot.com",
  messagingSenderId: "803991289685",
  appId: "1:803991289685:web:815a7960243e10e35c39dc",
  measurementId: "G-B0MYV6KJZ9"
};

let app;
if (firebase.apps.length === 0) {
  app = firebase.initializeApp(firebaseConfig)
} else {
  app = firebase.app();
}

const db = app.firestore();
const auth = firebase.auth();
// Get a reference to the storage service
const storage = getStorage(app);

export { db, auth, firebaseConfig};
```

4.1.3 Utils Folder

4.1.3.1 Dimensions.js

```
import { Dimensions } from 'react-native';
export const windowWidth = Dimensions.get('window').width;
export const windowHeight = Dimensions.get('window').height;
```

4.1.4 Screens Folder

4.1.4.1 OnboardingScreen.js

```
import React from 'react';
import { Button, Image, Text, View } from 'react-native';
import Onboarding from 'react-native-onboarding-swiper';

const OnboardingScreen = ({ navigation }) => {
    return(
    <Onboarding
    onSkip={() => navigation.replace('LoginScreen')}
    onDone={() => navigation.navigate('LoginScreen')}
```

```
    pages={[ {
      backgroundColor: '#0C4BAE',
      image: <Image style={{ width: 100, height: 100 }}
source={require('../assets/post.png')} />,
      title: 'Posts',
      subtitle: 'Share health-related posts, photos and comments with people of
similar heath interests',
    },
    {
      backgroundColor: '#0C4BAE',
      image: <Image style={{ width: 100, height: 100 }}
source={require('../assets/like.png')} />,
      title: 'Reactions',
      subtitle: "Interact with other user's posts and photos with likes and comments",
    },
    {
      backgroundColor: '#0C4BAE',
      image: <Image style={{ width: 100, height: 100 }}
source={require('../assets/link.png')} />,
      title: 'Connect',
      subtitle: 'Connect with those who share similar health interests and
experiences',
    },
    {
      backgroundColor: '#0C4BAE',
      image: <Image style={{ width: 100, height: 100 }}
source={require('../assets/chat.png')} />,
      title: 'Chat',
      subtitle: 'Communicate with people of similar health interests via private
messaging',
    },
    ]} />
    );
};

export default OnboardingScreen;
```

4.1.4.2 LoginScreen.js

```
import React, {useState} from 'react';
import { View, Text, Button, Image, StyleSheet, ScrollView, TouchableOpacity,
TouchableWithoutFeedback, Keyboard} from 'react-native';
// KeyboardAwareScrollView: does not hide the inputs when the keyboard is open
```

```
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-
view'
//import components
import FormInput from '../components/FormInput';
import FormButton from '../components/FormButton';
import SocialButton from '../components/SocialButton';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const LoginScreen = ({ navigation }) => {
  // state variables for the email and password
  const [email, setEmail] = useState();
  const [password, setPassword] = useState();

  // function to handle login
  const onLogin = () => {
    // set email to all lowercase
    const newEmail = email.toLowerCase();
    setEmail(newEmail);
    firebase.auth().signInWithEmailAndPassword(email, password)
    .then((result) => {
      console.log("Firebase login succesfull:", email, password)
    })
    .catch((error) => {
      console.log("Firebase login failed:", email, password)
    })
  }

  return(
  <TouchableWithoutFeedback onPress={Keyboard.dismiss} accessible={false}>
  <KeyboardAwareScrollView
    style={{backgroundColor: '#fff'}}
    contentContainerStyle={styles.container}
    resetScrollToCoords={{x: 0, y: 0}}
    scrollEnabled={false}
    >
    <Image source={require('../assets/illustration-one.png')} resizeMode='contain'
style={styles.illustration} />
    <Text style={styles.heading}>Welcome Back!</Text>
    <Text style={styles.subHeading}>Sign in to continue</Text>
    <FormInput
      labelValue={email}
```

```
          onChangeText={(email) => setEmail(email)}
          iconType='mail'
          placeholder='Email Address'
          autoCapitalise='none'/>
        <FormInput
          labelValue={password}
          onChangeText={(password) => setPassword(password)}
          iconType='lock'
          placeholder='Password'
          secureTextEntry={true}/>
        <TouchableOpacity style={styles.forgotButton} onPress={() => {}}>
        <View style={styles.forgotButtonContainer}>
          <Text style={styles.forgotButtonText}>Forgot Password?</Text>
        </View>
        </TouchableOpacity>
        <FormButton
          title="Sign In"
          onPress={onLogin} />
        <Text style={styles.connectUsingText} > Or connect using </Text>
        <SocialButton
          title='Sign in with Facebook'
          type='facebook'
          color='#4867AA'
          backgroundColor='#E6EAF4'/>
        <SocialButton
          title='Sign in with Google'
          type='google'
          color='#DE4D41'
          backgroundColor='#F5E7EA'/>
        <TouchableOpacity onPress={() => navigation.navigate('RegisterScreen')}>
        <Text style={styles.noAccountText}> Don't have an account? <Text
style={styles.signUpText}> Sign up! </Text></Text>
        </TouchableOpacity>
    </KeyboardAwareScrollView>
    </TouchableWithoutFeedback>
    );
};

export default LoginScreen;


const styles = StyleSheet.create({
    container:{
```

```
    flex: 1,
    backgroundColor: '#FFF',
    alignItems: 'center',
    justifyContent: 'center',
  },
  illustration:{
    width: 400,
    height: 200,
  },
  heading:{
    fontFamily: 'Arial',
    fontSize: 25,
  },
  subHeading:{
    fontFamily: 'Arial',
    fontSize: 15,
    marginTop: 15,
  },
  forgotButton:{
    alignSelf: 'flex-end',
    marginRight: '10%',
    marginTop: 15,
  },
  forgotButtonText:{
    fontFamily: 'Arial',
    fontSize: 14,
  },
  connectUsingText: {
    fontFamily: 'Arial',
    fontSize: 14,
    marginTop: 15,
  },
  noAccountText: {
    fontFamily: 'Arial',
    fontSize: 15,
    marginTop: 15,
  },
  signUpText: {
    color: '#0C4BAE',
    fontFamily: 'Arial',
    fontWeight: 'bold',
    fontSize: 15,
    marginTop: 15,
```

```
    }
});
```

4.1.4.3 RegisterScreen.js

```
import React, {useState} from 'react';
import { View, Text, Button, Image, StyleSheet, ScrollView, TouchableOpacity,
TouchableWithoutFeedback, Keyboard} from 'react-native';
// KeyboardAwareScrollView: does not hide the input when the keyboard is shown
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-
view'
// icons
import {Ionicons} from '@expo/vector-icons';
// image picker
import * as ImagePicker from 'expo-image-picker';
// import custom components
import FormInput from '../components/FormInput';
import FormButton from '../components/FormButton';
import SocialButton from '../components/SocialButton';
// firebase
import firebase from 'firebase/compat/app';
import { auth, db, firebaseConfig } from '../config/Firebase';
// image storage
import { getStorage, ref, uploadBytesResumable, getDownloadURL, uploadBytes }
from "firebase/storage";

const RegisterScreen = ({ navigation }) => {
    // state variables
    const [name, setName] = useState();
    const [email, setEmail] = useState();
    const [password, setPassword] = useState();
    const [confirmPass, setConfirmPass] = useState();
    const [image, setImage] = useState(null);

    // function to launch the image library, allowing the user to select an image
    const pickImage = async () => {
        // no permissions request are necessary
        let result = await ImagePicker.launchImageLibraryAsync({
            mediaTypes: ImagePicker.MediaTypeOptions.All,
            allowsEditing: true,
            aspect: [4, 3],
            quality: 1
        });
```

```
    if (!result.cancelled) {
      setImage(result.uri);
    }
};

// function to handle the sign up process
const handleSignup = async (image) => {
    // get the current time
    const time = new Date().getTime();

    // create the file metadata
    /** @type {any} */
    const metadata = {
      contentType: 'image/jpeg'
    };
    // upload the image to firebase storage
    const result = await fetch(image);
    // get the blob of the image
    const blob = await result.blob();
    // create a reference to the location where we want to store the image
    const storage = getStorage();
    const storageRef = ref(storage, 'images/' + time + '.jpg');
    // upload the image to the storage reference
    const uploadTask = await uploadBytesResumable(storageRef, blob, metadata);
    // get the download url of the image
    const downloadURL = await getDownloadURL(storageRef);

    // lowercase the email
    const newEmail = email.toLowerCase();
    setEmail(newEmail);

    // store the user's information in firebase
    if(password === confirmPass) {
      auth.createUserWithEmailAndPassword(email, password)
      .then(() => {
        // get the user's id
        const user_id = auth.currentUser.uid;
        // create a reference to the user's profile
        const userRef = db.collection('users').doc(user_id);
        // create a new document in the user's profile
        userRef.set({
          name: name,
          email: email,
```

```
                profile_picture: downloadURL,
                id: user_id,
                created_at: time,
                bio: '',
                condition: '',
                token: '',
              });
            })
            .catch(error => {
              console.log(error);
            });
        } else {
          alert('Passwords do not match');
        }
    };

    return(
    <TouchableWithoutFeedback onPress={Keyboard.dismiss} accessible={false}>
    <KeyboardAwareScrollView
        style={{backgroundColor: '#fff'}}
        contentContainerStyle={styles.container}
        resetScrollToCoords={{x: 0, y: 0}}
        scrollEnabled={false}
        >
        <Image source={require('../assets/illustration-two.png')} resizeMode='contain'
style={styles.illustration} />
        <Text style={styles.heading}>Let's Get Started!</Text>
        <Text style={styles.subHeading}>Create an account to continue</Text>
        <TouchableOpacity style={styles.avatarPlaceholder} onPress={pickImage}>
          <Image source={{uri: image}} style={styles.avatar} />
          <Ionicons name="ios-add" size={32} color="#FFF"
style={styles.image}></Ionicons>
        </TouchableOpacity>
        <FormInput
          labelValue={name}
          onChangeText={(name) => setName(name)}
          iconType='user'
          placeholder='Full Name'
          keyboardType='email-address' />
        <FormInput
          labelValue={email}
          onChangeText={(email) => setEmail(email)}
          iconType='mail'
```

```
          placeholder='Email Address'
          autoCapitalize='none'
          keyboardType='email-address' />
        <FormInput
          labelValue={password}
          onChangeText={(password) => setPassword(password)}
          iconType='lock'
          placeholder='Password'
          secureTextEntry={true} />
        <FormInput
          labelValue={confirmPass}
          onChangeText={(confirmPass) => setConfirmPass(confirmPass)}
          iconType='lock'
          placeholder='Confirm Password'
          secureTextEntry={true}  />
        <FormButton
          title="Sign Up"
          onPress={() => handleSignup(image)} />
        <TouchableOpacity onPress={() => navigation.navigate('LoginScreen')}>
        <Text style={styles.haveAccountText}> Already have an account? <Text
style={styles.signInText}>Sign in! </Text></Text>
     </TouchableOpacity>
    </KeyboardAwareScrollView>
    </TouchableWithoutFeedback>
    );
};

export default RegisterScreen;

const styles = StyleSheet.create({
    container:{
        flex: 1,
        backgroundColor: '#FFF',
        alignItems: 'center',
        justifyContent: 'center',
    },
    illustration:{
        width: 400,
        height: 200,
    },
    heading:{
        fontFamily: 'Arial',
        fontSize: 25,
```

```
    marginTop: 15,
  },
  subHeading:{
    fontFamily: 'Arial',
    fontSize: 15,
    marginTop: 15,
  },
  avatarPlaceholder: {
    width: 100,
    height: 100,
    backgroundColor: '#e1e2e6',
    borderRadius: 50,
    marginTop: 15,
    justifyContent: 'center',
    alignItems: 'center'
  },
  avatar: {
    position: 'absolute',
    width: 100,
    height: 100,
    borderRadius: 50
  },
  haveAccountText: {
    fontFamily: 'Arial',
    fontSize: 15,
    marginTop: 15,
  },
  signInText: {
    color: '#0C4BAE',
    fontFamily: 'Arial',
    fontWeight: 'bold',
    fontSize: 15,
    marginTop: 15,
  }
});
```

4.1.4.4 HomeScreen.js

```
import React, {useEffect, useState} from 'react';
import { StatusBar } from 'expo-status-bar';
import { RefreshControl, StyleSheet, Text, View, SafeAreaView, ScrollView,
TouchableOpacity, Image, Platform } from 'react-native';
//i cons
import { Fontisto } from '@expo/vector-icons';
```

```
import { Divider } from 'react-native-elements';
import Stories from '../components/Stories';
import Post from '../components/Post';
// notifications
import Constants from 'expo-constants';
import * as Notifications from 'expo-notifications';
import * as Device from 'expo-device';
// firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const HomeScreen = ({ navigation }) => {
  const [posts, setPosts] = useState([]);
  const [refreshing, setRefreshing] = useState(false);

  // function to get all posts
  const getPosts = async() => {
    db.collection('posts').orderBy('created_at', 'desc').onSnapshot(snapshot => {
      setPosts(snapshot.docs.map(doc => doc.data()));
    });
  }

  // on page load
  useEffect(() => {
    // fetch all posts
    getPosts();
    // check if the user has notifications enabled
    registerForPushNotificationsAsync();
  }, []);

  // function to handle sign out
  const handleSignout = () => {
    try {
      firebase.auth().signOut()
      console.log('Signed out successfully')
    } catch (error) {
      console.log(error)
    }
  }

  // as the home screen is the first screen after login, we can register the user for
notifications
  async function registerForPushNotificationsAsync() {
```

```
  // token is undefined initially
  let token;
  // check to see if the user has granted permission to notifications
  if (Device.isDevice) {
    const { status: existingStatus } = await Notifications.getPermissionsAsync();
    let finalStatus = existingStatus;
    if (existingStatus !== 'granted') {
      const { status } = await Notifications.requestPermissionsAsync();
      finalStatus = status;
    }
    if (finalStatus !== 'granted') {
      console.log('Failed to get push token for push notification!');
      return;
    }
    // permissions granted, get the token
    token = (await Notifications.getExpoPushTokenAsync()).data;
    console.log(token);
  } else {
    console.log('Must use physical device for Push Notifications');
  }

  // store the token in firebase
  if(token){
    const result = await
db.collection('users').doc(firebase.auth().currentUser.uid).update({
      token: token
    });
  }

  if (Platform.OS === 'android') {
    Notifications.setNotificationChannelAsync('default', {
      name: 'default',
      importance: Notifications.AndroidImportance.MAX,
      vibrationPattern: [0, 250, 250, 250],
      lightColor: '#FF231F7C',
    });
  }

  return token;
}

// timeout function used when refreshing the page
const wait = (timeout) => {
```

```
  return new Promise(resolve => {
    setTimeout(resolve, timeout);
  });
}

// refresh the page function
const onRefresh = () => {
  setRefreshing(true);
  getPosts();
  wait(1500).then(() => setRefreshing(false));
}

return (
  <SafeAreaView style={styles.container}>
  <View style={styles.headerContainer}>
    <TouchableOpacity onPress={handleSignout}>
      <Image
      style={styles.logo}
      source={require('../assets/logo.png')}
      />
    </TouchableOpacity>
    <View style={styles.iconContainer}>
      <TouchableOpacity onPress={() => navigation.navigate('Search User')}>
          <Fontisto
          name="search"
          size={22}
          style={styles.icon}/>
      </TouchableOpacity>
    </View>
  </View>
  <ScrollView refreshControl={
    <RefreshControl
      refreshing={refreshing}
      onRefresh={onRefresh}
    />
  }>
  <Stories />
  <Divider width={8} style={{paddingTop: 5}}/>
  {posts.map((post, index) => (
    <Post post={post} key={index} />
  ))}
  </ScrollView>
  </SafeAreaView>
```

```
  );
};

export default HomeScreen;

const styles = StyleSheet.create({
   container:{
     flex: 1,
     backgroundColor: '#FFF',
   },
   headerContainer:{
     flexDirection: 'row',
     alignItems: 'center',
     justifyContent: 'space-between'
   },
   iconContainer:{
      marginRight: 10,
   },
   icon:{
      width: 30,
      height: 30,
   },
   logo:{
      width: 150,
      height: 50,
      resizeMode: 'contain',
   }
});
```

4.1.4.5 SearchUserScreen.js

```
import React, {useEffect, useState} from 'react';
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View, Image, TouchableOpacity, ScrollView, TextInput }
from 'react-native';
// icons
import { FontAwesome5 } from '@expo/vector-icons';
import { FontAwesome } from '@expo/vector-icons';
import { Ionicons } from '@expo/vector-icons';
import { conditions } from '../data/conditions';
//import SearchableDropdown component
import SearchableDropdown from 'react-native-searchable-dropdown';
//firebase
import firebase from 'firebase/compat/app';
```

```
import { auth, db } from '../config/Firebase';

const SearchUserScreen = ({ navigation }) => { {

   // state variables
   const [users, setUsers] = useState([]);
   const [filteredUsers, setFilteredUsers] = useState([]);
   const [search, setSearch] = useState('');
   const [currentUserName, setCurrentUserName] = useState('');
   const [currentUserPic, setCurrentUserPic] = useState('');
   const [placeholder, setPlaceholder] = useState('Search for a condition..');
   const [id, setId] = useState('');
   const [conditionName, setConditionName] = useState();
   const [description, setDescription] = useState();
   const [symptoms, setSymptoms] = useState();
   const [treatment, setTreatment] = useState();

   // get all users except for the current user
   const getUsers = async () => {
      const users = [];
      await db.collection('users').where('email', '!=',
auth.currentUser.email).get().then(querySnapshot => {
         querySnapshot.forEach(doc => {
            users.push(doc.data());
         });
      });
      setUsers(users);
      setFilteredUsers(users);
   }

   // get current user from the firebase database
   const getCurrentUser = async () => {
      await db.collection('users').where('email', '==',
auth.currentUser.email).get().then(querySnapshot => {
         querySnapshot.forEach(doc => {
            setCurrentUserName(doc.data().name);
            setCurrentUserPic(doc.data().profile_picture);
         });
      });
   }

   // get the users and the current user when the screen is loaded
   useEffect(() => {
```

```
    getUsers();
    getCurrentUser();
  }, []);

  // function to filter the users based on the search input
  const searchFilter = (text) => {
   if(text){
     const newData = users.filter(user => {
       const userData = user.name ? user.name.toUpperCase() : ''.toUpperCase();
       const textData = text.toUpperCase();
       return userData.indexOf(textData) > -1;
     })
     setFilteredUsers(newData);
     setSearch(text);
   } else {
     setFilteredUsers(users);
     setSearch('');
   }
 }

  // create a deep copy of the conditions array
  // because I need to remove the description, symptoms, and treatment from the
array
  // so that it can be used in a dropdown menu
  const items = JSON.parse(JSON.stringify(conditions));

  // remove attributes from conditions
  // only need the name and id for dropdown menu
  items.forEach(items => {
    delete items.description;
    delete items.symptoms;
    delete items.treatment;
  });

  // function to handle condtion selection
  const handleConditionChange = (item) => {
    setId(item.id);
    setName(item.name);
    // loop through the conditions array to find the condition that was selected
    // and set the description, symptoms, and treatment to the state variables
    for(let i = 0; i < conditions.length; i++) {
      if(conditions[i].id === item.id) {
        setDescription(conditions[i].description);
```

```
            setSymptoms(conditions[i].symptoms);
            setTreatment(conditions[i].treatment);
          }
        }
      }

  return (
    <View style={styles.container}>
      <TextInput
        style={styles.search}
        placeholder="Search for a user.."
        onChangeText={(text) => searchFilter(text)}
        value={search} />
      <ScrollView>
        {filteredUsers.map((user, index) => (
          <TouchableOpacity key={index} style={styles.userContainer}
          onPress={() => navigation.navigate('Profile', { user_id: user.id, profile_picture:
user.profile_picture, name: user.name })}>
            <Image source={{uri: user.profile_picture}} style={styles.image} />
            <View style={styles.nameContainer}>
              <Text style={styles.name}>{user.name}</Text>
            </View>
          </TouchableOpacity>
        ))}
      </ScrollView>
    </View>
  );
}};

export default SearchUserScreen;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#FFF',
  },
  search: {
    height: 40,
    borderWidth: 1,
    borderColor: '#E5E5E5',
    marginTop: 10,
    marginLeft: 10,
    marginRight: 10,
```

```
    padding: 10,
    borderRadius: 5,
  },
  userContainer: {
    flexDirection: 'row',
    alignItems: 'center',
    padding: 10,
  },
  image: {
    marginLeft: 10,
    width: 40,
    height: 40,
    borderRadius: 20,
  },
  nameContainer: {
    marginLeft: 15,
  },
  name: {
    fontSize: 14,
    fontWeight: '500',
  },
  header: {
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    padding: 15,
  },
  profile_picture:{
     width: 40,
     height: 40,
     borderRadius: 100,
     marginTop: 0,
  },
});
```

4.1.4.6 UserScreen.js

```
import React, {useEffect, useState} from 'react';
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View, Image, TouchableOpacity, ScrollView } from 'react-
native';
//components
import { Divider } from 'react-native-elements';
import Stories from '../components/Stories';
```

```
import Post from '../components/Post';
import {windowHeight, windowWidth} from '../utils/Dimensions';
// icons
import { FontAwesome5 } from '@expo/vector-icons';
import { FontAwesome } from '@expo/vector-icons';
import { Ionicons } from '@expo/vector-icons';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const UserScreen = ({ route, navigation }) => { {
   // route params
   const {profile_picture, name, user_id} = route.params;
   // state variables
   const [user, setUser] = useState(null);
   const [currentUser, setCurrentUser] = useState(null);
   const [posts, setPosts] = useState([]);
   const [following, setFollowing] = useState(false);
   const [followingText, setFollowingText] = useState('Follow');
   const [followingCount, setFollowingCount] = useState(0);
   const [followerCount, setFollowerCount] = useState(0);
   const [bio, setBio] = useState();
   const [condition, setCondition] = useState();

   // get user information that matches the ID passed in from the route
   const getUser = async () => {
    const user = await db.collection('users').doc(user_id).get();
    setUser(user.data());
    setBio(user.data().bio);
    setCondition(user.data().condition);
   }

   // get the current user's information
   const getCurrentUser = async () => {
    const currentUser = await
db.collection('users').doc(firebase.auth().currentUser.uid).get();
    setCurrentUser(currentUser.data());
   }

   // send a push notification when the user receives a new follwer
   const sendPushNotification = async (token) => {
    const message = {
     to: token,
```

```
      sound: 'default',
      title: 'HealthSpace',
      body: currentUser.name + ' started following you!',
      data: { someData: 'goes here' },
    };

    await fetch('https://exp.host/--/api/v2/push/send', {
      method: 'POST',
      headers: {
        Accept: 'application/json',
        'Accept-encoding': 'gzip, deflate',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(message),
    });
  }

  // get posts that match the user's ID passed in from the route
  // order the posts by the date they were created
  const getPosts = async () => {
    const posts = await db.collection('posts').where('user_id', '==',
user_id).orderBy('created_at', 'desc').get();
    setPosts(posts.docs.map(doc => doc.data()));
    console.log(posts.docs.map(doc => doc.data()));
  }

  // check if the current user is following the user profile you clicked on
  const checkFollowing = async() => {
    db.collection('following')
    .doc(firebase.auth().currentUser.uid)
    .collection('userFollowing')
    .doc(user_id)
    .get()
    .then((documentSnapshot) => {
      if(documentSnapshot.exists ) {
        setFollowing(true);
        setFollowingText('Following');
      } else{
        setFollowing(false);
        setFollowingText('Follow');
      }
    })
  }
```

```
// handle the follow button being pressed
const handleFollowBtn = async() => {
  // check if following
  if(following) {
    // unfollow the user from the following collection
    db.collection('following')
    .doc(firebase.auth().currentUser.uid)
    .collection('userFollowing')
    .doc(user_id)
    .delete()
    .then(() => {
      setFollowing(false);
      setFollowingText('Follow');
    })

    // remove the current user from the user's follower collection
    db.collection('followers')
    .doc(user_id)
    .collection('userFollowers')
    .doc(firebase.auth().currentUser.uid)
    .delete()

    // remove the last notification from the user's notification collection
    db.collection('notifications')
    .doc(user_id)
    .collection('userNotifications')
    .doc(firebase.auth().currentUser.uid)
    .delete()

  } else {
    // follow the user
    db.collection('following')
    .doc(firebase.auth().currentUser.uid)
    .collection('userFollowing')
    .doc(user_id)
    .set({})
    .then(() => {
      setFollowing(true);
      setFollowingText('Following');
    })

    // add the current user to the user's follower collection
```

```
      db.collection('followers')
      .doc(user_id)
      .collection('userFollowers')
      .doc(firebase.auth().currentUser.uid)
      .set({})

      // send a push notification if the user has a token
      if(user.token) {
       sendPushNotification(user.token);
      }

      // notify the user that they have a new follower
      // including the document id so that the notification can be deleted later if
needed
      const newDocRef =
db.collection('notifications').doc(user_id).collection('allNotifications').doc();
      newDocRef.set({
       // need the current user's profile picture, name and text for the notification
       // and the time of the notification
       profile_picture: currentUser.profile_picture,
       name: currentUser.name,
       text: 'Started following you',
       time: firebase.firestore.FieldValue.serverTimestamp(),
       id: newDocRef.id
      })
    }
    // finally, update the follower count
    getFollowerCount();
  }

  // check how many people the user is following
  const getFollowingCount = async() => {
    db.collection('following')
    .doc(user_id)
    .collection('userFollowing')
    .get()
    .then((querySnapshot) => {
     setFollowingCount(querySnapshot.size);
    })
  }

  // check how many followers the user has
  const getFollowerCount = async() => {
```

```
  db.collection('followers')
  .doc(user_id)
  .collection('userFollowers')
  .get()
  .then((querySnapshot) => {
   setFollowerCount(querySnapshot.size);
  })
 }

 // check if the user has a bio and / or condition set
 // if they do, render a component with the bio and condition
 const aboutMeComponent = () => {
  if(bio && condition) {
   return (
    <View style={styles.aboutMeContainer}>
     <Text style={{fontSize: 16, fontWeight: '600', marginBottom: 10,}}>About
Me</Text>
     <View style={{flexDirection: 'row', alignItems: 'center'}}>
      <FontAwesome name="book" size={20} color="black" />
      <Text style={styles.aboutMeText}>{bio}</Text>
     </View>
     <View style={{flexDirection: 'row', alignItems: 'center', marginTop: 10}}>
      <FontAwesome5 name="heartbeat" size={20} color="black" />
      <Text style={styles.aboutMeText}>{condition}</Text>
     </View>
    </View>
   )
  }

  // if the user has a bio but not a condition, render a component with the bio
  else if(bio) {
   return (
    <View>
     <Text style={{fontSize: 16, fontWeight: '600', marginBottom: 10,}}>About
Me</Text>
     <View style={{flexDirection: 'row', alignItems: 'center'}}>
      <FontAwesome name="book" size={20} color="black" />
      <Text style={styles.aboutMeText}>{bio}</Text>
     </View>
    </View>
   )
  }
```

```
  // if the user has a condition but not a bio, render a component with the
condition
  else if(condition) {
   return (
    <View>
     <Text style={{fontSize: 16, fontWeight: '600', marginBottom: 10,}}>About
Me</Text>
     <View style={{flexDirection: 'row', alignItems: 'center'}}>
      <FontAwesome5 name="heartbeat" size={20} color="black" />
      <Text style={styles.aboutMeText}>{condition}</Text>
     </View>
    </View>
    )
  }
 }

 // on page load call all the above functions
 // 1. get the current user's information
 // 2. get the selected user's information
 // 3. check if the current user is following the selected user
 // 4. check the selected user's following count
 // 5. check the selected user's follower count
 // 6. get the current user's information
 useEffect(async() => {
  await getUser();
  await getPosts();
  await checkFollowing();
  await getFollowingCount();
  await getFollowerCount();
  await getCurrentUser();
 }, [])

 return (
   <ScrollView style={styles.container}>
   <View style={styles.content_container}>
   <View style={{alignItems: 'center'}}>
     <Image source={{uri: profile_picture}}  style={styles.userImg} />
   </View>
   <View style={{alignItems: 'center'}}>
     <Text style={styles.username}>{name}</Text>
   </View>
   </View>
   <View style={styles.counters_container}>
```

```
        <View style={{justifyContent: 'center'}}>
          <Text style={styles.title}>{posts.length}</Text>
          <Text style={styles.subtitle}>Posts</Text>
        </View>
        <View style={{justifyContent: 'center'}}>
          <Text style={styles.title}>{followerCount}</Text>
          <Text style={styles.subtitle}>Followers</Text>
        </View>
        <View style={{justifyContent: 'center'}}>
          <Text style={styles.title}>{followingCount}</Text>
          <Text style={styles.subtitle}>Following</Text>
        </View>
        </View>
        <View style={{justifyContent: 'center',}}>
        <TouchableOpacity style={styles.followBtn} onPress={handleFollowBtn}>
          <Text style={styles.buttonText}>{followingText}</Text>
        </TouchableOpacity>
        </View>
        <View style={{marginLeft: 20, marginTop: 0}}>
        <View style={styles.aboutMe}>
          {aboutMeComponent()}
        </View>
        </View>
        {posts.map((post, index) => (
        <Post post={post} key={index} />
        ))}
    </ScrollView>
    );
 }};

export default UserScreen;

const styles = StyleSheet.create({
    container:{
      flex: 1,
      backgroundColor: '#FFF',
    },
    content_container:{
      alignItems: 'center',
    },
    userImg: {
      height: 125,
      width: 125,
```

```
  borderRadius: 75,
  marginTop: 15,
 },
 counters_container: {
  flexDirection: 'row',
  justifyContent: 'space-around',
  width: '100%',
  marginTop: 15,
 },
 username: {
  marginTop: 15,
  fontSize: 20,
 },
 followBtn:{
  alignItems: 'center',
  justifyContent: 'center',
  margin: 15,
  borderRadius: 3,
  backgroundColor: '#FFF',
  height: windowHeight / 20,
  borderColor: 'grey',
  borderWidth: 1,
 },
 buttonText:{
  fontSize: 14,
  color: '#000',
  fontFamily: 'Verdana',
 },
 title: {
  fontSize: 20,
  fontWeight: 'bold',
  marginBottom: 5,
  textAlign: 'center',
 },
 subtitle: {
  fontSize: 12,
  color: '#000',
  textAlign: 'center',
 },
 aboutMeContainer: {
  flexDirection: 'column',
 },
 aboutMeText:{
```

```
      fontSize: 14,
      marginLeft: 10,
    },
  });
```

4.1.4.7 CommentScreen.js

```
import React, {useState, useEffect} from 'react';
import {View, Text, Image, TouchableOpacity, TextInput, StyleSheet,
TouchableWithoutFeedback, Keyboard, ScrollView} from 'react-native';
import { Divider } from 'react-native-elements';
import Feather from 'react-native-vector-icons/Feather';
import AntDesign from 'react-native-vector-icons/AntDesign';
import Ionic from 'react-native-vector-icons/Ionicons';
import FontAwesome from 'react-native-vector-icons/FontAwesome';
import Comment from '../components/Comment';
// to ensure the keyboard doesn't go over the input box
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-
view'
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const CommentScreen = ({ route, navigation }) => { {
  const {profile_picture, name, text, like, likes, image, id, user_id, token} =
route.params;
  // state variables for the current user and their profile picture
  const [user, setUser] = useState(null)
  const [userPic, setUserPic] = useState();
  // state variables to deal with a single comment + load all comments for current
post
  const [comment, setComment] = useState('');
  const [comments, setComments] = useState([]);

  // get the current user's information
  const getUser = async () => {
    db.collection('users')
    .doc(firebase.auth().currentUser.uid)
    .get()
    .then((documentSnapshot) => {
      if( documentSnapshot.exists ) {
        setUser(documentSnapshot.data());
        setUserPic(documentSnapshot.data().profile_picture);
        console.log('User Data', documentSnapshot.data());
      } else{
```

```
      console.log('User not found');
    }
  })
}

// get all comments for the current post
const getComments = async () => {
  db.collection('posts')
  .doc(id)
  .collection('comments')
  .orderBy('timestamp', 'desc')
  .get()
  .then((querySnapshot) => {
    const comments = [];
    querySnapshot.forEach((doc) => {
      comments.push(doc.data());
    });
    setComments(comments);
  })
}

// load the current user, post ID and comments upon opening the screen
useEffect(async () => {
  await getUser();
  await getComments();
  console.log('Token', token);
  console.log('User ID', user_id);
}, []);

// send a push notification when someone comments on a post
const sendPushNotification = async (token) => {
  const message = {
    to: token,
    sound: 'default',
    title: 'HealthSpace',
    body: user.name + ' commented on your post.',
    data: { someData: 'goes here' },
  };

  await fetch('https://exp.host/--/api/v2/push/send', {
    method: 'POST',
    headers: {
      Accept: 'application/json',
```

```
      'Accept-encoding': 'gzip, deflate',
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(message),
  });
}


// submit comment to firebase
const submitComment = async() => {
  db.collection('posts')
  .doc(id)
  .collection('comments')
  .add({
    name: user.name,
    profile_picture: userPic,
    text: comment,
    likes: 0,
    timestamp: firebase.firestore.FieldValue.serverTimestamp(),
  })
  .then(() => {
    setComment('');
    Keyboard.dismiss();
    // reload comments
    getComments();
    // send a push notification if if the author of the post is not the current user
and has a notification token
    if(token) {
      sendPushNotification(token);
      console.log('Push notification sent');
    }
    // send notification to the owner of the post
    // including the document id so that the notification can be deleted later if
needed
    const newDocRef =
db.collection('notifications').doc(user_id).collection('allNotifications').doc();
    newDocRef.set({
      profile_picture: userPic,
      name: user.name,
      text: 'commented on your post',
      time: firebase.firestore.FieldValue.serverTimestamp(),
      id: newDocRef.id
    })
```

```
  })
  .catch((error) => {
    console.log('Error adding comment', error);
  });
 }

 return (
 <ScrollView style={styles.container}>
 <TouchableWithoutFeedback onPress={Keyboard.dismiss} accessible={false}>
 <KeyboardAwareScrollView
    style={{backgroundColor: '#fff'}}
    contentContainerStyle={styles.container}
    resetScrollToCoords={{x: 0, y: 0}}
    scrollEnabled={false}
    >
  <View style={styles.header}>
    <View style={{flexDirection: 'row', alignItems: 'center'}}>
    <Image source={{uri: profile_picture}} style={styles.profile_picture}/>
    {/* Padding to create space between the profile picture and the user's name
*/}
    <View style={{paddingLeft: 10}}>
      <Text style={{fontSize: 15, fontWeight: '500',}}>{name}</Text>
    </View>
  </View>
    <Feather name="more-vertical" style={{fontSize: 20}} />
  </View>
  <View style={styles.textContainer}>
    <Text>{text}</Text>
  </View>
  {/* If there is no image posted, there is no need to create space for an image */}
  <View>
    {image != null ? (
    <Image source={{uri: image}} style={{width: '100%', height: 180, marginTop:
15,}}/>
    ) : ( <View /> )}
  </View>
  <View style={{
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    paddingHorizontal: 15,
    paddingVertical: 15,
  }}>
```

```
    {/* Icon and likes count -  */}
    <View style={{flexDirection: 'row', alignItems: 'center'}}>
      <AntDesign name='like2' style={{fontSize: 15, paddingRight: 2, color:
'#0C4BAE'}}/>
      <Text style={{fontSize: 12}}>
      Liked by {like ? 'you and' : ''}{' '}
      {like ? likes -1: likes} others
      </Text>
    </View>
    </View>
    {/* Add a comment (profile picture - text input ) */}
    <View style={styles.header}>
      <View style={{flexDirection: 'row', alignItems: 'center'}}>
      <Image source={{uri: userPic}} style={styles.profile_picture}/>
      {/* Padding to create space between the current users picture and the text
box */}
      <View style={{paddingLeft: 10}}>
        <TextInput style={{width: 250}}
              placeholder="Add a comment..."
              maxLength={120} multiline
              numberOfLines={4}
              onChangeText={(comment) => setComment(comment)}
              value={comment}
              />
      </View>
      </View>
      {/* Submit button */}
      <TouchableOpacity style={{paddingRight: 0}} onPress={submitComment}>
        <AntDesign name='rightcircle' style={{fontSize: 22, color: '#0C4BAE'}}/>
      </TouchableOpacity>
    </View>
    {/* Display all comments */}
    <View>
    {comments.map((comment, index) => (
     <Comment comment={comment} key={index} />
    ))}
    </View>
    </KeyboardAwareScrollView>
    </TouchableWithoutFeedback>
    </ScrollView>
  );
}};
```

```
export default CommentScreen;

const styles = StyleSheet.create({
    container: {
        backgroundColor: '#FFF',
    },
    header: {
        flexDirection: 'row',
        alignItems: 'center',
        justifyContent: 'space-between',
        padding: 15,
    },
    profile_picture:{
        width: 40,
        height: 40,
        borderRadius: 100
    },
    textContainer:{
        marginHorizontal: 15,
    },
});
```

4.1.4.8 ChatScreen.js

```
import React, {useEffect, useState} from 'react';
import { StatusBar } from 'expo-status-bar';
import { RefreshControl, StyleSheet, Text, View, Image, TouchableOpacity,
ScrollView } from 'react-native';
// icons
import { FontAwesome5 } from '@expo/vector-icons';
import { FontAwesome } from '@expo/vector-icons';
import { Ionicons } from '@expo/vector-icons';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const ChatScreen = ({ navigation }) => { {
  // state variables
  const [chatrooms, setChatrooms] = useState([]);
  const [refreshing, setRefreshing] = useState(false);

  // get all chat rooms to begin with
  // these will be filtered later
  const getChatrooms = async () => {
```

```
    const chatrooms = [];
    await db.collection('chatrooms').get().then(querySnapshot => {
      querySnapshot.forEach(doc => {
        chatrooms.push(doc.data());
      });
    });
    setChatrooms(chatrooms);

    // loop through all rooms to see if either of the ids match the current user
    // if they do, then add the room to the list of rooms
    const filteredChatrooms = [];
    chatrooms.forEach(room => {
       if(room.users[0].id === auth.currentUser.uid || room.users[1].id ===
auth.currentUser.uid){
          filteredChatrooms.push(room);
       }
    });
    setChatrooms(filteredChatrooms);

  }

  // get the chatrooms when the screen is loaded
  useEffect(() => {
    getChatrooms();
  }, []);

  // timeout function used when refreshing the chatrooms
  const wait = (timeout) => {
    return new Promise(resolve => {
      setTimeout(resolve, timeout);
    });
  }

  // function to refresh the chatrooms
  const onRefresh = () => {
    setRefreshing(true);
    getChatrooms();
    wait(1500).then(() => setRefreshing(false));
  }

  return (
    <View style={styles.container}>
    <View style={styles.header}>
```

```
    <View style={styles.left} />
    <View style={styles.middle}>
     <Text style={styles.text}>Chat</Text>
    </View>
    <View style={styles.right}>
     <TouchableOpacity onPress={() => navigation.navigate('Users')}>
      <Ionicons name="create-sharp" size={24} color="#000" />
     </TouchableOpacity>
    </View>
   </View>
   <ScrollView style={{marginLeft: 15}} refreshControl={
    <RefreshControl
     refreshing={refreshing}
     onRefresh={onRefresh}
    />
   }>
    {chatrooms.map((chatroom, index ) => (
     <View key={index} >
      <View style={{marginBottom: 15}}>
       {/* The chatroom contains two users in an array, so we need to check if the
current user is the first or second user in the chatroom,
          if the array[0].id === auth.currentUser.uid, then the current user is the
first user in the chatroom
          so we need to pass the second users name, id and avatar to the messages
screen
       */}
       {chatroom.users[0].id === auth.currentUser.uid ? (
        <TouchableOpacity style={styles.chatroom} onPress={() =>
navigation.navigate('Messages', {name: chatroom.users[1].name, id:
chatroom.users[1].id, avatar: chatroom.users[1].profile_picture})}>
         <View style={{flexDirection: 'row', alignItems: 'center'}}>
          <Image source={{ uri: chatroom.users[1].profile_picture }}
style={styles.profile_picture}/>
         <View style={{paddingLeft: 10}}>
           <Text style={{fontSize: 15, fontWeight:
'500',}}>{chatroom.users[1].name}</Text>
           <Text style={{paddingTop: 5}}>{chatroom.lastMessage}</Text>
         </View>
         </View>
        </TouchableOpacity>
       ) : (
```

```
          <TouchableOpacity style={styles.chatroom} onPress={() =>
navigation.navigate('Messages', {name: chatroom.users[0].name, id:
chatroom.users[0].id, avatar: chatroom.users[0].profile_picture})}>
            <View style={{flexDirection: 'row', alignItems: 'center'}}>
              <Image source={{ uri: chatroom.users[0].profile_picture }}
style={styles.profile_picture}/>
            <View style={{paddingLeft: 10}}>
              <Text style={{fontSize: 15, fontWeight:
'500',}}>{chatroom.users[0].name}</Text>
              <Text style={{paddingTop: 5}}>{chatroom.lastMessage}</Text>
            </View>
            </View>
          </TouchableOpacity>
         )}
        </View>
      </View>
    ))}
  </ScrollView>
  </View>
 );
}};

export default ChatScreen;

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: '#fff',
 },
 header: {
   flexDirection: 'row',
   justifyContent: 'space-between',
   alignItems: 'flex-end',
   backgroundColor: '#FFF',
   height: 50,
   marginTop: 45,
   borderBottomWidth: 0.5,
   borderBottomColor: '#E5E5E5',
   marginBottom: 10,
 },
 left: {
   width: 50,
   height: 50,
```

```
  },
  middle: {
    width: 100,
    height: 50,
    justifyContent: 'center',
    alignItems: 'center',
  },
  right: {
    width: 50,
    height: 50,
    marginRight: 10,
    justifyContent: 'center',
    alignItems: 'center',
  },
  text: {
    fontWeight: '700',
    fontSize: 16,
    color: '#000',
  },
  profile_picture: {
    width: 40,
    height: 40,
    borderRadius: 20,
  },
})
```

4.1.4.9 CreateChatScreen.js

```
import React, {useEffect, useState} from 'react';
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View, Image, TouchableOpacity, ScrollView, TextInput }
from 'react-native';
// icons
import { FontAwesome5 } from '@expo/vector-icons';
import { FontAwesome } from '@expo/vector-icons';
import { Ionicons } from '@expo/vector-icons';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const CreateChatScreen = ({ navigation }) => { {

  // get all users from the firebase database
  const [users, setUsers] = useState([]);
```

```
  const [filteredUsers, setFilteredUsers] = useState([]);
  const [search, setSearch] = useState('');
  const [currentUserName, setCurrentUserName] = useState('');
  const [currentUserPic, setCurrentUserPic] = useState('');

  // get all users except for the current user
  const getUsers = async () => {
    const users = [];
    await db.collection('users').where('email', '!=',
auth.currentUser.email).get().then(querySnapshot => {
      querySnapshot.forEach(doc => {
        users.push(doc.data());
      });
    });
    setUsers(users);
    setFilteredUsers(users);
  }

  // get current user from the firebase database
  const getCurrentUser = async () => {
    await db.collection('users').where('email', '==',
auth.currentUser.email).get().then(querySnapshot => {
      querySnapshot.forEach(doc => {
        setCurrentUserName(doc.data().name);
        setCurrentUserPic(doc.data().profile_picture);
      });
    });
  }

  // get the users and the current user when the screen is loaded
  useEffect(() => {
    getUsers();
    getCurrentUser();
  }, []);

  // function to filter the users based on the search input
  const searchFilter = (text) => {
   if(text){
     const newData = users.filter(user => {
       const userData = user.name ? user.name.toUpperCase() : ''.toUpperCase();
       const textData = text.toUpperCase();
       return userData.indexOf(textData) > -1;
     })
```

```
        setFilteredUsers(newData);
        setSearch(text);
      } else {
        setFilteredUsers(users);
        setSearch('');
      }
    }

    return (
      <View style={styles.container}>
        <TextInput
          style={styles.search}
          placeholder="Search for a user.."
          onChangeText={(text) => searchFilter(text)}
          value={search} />
        <ScrollView>
          {filteredUsers.map((user, index) => (
            <TouchableOpacity key={index} style={styles.userContainer}
            onPress={() => navigation.navigate('Messages', {currentUserName:
currentUserName, currentUserPic: currentUserPic, name: user.name, id: user.id,
avatar: user.profile_picture})}>
              <Image source={{uri: user.profile_picture}} style={styles.image} />
              <View style={styles.nameContainer}>
                <Text style={styles.name}>{user.name}</Text>
              </View>
            </TouchableOpacity>
          ))}
        </ScrollView>
      </View>
    );
};

export default CreateChatScreen;

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
  },
  search: {
    height: 40,
    borderWidth: 1,
    borderColor: '#E5E5E5',
```

```
    marginTop: 10,
    marginLeft: 10,
    marginRight: 10,
    padding: 10,
    borderRadius: 5,
  },
  userContainer: {
    flexDirection: 'row',
    alignItems: 'center',
    padding: 10,
  },
  image: {
    marginLeft: 10,
    width: 40,
    height: 40,
    borderRadius: 20,
  },
  nameContainer: {
    marginLeft: 15,
  },
  name: {
    fontSize: 14,
    fontWeight: '500',
  },
  header: {
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'space-between',
    padding: 15,
  },
  profile_picture:{
     width: 40,
     height: 40,
     borderRadius: 100,
     marginTop: 0,
  },
});
```

ChatRoomScreen.js

```
import React, { useState, useCallback, useEffect } from 'react'
import { StatusBar } from 'expo-status-bar';
import { StyleSheet, Text, View, Image, TouchableOpacity, ScrollView } from 'react-
native';
// gifted chat
```

```
import { GiftedChat, Bubble } from 'react-native-gifted-chat'
// icons
import { FontAwesome5 } from '@expo/vector-icons';
import { FontAwesome } from '@expo/vector-icons';
import { Ionicons } from '@expo/vector-icons';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const ChatRoomScreen = ({ navigation, route }) => { {
   // get the current user's name, profile picture and the other user's name, id and
profile picture from the route
   const { currentUserName, currentUserPic, name, id, avatar } = route.params;
   // state variable to hold the messages
   const [messages, setMessages] = useState([]);

   useEffect(() => {
    // current user's id
    const uid = auth.currentUser.uid;

    /*
    Note: uid + id will not work when generating a chatroom id

    For example: uid = '123' and id = '456'. The chatroom id will be '123456'.

    However, when the current user is '456' and the other user is '123', the
chatroom id will be '456123'

    To fix this, a ternary operator is used to generate the chatroom id.
    if(uid > id) then id + uid else uid + id

    Now, since 123 is less than 456, the chatroom id will always be '123456' for
both users.
    */

    const docId = uid > id ? id + "-" + uid : uid + "-" + id;
    // create a reference to the chatroom's messages collection and order the
messages by createdAt
    const ref = db.collection('chatrooms')
                  .doc(docId)
                  .collection('messages')
                  .orderBy('createdAt', 'desc')
    ref.onSnapshot(querySnapshot => {
```

```
      const allMessages = querySnapshot.docs.map(query => {
      // bug: sometimes the createdAt field was not being set, so we need to check
if it is set or not
      const data = query.data();
      if(data.createdAt){
        return {
          ...query.data(),
          createdAt: query.data().createdAt.toDate()
        }
      } else{
        return {
          ...query.data(),
          createdAt: new Date()
        }
      }
    })
    // update the messages state variable
    setMessages(allMessages);
    })

    // add the two user ids to the chatroom collection
    const chatroomRef = db.collection('chatrooms').doc(docId);
    chatroomRef.get().then(doc => {
      if(!doc.exists){
        chatroomRef.set({
          // create a user array and store two user objects
          // store the last message in the chatroom
          lastMessage: 'No messages yet.',
          createdAt: new Date(),
          users: [
            {
              id: uid,
              name: currentUserName,
              profile_picture: currentUserPic
            },
            {
              id: id,
              name: name,
              profile_picture: avatar
            }
          ]
        })
      }
```

```
    }
  )
  }, [])

  // function to send a message
  const onSend = useCallback((messageArray) => {
    const message = messageArray[0];
    const newMessage = {
      ...message,
      sentBy: auth.currentUser.uid,
      sendTo: id,
      createdAt: new Date(),
    };
    console.log(newMessage);
    // update the messages
    setMessages(previousMessages => GiftedChat.append(previousMessages,
newMessage));
    const uid = auth.currentUser.uid;
    const docId = uid > id ? id + "-" + uid : uid + "-" + id;
    // add the message to the chatroom's messages collection
    db.collection('chatrooms')
    .doc(docId)
    .collection('messages')
    .add({
      ...newMessage,
      createdAt: firebase.firestore.FieldValue.serverTimestamp(),
    });

    // update the last message in the chatroom and the createdAt field
    db.collection('chatrooms')
    .doc(docId)
    .update({
      lastMessage: message.text,
      createdAt: firebase.firestore.FieldValue.serverTimestamp(),
    })

  }, []);

  return (
    <GiftedChat
      messages={messages}
      onSend={messages => onSend(messages)}
      user={{
```

```
        _id: auth.currentUser.uid,
      }}
      renderBubble={ (props) => {
       return (
        <Bubble
         {...props}
         wrapperStyle={{
          right: {
           backgroundColor: '#0C4BAE',
           padding: 2,
          },
          left: {
           backgroundColor: '#dfe6e9',
           padding: 2,
          }
         }}
        />
       )
      }}
      />
    )
  }
}

export default ChatRoomScreen;

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: '#fff',
 },
})
```

4.1.4.10 PostScreen.js

```
import React, { Component, useState, useEffect} from 'react';
import { Alert, View, Text,StyleSheet, SafeAreaView, TouchableOpacity, Image,
TextInput, Keyboard, TouchableWithoutFeedback } from 'react-native';
// icons
import { Ionicons } from '@expo/vector-icons';
import AntDesign from 'react-native-vector-icons/AntDesign';
// image picker
import * as ImagePicker from 'expo-image-picker';
```

```
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';
// image storage
import { getStorage, ref, uploadBytesResumable, getDownloadURL, uploadBytes }
from "firebase/storage";

const PostScreen = ({ navigation }) => {{
  // state variables
  // the posts text input and image
  const [text, setText] = useState();
  const [image, setImage] = useState(null);
  //user data
  const [userId, setUserId] = useState();
  const [name, setName] = useState();
  const [profilePicture, setProfilePicture] = useState();
  const [email, setEmail] = useState();
  const [token, setToken] = useState();

  // a function to launch the image library, allowing the user to select an image
  const pickImage = async () => {
    // No permissions request are necessary
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.All,
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1,
    });
    if (!result.cancelled) {
      setImage(result.uri);
    }
  };

  // get the user's token for from user's collection
  const getToken = async () => {
    const data = await
db.collection('users').doc(firebase.auth().currentUser.uid).get();
    setToken(data.data().token);
    // check if token is null
    if (token === null) {
      setToken('');
    }
  };
```

```
// get the current user's info
const getUser = async() => {
  db.collection('users')
  .doc(firebase.auth().currentUser.uid)
  .get()
  .then((documentSnapshot) => {
    if( documentSnapshot.exists ) {
      console.log('User Data', documentSnapshot.data());
      setProfilePicture(documentSnapshot.data().profile_picture);
      setName(documentSnapshot.data().name);
      setUserId(documentSnapshot.data().id);
    }
  })
}

// get the current user on page load
useEffect(() => {
  getToken();
  getUser();
  console.log('token', token);
}, []);

//handle the post's submission
const submitPost = async () => {
  // get the current time
  const time = new Date().getTime();
  let downloadURL = null;
  // check if image is null
  if(image === null) {
    console.log('no image');
  } else {
    // upload the image to firebase storage
    //upload the image to firebase storage
    // create the file metadata
    /** @type {any} */
    const metadata = {
      contentType: 'image/jpeg'
    };
    // upload the image to firebase storage
    const result = await fetch(image);
    // get the blob of the image
    const blob = await result.blob();
```

```
  // create a reference to the location where we want to store the image
  const storage = getStorage();
  const storageRef = ref(storage, 'posts/' + time + '.jpg');
  // upload the image to the storage reference
  const uploadTask = await uploadBytesResumable(storageRef, blob, metadata);
  // get the download url of the image
  downloadURL = await getDownloadURL(storageRef);
  console.log('downloadURL', downloadURL);
}

// get a reference to the document
const docRef = db.collection('posts').doc();
// create a new post object
const post = {
  id: docRef.id,
  user_id: userId,
  name: name,
  profile_picture: profilePicture,
  text: text,
  image: downloadURL,
  time: time,
  created_at: firebase.firestore.FieldValue.serverTimestamp(),
  report_count: 0,
  token: token,
};

// add the post to the database
docRef.set(post)
// alert the user that the post was submitted
.then(() => {
  setText('');
  setImage(null);
  console.log('Post added');
  navigation.navigate('Home');
  Alert.alert(
    "Post Submitted",
    "Your post has been submitted",
    [
      {
        text: "OK",
        onPress: () => navigation.goBack() }
    ],
    { cancelable: false }
```

```
      );
    })
    .catch((error) => {
      console.log('Error adding document: ', error);
    });
  }

  return (
    <TouchableWithoutFeedback onPress={Keyboard.dismiss} accessible={false}>
    <SafeAreaView style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity>
          <Ionicons name='md-arrow-back' size={24}color='#000'></Ionicons>
        </TouchableOpacity>
        <TouchableOpacity onPress={submitPost}>
          <Text style={{color:'#000', fontWeight: 'bold', marginTop: 2}}>Submit Post
</Text>
        </TouchableOpacity>
      </View>
      <View style={styles.inputContainer}>
        <Image source={{uri: profilePicture}} style={styles.avatar}/>
        <TextInput
        autoFocus={true}
        multiline={true}
        numberOfLines={4}
        style={{flex:1}}
        placeholder='Want to share something?'
        onChangeText={(text) => setText(text)}
        value={text}></TextInput>
        <TouchableOpacity style={styles.photoIcon}>
          <Ionicons name="md-camera" size={32} color="#000" onPress={pickImage}/>
        </TouchableOpacity>
      </View>
      <View>
      {image != null ? (
      <View style={{marginHorizontal:32, marginTop:10, height:300}}>
          <Image source={{uri: image}} style={{width:'100%', height:'100%'}}></Image>
      </View> ) : ( <View /> )}
      {image != null ? (
        <TouchableOpacity>
          <Text onPress={(image) => setImage(null)} style={styles.removeText}>
Remove </Text>
        </TouchableOpacity>
```

```
      ) : ( <View />)}
      </View>
      <View style={{alignItems: 'center'}}>
      </View>
    </SafeAreaView>
    </TouchableWithoutFeedback>
    );
  }
}

export default PostScreen;

const styles=StyleSheet.create({
    container:{
      flex:1,
      backgroundColor: '#FFF'
    },
    header:{
      flexDirection:'row',
      justifyContent:'space-between',
      paddingHorizontal:35,
      paddingVertical:10,
      borderBottomWidth:1,
      borderBottomColor:'#D8D9DB'
    },
    inputContainer:{
      margin:32,
      flexDirection:'row',
    },
    avatar:{
      width:48,
      height:48,
      borderRadius:24,
      marginRight:16
    },
    photoIcon:{
      alignItems:'flex-end',
      marginHorizontal: 10
    },
    removeText:{
      color: 'red',
      fontWeight: '700',
      textAlign: 'center',
```

```
      marginTop: 10,
    },
})
```

4.1.4.11 NotificationsScreen.js

```
import React, { Component, useState, useEffect} from 'react';
import { Alert, View, Text,StyleSheet, SafeAreaView, TouchableOpacity, Image,
TextInput, Keyboard, TouchableWithoutFeedback } from 'react-native';
// icons
import { Ionicons } from '@expo/vector-icons';
import AntDesign from 'react-native-vector-icons/AntDesign';
// image picker
import * as ImagePicker from 'expo-image-picker';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';
// image storage
import { getStorage, ref, uploadBytesResumable, getDownloadURL, uploadBytes }
from "firebase/storage";

const PostScreen = ({ navigation }) => {{
  // state variables
  // the posts text input and image
  const [text, setText] = useState();
  const [image, setImage] = useState(null);
  //user data
  const [userId, setUserId] = useState();
  const [name, setName] = useState();
  const [profilePicture, setProfilePicture] = useState();
  const [email, setEmail] = useState();
  const [token, setToken] = useState();

  // a function to launch the image library, allowing the user to select an image
  const pickImage = async () => {
    // No permissions request are necessary
    let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.All,
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1,
    });
    if (!result.cancelled) {
      setImage(result.uri);
```

```
  }
 };

 // get the user's token for from user's collection
 const getToken = async () => {
  const data = await
db.collection('users').doc(firebase.auth().currentUser.uid).get();
  setToken(data.data().token);
  // check if token is null
  if (token === null) {
   setToken('');
  }
 };

 // get the current user's info
 const getUser = async() => {
  db.collection('users')
  .doc(firebase.auth().currentUser.uid)
  .get()
  .then((documentSnapshot) => {
   if( documentSnapshot.exists ) {
    console.log('User Data', documentSnapshot.data());
    setProfilePicture(documentSnapshot.data().profile_picture);
    setName(documentSnapshot.data().name);
    setUserId(documentSnapshot.data().id);
   }
  })
 }

 // get the current user on page load
 useEffect(() => {
  getToken();
  getUser();
  console.log('token', token);
 }, []);

 //handle the post's submission
 const submitPost = async () => {
  // get the current time
  const time = new Date().getTime();
  let downloadURL = null;
  // check if image is null
  if(image === null) {
```

```
  console.log('no image');
} else {
  // upload the image to firebase storage
  //upload the image to firebase storage
  // create the file metadata
  /** @type {any} */
  const metadata = {
    contentType: 'image/jpeg'
  };
  // upload the image to firebase storage
  const result = await fetch(image);
  // get the blob of the image
  const blob = await result.blob();
  // create a reference to the location where we want to store the image
  const storage = getStorage();
  const storageRef = ref(storage, 'posts/' + time + '.jpg');
  // upload the image to the storage reference
  const uploadTask = await uploadBytesResumable(storageRef, blob, metadata);
  // get the download url of the image
  downloadURL = await getDownloadURL(storageRef);
  console.log('downloadURL', downloadURL);
}

// get a reference to the document
const docRef = db.collection('posts').doc();
// create a new post object
const post = {
  id: docRef.id,
  user_id: userId,
  name: name,
  profile_picture: profilePicture,
  text: text,
  image: downloadURL,
  time: time,
  created_at: firebase.firestore.FieldValue.serverTimestamp(),
  report_count: 0,
  token: token,
};

// add the post to the database
docRef.set(post)
// alert the user that the post was submitted
.then(() => {
```

```
    setText('');
    setImage(null);
    console.log('Post added');
    navigation.navigate('Home');
    Alert.alert(
      "Post Submitted",
      "Your post has been submitted",
      [
        {
          text: "OK",
          onPress: () => navigation.goBack() }
      ],
      { cancelable: false }
    );
  })
  .catch((error) => {
    console.log('Error adding document: ', error);
  });
 }

 return (
  <TouchableWithoutFeedback onPress={Keyboard.dismiss} accessible={false}>
  <SafeAreaView style={styles.container}>
    <View style={styles.header}>
      <TouchableOpacity>
        <Ionicons name='md-arrow-back' size={24}color='#000'></Ionicons>
      </TouchableOpacity>
      <TouchableOpacity onPress={submitPost}>
        <Text style={{color:'#000', fontWeight: 'bold', marginTop: 2}}>Submit Post
</Text>
      </TouchableOpacity>
    </View>
    <View style={styles.inputContainer}>
      <Image source={{uri: profilePicture}} style={styles.avatar}/>
      <TextInput
      autoFocus={true}
      multiline={true}
      numberOfLines={4}
      style={{flex:1}}
      placeholder='Want to share something?'
      onChangeText={(text) => setText(text)}
      value={text}></TextInput>
      <TouchableOpacity style={styles.photoIcon}>
```

```
        <Ionicons name="md-camera" size={32} color="#000" onPress={pickImage}/>
      </TouchableOpacity>
    </View>
    <View>
    {image != null ? (
    <View style={{marginHorizontal:32, marginTop:10, height:300}}>
      <Image source={{uri: image}} style={{width:'100%', height:'100%'}}></Image>
    </View> ) : ( <View /> )}
    {image != null ? (
      <TouchableOpacity>
        <Text onPress={(image) => setImage(null)} style={styles.removeText}>
Remove </Text>
      </TouchableOpacity>
    ) : ( <View />)}
    </View>
    <View style={{alignItems: 'center'}}>
    </View>
   </SafeAreaView>
   </TouchableWithoutFeedback>
   );
  }
}

export default PostScreen;

const styles=StyleSheet.create({
   container:{
     flex:1,
     backgroundColor: '#FFF'
    },
    header:{
     flexDirection:'row',
     justifyContent:'space-between',
     paddingHorizontal:35,
     paddingVertical:10,
     borderBottomWidth:1,
     borderBottomColor:'#D8D9DB'
    },
    inputContainer:{
     margin:32,
     flexDirection:'row',
    },
    avatar:{
```

```
    width:48,
    height:48,
    borderRadius:24,
    marginRight:16
  },
  photoIcon:{
    alignItems:'flex-end',
    marginHorizontal: 10
  },
  removeText:{
    color: 'red',
    fontWeight: '700',
    textAlign: 'center',
    marginTop: 10,
  },
})
```

4.1.4.12 ProfileScreen.js

```
import React, {useEffect, useState} from 'react';
import { StatusBar } from 'expo-status-bar';
import { RefreshControl, StyleSheet, Text, View, Image, TouchableOpacity,
ScrollView } from 'react-native';
import { CONDITIONS } from '../data/conditions';
//components
import { Divider } from 'react-native-elements';
import Stories from '../components/Stories';
import Post from '../components/Post';
import {windowHeight, windowWidth} from '../utils/Dimensions';
// icons
import { FontAwesome5 } from '@expo/vector-icons';
import { FontAwesome } from '@expo/vector-icons';
import { Ionicons } from '@expo/vector-icons';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const ProfileScreen = ({ navigation }) => { {
  // state variables
  const [text, setText] = useState(null)
  const [profilePicture, setProfilePicture] = useState();
  const [posts, setPosts] = useState([]);
  const [bio, setBio] = useState();
  const [condition, setCondition] = useState();
```

```
  const [similarUsers, setSimilarUsers] = useState([]);
  const [followerCount, setFollowerCount] = useState(0);
  const [followingCount, setFollowingCount] = useState(0);
  const [refreshing, setRefreshing] = useState(false);

  // get the current user's information
  const getUser = async() => {
    db.collection('users')
    .doc(firebase.auth().currentUser.uid)
    .get()
    .then((documentSnapshot) => {
      if(documentSnapshot.exists ) {
        setBio(documentSnapshot.data().bio);
        setText(documentSnapshot.data().name);
        setCondition(documentSnapshot.data().condition);
        console.log('condition: ', condition);
        setProfilePicture(documentSnapshot.data().profile_picture);
        console.log('User Data', documentSnapshot.data());
      } else{
        console.log('User not found');
      }
    })
  }

  // get similar users where the user's condition is similar to the current user's
condition
  const getSimilarUsers = async() => {
    db.collection('users')
    .where('condition', '==', condition)
    .get()
    .then((snapshot) => {
      const similarUsers = snapshot.docs.map(doc => doc.data());
      setSimilarUsers(similarUsers);
    })
  }

  // check how many people the user is following
  const getFollowingCount = async() => {
    db.collection('following')
    .doc(firebase.auth().currentUser.uid)
    .collection('userFollowing')
    .get()
    .then((querySnapshot) => {
```

```
    setFollowingCount(querySnapshot.size);
  })
 }

 // check how many followers the user has
 const getFollowerCount = async() => {
  db.collection('followers')
  .doc(firebase.auth().currentUser.uid)
  .collection('userFollowers')
  .get()
  .then((querySnapshot) => {
    setFollowerCount(querySnapshot.size);
  })
 }

 // load the current user and similar users on page load
 useEffect(async () => {
  await getUser();
  // (not working) await getSimilarUsers();
  await getFollowingCount();
  await getFollowerCount();
 }, [])


 // fetch all posts for current user using id
 useEffect(() => {
  // create a reference to the posts collection
  const postsRef = db.collection('posts');
  // create a query against the collection
  const query = postsRef.where('user_id', '==',
firebase.auth().currentUser.uid).orderBy('created_at', 'desc');
  // listen to the query
  query.onSnapshot(snapshot => {
    setPosts(snapshot.docs.map(doc => doc.data()));
  }
  );
 }, []);

 // check if the user has a bio and / or condition set
 // if they do, render a component with the bio and condition
 const aboutMeComponent = () => {
  if(bio && condition) {
    return (
```

```
      <View style={styles.aboutMeContainer}>
        <Text style={{fontSize: 16, fontWeight: '600', marginBottom: 10,}}>About
Me</Text>
        <View style={{flexDirection: 'row', alignItems: 'center'}}>
          <FontAwesome name="book" size={20} color="black" />
          <Text style={styles.aboutMeText}>{bio}</Text>
        </View>
        <View style={{flexDirection: 'row', alignItems: 'center', marginTop: 10}}>
          <FontAwesome5 name="heartbeat" size={20} color="black" />
          <Text style={styles.aboutMeText}>{condition}</Text>
        </View>
      </View>
      )
    }

    // if the user has a bio but not a condition, render a component with the bio
    else if(bio) {
      return (
        <View>
          <Text style={{fontSize: 16, fontWeight: '600', marginBottom: 10,}}>About
Me</Text>
        <View style={{flexDirection: 'row', alignItems: 'center'}}>
          <FontAwesome name="book" size={20} color="black" />
          <Text style={styles.aboutMeText}>{bio}</Text>
        </View>
      </View>
      )
    }

    // if the user has a condition but not a bio, render a component with the
condition
    else if(condition) {
      return (
        <View>
          <Text style={{fontSize: 16, fontWeight: '600', marginBottom: 10,}}>About
Me</Text>
        <View style={{flexDirection: 'row', alignItems: 'center'}}>
          <FontAwesome5 name="heartbeat" size={20} color="black" />
          <Text style={styles.aboutMeText}>{condition}</Text>
        </View>
      </View>
      )
    }
```

```
  }

  // render a component of recommended users
  const similarUsersComponent = () => {
    if(similarUsers.length > 0) {
      return (
        <View>
          <Text style={{fontSize: 16, fontWeight: '600', marginTop: 10,}}>Similar
Users</Text>
        <View style={styles.similarUsersContainer}>
          <ScrollView horizontal={true} showsHorizontalScrollIndicator={false}>
        {similarUsers.map((user, index) => {
          return (
            <View key={index} style={styles.similarUser}>
              <Image source={{uri: user.profile_picture}} style={styles.similarUserImage}
/>
              <Text style={styles.similarUserName}>{user.name}</Text>
              <TouchableOpacity style={styles.similarUserBtn}>
                <Text style={{color: '#FFF'}}>View Profile</Text>
              </TouchableOpacity>
            </View>
          )
        })}
        </ScrollView>
        </View>
        </View>
      )
    }
  }

  // timeout function used when refreshing the page
  const wait = (timeout) => {
    return new Promise(resolve => {
      setTimeout(resolve, timeout);
    });
  }

  // refresh the page function
  const onRefresh = () => {
    setRefreshing(true);
    getUser();
    wait(1500).then(() => setRefreshing(false));
  }
```

```
  return (
    <ScrollView style={styles.container} refreshControl={
      <RefreshControl
        refreshing={refreshing}
        onRefresh={onRefresh}
      />
    }>
    <View style={{marginTop: 40,}}>
    <Ionicons name="menu" size={32} color="#000" style={{marginLeft: 5}}
onPress={() => navigation.openDrawer()}></Ionicons>
    </View>
    <View style={styles.image_container}>
    <View style={{alignItems: 'center'}}>
      <Image source={{url: profilePicture}}  style={styles.userImg} />
    </View>
    <View style={{alignItems: 'center'}}>
      <Text style={styles.username}>{text}</Text>
    </View>
    </View>
    <View style={styles.counters_container}>
        <View style={{justifyContent: 'center'}}>
          <Text style={styles.title}>{posts.length}</Text>
          <Text style={styles.subtitle}>Posts</Text>
        </View>
        <View style={{justifyContent: 'center'}}>
          <Text style={styles.title}>{followerCount}</Text>
          <Text style={styles.subtitle}>Followers</Text>
        </View>
        <View style={{justifyContent: 'center'}}>
          <Text style={styles.title}>{followingCount}</Text>
          <Text style={styles.subtitle}>Following</Text>
        </View>
      </View>
      <View>
      <TouchableOpacity style={styles.editProfileBtn} onPress={() =>
navigation.navigate('Edit Profile')}>
        <Text style={styles.buttonText}>Edit Profile</Text>
      </TouchableOpacity>
      </View>
      <View style={{marginLeft: 20, marginTop: 0}}>
      <View style={styles.aboutMe}>
       {aboutMeComponent()}
```

```
      </View>
      <View style={styles.similarUsers}>
        {similarUsersComponent()}
      </View>
      </View>
      {posts.map((post, index) => (
        <Post post={post} key={index} />
      ))}
    </ScrollView>
  );
}};

export default ProfileScreen;

const styles = StyleSheet.create({
  container:{
    flex: 1,
    backgroundColor: '#FFF',
  },
  image_container:{
    alignItems: 'center',
  },
  userImg: {
    height: 125,
    width: 125,
    borderRadius: 75,
  },
  username: {
    marginTop: 15,
    fontSize: 20,
  },
  counters_container: {
    flexDirection: 'row',
    justifyContent: 'space-around',
    width: '100%',
    marginTop: 15,
  },
  title: {
    fontSize: 20,
    fontWeight: 'bold',
    marginBottom: 5,
    textAlign: 'center',
  },
```

```
subtitle: {
  fontSize: 12,
  color: '#000',
  textAlign: 'center',
},
editProfileBtn:{
  alignItems: 'center',
  justifyContent: 'center',
  margin: 15,
  borderRadius: 3,
  backgroundColor: '#FFF',
  height: windowHeight / 20,
  borderColor: 'grey',
  borderWidth: 1,
},
buttonText:{
  fontSize: 14,
  color: '#000',
  fontFamily: 'Verdana',
},
aboutMeContainer: {
  flexDirection: 'column',
},
aboutMeText:{
  fontSize: 14,
  marginLeft: 10,
},
similarUsersContainer: {
  marginTop: 10,
  flexDirection: 'row',
  justifyContent: 'center',
  alignItems: 'center',
},
similarUser:{
  marginRight: 5,
  alignItems: 'center',
  justifyContent: 'center',
  borderWidth: 0.5,
  borderColor: '#DEDEDE',
  borderRadius: 2,
  height: 160,
  width: 130,
  padding: 5,
```

```
  },
  similarUserImage:{
    height: 60,
    width: 60,
    borderRadius: 30,
    marginBottom: 10,
  },
  similarUserBtn:{
    alignItems: 'center',
    justifyContent: 'center',
    marginTop: 10,
    borderRadius: 3,
    backgroundColor: '#0C4BAE',
    padding: 8,
  },
});
```

4.1.4.13 EditProfileScreen.js

```
import React, {useEffect, useState} from 'react';
import { StatusBar } from 'expo-status-bar';
import { Alert, StyleSheet, TextInput, SafeAreaView, Text, View, Image,
TouchableOpacity, ScrollView, TouchableWithoutFeedback, Keyboard } from
'react-native';
// mobiles height and width
import {windowHeight, windowWidth} from '../utils/Dimensions';
// load all the conditions
import { conditions } from '../data/conditions';
// keyboard aware scroll view - does not hide text inputs when keyboard is open
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-
view'
//import SearchableDropdown component
import SearchableDropdown from 'react-native-searchable-dropdown';
// icons
import { Ionicons } from '@expo/vector-icons';
// image picker
import * as ImagePicker from 'expo-image-picker';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const EditProfile = ({ navigation }) => {

  // state variables
```

```
const [profilePicture, setProfilePicture] = useState();
const [newImage, setNewImage] = useState(null);
const [name, setName] = useState();
const [condition, setCondition] = useState();
const [placeholder, setPlaceholder] = useState('Select a condition..');
const [bio, setBio] = useState();

// image picker
const pickImage = async () => {
   let result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.All,
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1
   });
   if (!result.cancelled) {
      setNewImage(result.uri);
   }
};

// function to get the current user's information
const getUser = async() => {
  db.collection('users')
  .doc(firebase.auth().currentUser.uid)
  .get()
  .then((documentSnapshot) => {
    if( documentSnapshot.exists ) {
      setName(documentSnapshot.data().name);
      setProfilePicture(documentSnapshot.data().profile_picture);
      console.log('User Data', documentSnapshot.data());
    } else{
      console.log('User not found');
    }
  })
}

// get the current user on page load
useEffect(() => {
  getUser();
}, []);

// create a deep copy of the conditions array
```

```
  // because I need to remove the description, symptoms, and treatment from the
array
  // so that it can be used in a dropdown menu
  const items = JSON.parse(JSON.stringify(conditions));

  // remove attributes from conditions
  // only need the name and id for dropdown menu
  items.forEach(items => {
    delete items.description;
    delete items.symptoms;
    delete items.treatment;
  });

  // function to handle dropdown menu change
  const handleConditionChange = (condition) => {
    setCondition(condition.name);
    console.log('Condition: ', condition.name);
    setPlaceholder(condition.name);
  };

  // function to handle submit
  const handleSubmit = async () => {
    // check if user has selected a condition
    if(condition === undefined) {
      // do nothing
    } else {
      // update user's condition
      db.collection('users')
      .doc(firebase.auth().currentUser.uid)
      .update({
        condition: condition
      })
      .then(() => {
        console.log('Condition updated');
      })
      .catch((error) => {
        console.log('Error updating condition: ', error);
      });
    }
    // update user's bio
    if(bio === undefined) {
      // do nothing
    } else {
```

```
    db.collection('users')
    .doc(firebase.auth().currentUser.uid)
    .update({
      bio: bio
    })
    .then(() => {
      console.log('Bio updated');
    })
    .catch((error) => {
      console.log('Error updating bio: ', error);
    });
  }
  // if any of the above are not empty, alert the user
  if(condition !== undefined || bio !== undefined) {
    Alert.alert(
      'Success!',
      'Your profile has been updated!',
      [
        {text: 'OK', onPress: () => navigation.navigate('ProfileScreen')},
      ],
      {cancelable: false},
    );
  }
};

 return (
  <SafeAreaView style={styles.container}>
    <View style={styles.content_container}>
    <View style={{alignItems: 'center'}}>
    <TouchableOpacity style={styles.imagePlaceholder} onPress={pickImage}>
        <Image source={{uri: profilePicture}} style={styles.image} />
        <Ionicons name="ios-add" size={32} color="#FFF"/>
    </TouchableOpacity>
    </View>
    <View style={{alignItems: 'center'}}>
      <Text style={styles.username}>{name}</Text>
    </View>
    </View>
    <View style={styles.info_container}>
    <Text style={styles.title}>Bio</Text>
    <TextInput style={styles.input} placeholder="Tell us about yourself.."
onChangeText={setBio} />
      <Text style={styles.title}>Condition</Text>
```

```
      <SearchableDropdown
        onTextChange={(text) => console.log(text)}
        onItemSelect={(item) => handleConditionChange(item)}
         //suggestion container style
        containerStyle={{ padding: 5 }}
        //inserted text style
        textInputStyle={{ padding: 12, borderWidth: 1, borderColor: '#ccc',
backgroundColor: '#FAFAFA'}}
        itemStyle={{ padding: 10, marginTop: 2, backgroundColor: '#F5F5F5',
borderColor: '#bbb', borderWidth: 1,}}
        itemTextStyle={{ color: '#222',}}
        //items container style you can pass maxHeight
        itemsContainerStyle={{ maxHeight: '60%'}}
        //mapping of item array (conditions)
        items={items}
        //place holder for the search input
        placeholder={placeholder}
        placeholderTextColor="#000"
        //reset textInput Value with true and false state
        resetValue={false}
        //To remove the underline from the android input
        underlineColorAndroid="transparent"
      />
      <TouchableOpacity style={styles.updateBtn} onPress={handleSubmit}>
        <Text style={styles.buttonText}>Update</Text>
      </TouchableOpacity>
    </View>
  </SafeAreaView>
 );
};

export default EditProfile;

const styles = StyleSheet.create({
 container:{
  flex: 1,
  backgroundColor: '#FFF',
 },
 content_container:{
  alignItems: 'center',
  marginTop: 10,
 },
 imagePlaceholder: {
```

```
  width: 100,
  height: 100,
  backgroundColor: '#e1e2e6',
  borderRadius: 50,
  marginTop: 10,
  justifyContent: 'center',
  alignItems: 'center'
},
image: {
    position: 'absolute',
    width: 100,
    height: 100,
    borderRadius: 50
},
username: {
  marginTop: 15,
  fontSize: 20,
},
info_container:{
  marginTop: 15,
  marginLeft: 15,
  marginRight: 15,
},
title:{
  fontWeight: 'bold',
  fontSize: 18,
  marginBottom: 5,
  marginLeft: 5,
},
input:{
  marginBottom: 15,
  marginLeft: 5,
},
updateBtn:{
  alignItems: 'center',
  justifyContent: 'center',
  marginTop: 15,
  borderRadius: 3,
  backgroundColor: '#0C4BAE',
  width: '100%',
  height: windowHeight / 20,
  borderColor: '#0C4BAE',
  borderWidth: 1,
```

```
  },
  buttonText:{
    color: '#FFF',
    fontSize: 18,
  }
});
```

4.1.4.14 LearnMore.js

```
import React, {useEffect, useState} from 'react';
import { StatusBar } from 'expo-status-bar';
import { Alert, StyleSheet, TextInput, SafeAreaView, Text, View, Image,
TouchableOpacity, ScrollView } from 'react-native';
import {windowHeight, windowWidth} from '../utils/Dimensions';
import { conditions } from '../data/conditions';
//import SearchableDropdown component
import SearchableDropdown from 'react-native-searchable-dropdown';
import { Ionicons } from '@expo/vector-icons';
//firebase
import firebase from 'firebase/compat/app';
import { auth, db } from '../config/Firebase';

const LearnMore = ({ navigation }) => { {
    // state variables
    const [placeholder, setPlaceholder] = useState('Search for a condition..');
    const [id, setId] = useState('');
    const [name, setName] = useState();
    const [description, setDescription] = useState();
    const [symptoms, setSymptoms] = useState();
    const [treatment, setTreatment] = useState();

   // create a deep copy of the conditions array
   // because I need to remove the description, symptoms, and treatment from the
array
   // so that it can be used in a dropdown menu
   const items = JSON.parse(JSON.stringify(conditions));

   // remove attributes from conditions
   // only need the name and id for dropdown menu
   items.forEach(items => {
      delete items.description;
      delete items.symptoms;
      delete items.treatment;
   });
```

```
// function to handle condtion selection
const handleConditionChange = (item) => {
   setId(item.id);
   setName(item.name);
   // loop through the conditions array to find the condition that was selected
   // and set the description, symptoms, and treatment to the state variables
   for(let i = 0; i < conditions.length; i++) {
      if(conditions[i].id === item.id) {
         setDescription(conditions[i].description);
         setSymptoms(conditions[i].symptoms);
         setTreatment(conditions[i].treatment);
      }
   }
}

// check if the user has selected a condition
// return information if they have, otherwise return an empty view
const displayConditions = () => {
   if(id !== '') {
      return (
         <View style={styles.content_container}>
            <Text style={styles.title}>Name:</Text>
            <Text style={styles.text}>{name}</Text>
            <Text style={styles.title}>Description:</Text>
            <Text style={styles.text}>{description}</Text>
            <Text style={styles.title}>Symptoms:</Text>
            <Text style={styles.text}>{symptoms}</Text>
            <Text style={styles.title}>Treatment:</Text>
            <Text style={styles.text}>{treatment}</Text>
         </View>
      )
   } else {
      return (
         <View/>
      )
   }
}

return(
   <View style={styles.container}>
      <SearchableDropdown
         onTextChange={(text) => console.log(text)}
```

```
            onItemSelect={(item) => handleConditionChange(item)}
            //suggestion container style
            containerStyle={{ padding: 5 }}
            //inserted text style
            textInputStyle={{ padding: 12, borderWidth: 1, borderColor: '#ccc',
backgroundColor: '#FAFAFA'}}
            itemStyle={{ padding: 10, marginTop: 2, backgroundColor: '#F5F5F5',
borderColor: '#bbb', borderWidth: 1,}}
            itemTextStyle={{ color: '#222',}}
            //items container style you can pass maxHeight
            itemsContainerStyle={{ maxHeight: '60%'}}
            //mapping of item array (conditions)
            items={items}
            //place holder for the search input
            placeholder={placeholder}
            placeholderTextColor="#000"
            //reset textInput Value with true and false state
            resetValue={false}
            //To remove the underline from the android input
            underlineColorAndroid="transparent"
          />
          {displayConditions()}
        </View>
    )
}};

export default LearnMore;

const styles = StyleSheet.create({
    container: {
        margin: 10,
        flex: 1
    },
    content_container: {
        margin: 10,
        flex: 1,
    },
    title: {
        fontWeight: '500',
        fontSize: 20,
        marginTop: 10,
        marginBottom: 5,
    },
```

```
    text: {
      fontSize: 15,
      marginBottom: 10
    }
});
```

4.2 Admin Site

4.2.1 Components

4.2.1.1 Sidebar.jsx

```
import React from 'react'
import "./sidebar.css"
import { FaUser, FaFolder } from 'react-icons/fa'
import { BrowserRouter as Router, Routes, Route, Link } from "react-router-dom";

export default function Sidebar() {
  return (
    <div className="sidebar">
      <div className="sidebarContainer">
        <div className="sidebarMenu">
        <h1 className="sidebarTitle">HealthSpace</h1>
        <h3 className="sidebarSubTitle">Admin Panel</h3>
          <ul className="sidebarList">
            <li className="sidebarItem">
              <Link to="/" className="link">
                <FaUser className="sidebarIcon"/>Users
              </Link>
            </li>
            <li className="sidebarItem">
              <Link to="/posts" className="link">
                <FaFolder className="sidebarIcon"/> Posts
              </Link>
            </li>
          </ul>
        </div>
      </div>
    </div>
  )
}
```

4.2.1.2 Sidebar.css

```
.sidebar{
    flex: 1;
    height: 100vh;
```

```
    padding-left: 20px;
    background-color: rgb(243, 243, 249);
}

.sidebarTitle{
    color: #0C4BAE;
    font-size: 35px;
    font-weight: bold;
    font-family: 'Lobster';
    padding: 20px;
}

.sidebarSubTitle{
    color: rgb(61, 61, 61);
    font-size: 20px;
    font-weight: bold;
    padding: 0px 20px;
}

.sidebarMenu{
    margin-bottom: 10px;
}

.sidebarList{
    list-style-type: none;
    padding: 0px;
}

.sidebarItem{
    font-size: 20px;
    color: rgb(61, 61, 61);
    cursor: pointer;
    display: flex;
    align-items: center;
    padding: 8px 40px;
    border-radius: 10px;
}

.sidebarItem.active,
.sidebarItem:hover{
    background-color: rgb(237, 237, 249);
    margin-right: 30px;
}
```

```
.sidebarIcon{
   padding: 8px;
}
```

4.2.2 Pages

4.2.2.1 Users.jsx

```
import "./users.css"
/* user table */
import { DataGrid } from "@material-ui/data-grid";
import { useState, useEffect } from "react";
// firebase
import firebase from 'firebase/compat/app';
import { auth, db, firebaseConfig } from '../../config/Firebase';

export default function Users() {

  // columns
  const columns = [
    { field: 'id', headerName: 'ID', width: 250},
    { field: 'profile_picture', headerName: 'Image', width: 130, renderCell: (params)
=> {
      return (
        <div>
          <img src={params.value} alt="" className="profilePicture" />
        </div>
      )
    }},
    { field: 'name', headerName: 'Name', width: 200},
    { field: 'email', headerName: 'Email', width: 200},
    { field: 'action', headerName: 'Action ', width: 130, renderCell: (params) => {
      return (
        <div>
          <button className="deleteButton" onClick={() =>
deleteUser(params.row.id)}>Delete</button>
        </div>
      )
    }}
  ];

  // state variable to store and deal with every user
  const [users, setUsers] = useState([]);
```

```
  // get users from database
  const getUsers = () => {
    db.collection('users').get().then((snapshot) => {
      snapshot.docs.forEach((doc) => {
        setUsers(users => [...users, doc.data()]);
      });
    });
  };

  // get users on load
  useEffect(() => {
    getUsers();
  }, []);


  // function to delete user from table + firebase
  const deleteUser = (id) => {
    // remove user from the table, filter the data returning all users except the one
to be deleted
    setUsers(users.filter((user) => user.id !== id));
    // remove user from firebase
    db.collection('users').doc(id).delete();
  };


  return (
    <div className="userList">
      <DataGrid
        disableSelectionOnClick
        rows={users}
        columns={columns}
        checkboxSelection
      />
    </div>
  );
}
```

4.2.2.2 Users.css

```
 .userList{
    flex: 5;
 }
```

```
.profilePicture{
    width: 35px;
    height: 35px;
    border-radius: 50%;
    margin-top: 20px;
}

.deleteButton{
    margin-right: 10px;
    padding: 7px 10px;
    background-color: rgb(227, 58, 58);
    border: 0ch;
    color: white;
    font-size: 13px;
    cursor: pointer;
}

.deleteButton:hover{
    background-color: #39a16f;
    background-color: rgb(212, 46, 46);
}
```

4.2.2.3 Posts.jsx

```
import "./posts.css"
/* user table */
import { DataGrid } from "@material-ui/data-grid";
import { useState, useEffect } from "react";
// firebase
import firebase from 'firebase/compat/app';
import { auth, db, firebaseConfig } from '../../config/Firebase';

export default function Posts() {

  // columns
  const columns = [
    { field: 'id', headerName: 'ID', width: 200},
    { field: 'profile_picture', headerName: 'Image', width: 130, renderCell: (params)
=> {
      return (
        <div>
          <img src={params.value} alt="" className="profilePicture" />
        </div>
      )
```

```
    }},
    { field: 'name', headerName: 'Name', width: 180},
    { field: 'text', headerName: 'Post', width: 650},
    { field: 'report_count', headerName: 'Reported', width: 180},
    { field: 'action', headerName: 'Action ', width: 130, renderCell: (params) => {
      return (
        <div>
          <button className="clearButton" onClick={() =>
clearReportCount(params.row.id)}>Clear</button>
          <button className="deleteButton" onClick={() =>
deletePost(params.row.id)}>Delete</button>
        </div>
      )
    }}
  ];

  // state variable to store and deal with every user
  const [posts, setPosts] = useState([]);

  // function to get posts from database
  const getPosts = () => {
    db.collection('posts').get().then((snapshot) => {
      snapshot.docs.forEach((doc) => {
        setPosts(posts => [...posts, doc.data()]);
      });
    });
  };

  // get posts on page load
  useEffect(() => {
    getPosts();
  }, []);


  // function to delete post from table + firebase
  const deletePost = (id) => {
    // remove user from the table, filter the data returning all posts except the one
to be deleted
    setPosts(posts.filter((post) => post.id !== id));
    // remove user from firebase
    db.collection('posts').doc(id).delete();
  };
```

```
  // function to clear the report count of a post after review
  const clearReportCount = (id) => {
    db.collection('posts').doc(id).update({
      report_count: 0
    });
    // reload the page to update the table
    getPosts();
  };

  return (
    <div className="postList">
      <DataGrid
        disableSelectionOnClick
        rows={posts}
        columns={columns}
        checkboxSelection
      />
    </div>
  );
}
```

4.2.2.4 Posts.css

```
.postList{
    flex: 5;
}

.profilePicture{
    width: 35px;
    height: 35px;
    border-radius: 50%;
    margin-top: 20px;
}

.clearButton{
    margin-right: 10px;
    padding: 7px 10px;
    background-color: #3BB077;
    border: 0ch;
    color: white;
    font-size: 13px;
    cursor: pointer;
}
```

```
.clearButton:hover{
    background-color: #39a16f;
}

.deleteButton{
    margin-right: 10px;
    padding: 7px 10px;
    background-color: rgb(227, 58, 58);
    border: 0ch;
    color: white;
    font-size: 13px;
    cursor: pointer;
}

.deleteButton:hover{
    background-color: #39a16f;
    background-color: rgb(212, 46, 46);
}
```

4.2.3 Config

4.2.3.1 Firebase.js

```
import firebase from 'firebase/compat/app';
import 'firebase/compat/auth';
import 'firebase/compat/firestore';
import { getStorage } from "firebase/storage";

const firebaseConfig = {
  apiKey: "AIzaSyC1bnf2ZjNR4YYxHrXH7ftCMqquS656Fmg",
  authDomain: "healthspace-12f87.firebaseapp.com",
  projectId: "healthspace-12f87",
  storageBucket: "healthspace-12f87.appspot.com",
  //storageBucket: "gs://healthspace-12f87.appspot.com",
  messagingSenderId: "803991289685",
  appId: "1:803991289685:web:815a7960243e10e35c39dc",
  measurementId: "G-B0MYV6KJZ9"
};

const app = firebase.initializeApp(firebaseConfig);
const db = app.firestore();
const auth = firebase.auth();
// Get a reference to the storage service
const storage = getStorage(app);
```

```
export { db, auth, firebaseConfig};
```

4.2.4 Project Root

4.2.5 App.js

```
import "./app.css";
/* components */
import Sidebar from './components/sidebar/Sidebar.jsx';
/* screens */
import Users from './pages/users/Users.jsx';
import Posts from './pages/posts/Posts.jsx';
/* react router dom */
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
// firebase
import firebase from 'firebase/compat/app';
import { auth, db, firebaseConfig } from './config/Firebase.js';

function App() {
  return (
    <Router>
      <div className="container">
      <Sidebar/>
      <Routes>
        <Route path="/" element={<Users/>} />
        <Route path="/posts" element={<Posts/>} />
      </Routes>
      </div>
    </Router>
  );
}

export default App;
```

4.2.6 App.css

```
.container{
    display: flex;
    position: sticky;
    background-color: rgb(251, 251, 255);
}

.content{
    flex: 4;
    background-color: rgb(251, 251, 255);
```

```
}

.link{
    display: flex;
    align-items: center;
    text-decoration: none;
    color: inherit;
}
```

5 Usage

To begin with, it is important to download the Node.js source code or pre-built installer for your operating system. This can be done by visting the site: https://nodejs.org/en/download/

Following that, open the command prompt or terminal and change directory to a folder where you wish to store the project. This is demonstrated by figure 4 below.

```
C:\>cd HealthSpace

C:\HealthSpace>
```

Figure 4. Change Directory.

The next step is to clone the project from the GitHub repository found at https://github.com/Adamcoakley/HealthSpace. To clone the repository using HTTPS, under "Clone with HTTPS," click the copy icon. Return to your terminal and type "git clone" and then paste the URL you copied earlier. For example: git clone https://github.com/Adamcoakley/HealthSpace.git. Once that is complete, the project can be run by entering the commands visible in figures 5 and 6.

```
C:\HealthSpace>yarn add expo
```

Figure 5. Yarn add expo.

```
C:\HealthSpace>expo start
```

Figure 6. Expo start.

Following the command "expo start" a local version of the application will automatically run in the browser. To run the application on an Android or iOS simulator select "Run on Android device/emulator" or "Run on iOS simulator." However, to run the application on a physical device, the Expo app is necessary. This can be downloaded from the iOS or Google Play Store. Finally, you can scan the QR code visible within the command prompt visible in figure 7 which will load the application using Expo.

Figure 7. QR code.