

Chittr App evidence document

Readme

ChittrApp

ChittrApp is a social media app which allows users to communicate via 'chits'. An account is not required to see user's chits. However, to post chits, follow and unfollow and access location, a user can sign up for free.

Installation

Use **npm install** while inside the root directory of the ChittrApp – This installs the node modules required, *may need to be used more than once – 2/3 in my case*

```
npm install
```

Then run android via react native while in the **root directory** – *This also may need to be used a few times, 4/5 in my case*

```
react-native run-android
```

NB - *I had to restart the server after building the app, as this fixed the app just attempting/stuck to load chits. It wouldn't build the first few tries, eventually it builds.*

Usage

The application consists of 3 tabs to navigate: **Home**, **Search** and **Account**
To create an account or log in, the icon in the top left has a log in icon, press and follow steps accordingly.

Home: The home screen shows the chits of all users, scroll down to consistently load more. If logged in, this is where you can compose chits and post them for the world to see.

Search: The search screen allows a user to search for other people on the app, you can press on these users to view their profile and chits. If logged in, you can follow/unfollow the user.

Account: The account screen only shows details when a user is logged in. It allows a user to update their details and see their profile stats- follow/follower count etc.

License

Lewis Frater, copyright 2020.

How this project was managed

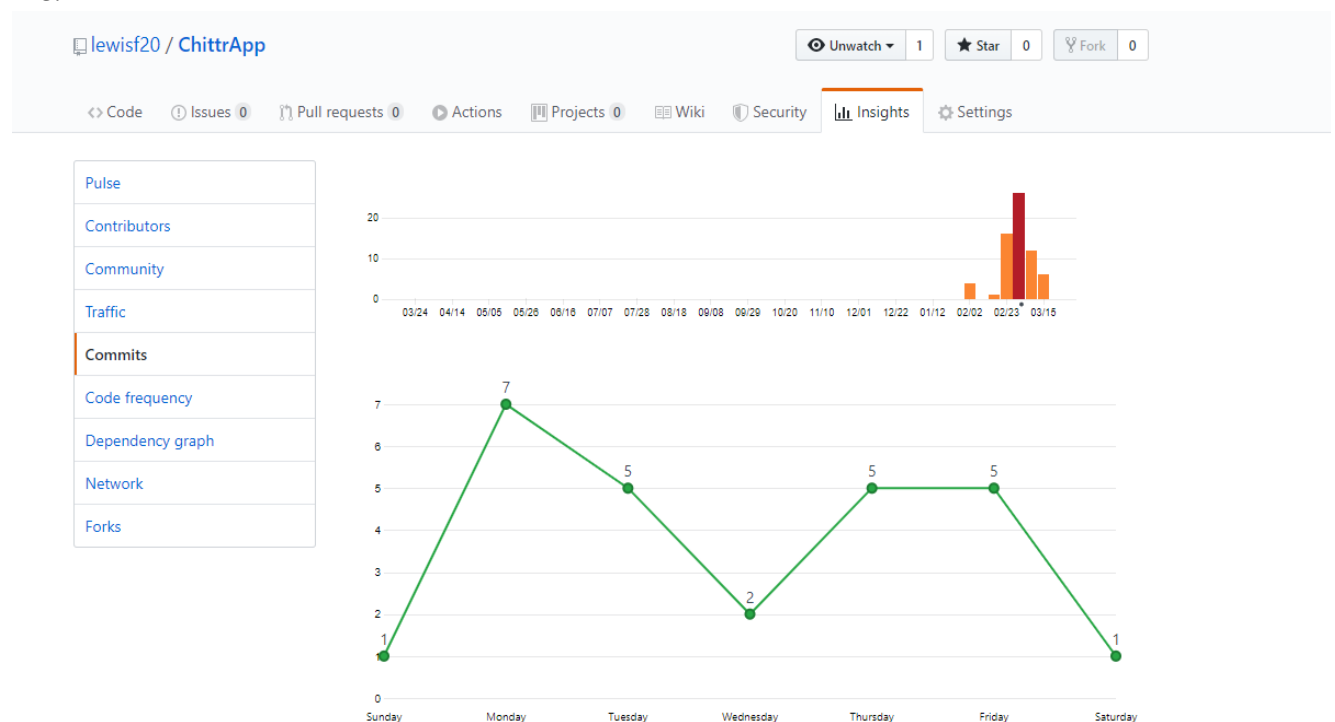
GitHub link - <https://github.com/lewisf20/ChittrApp>

This project was consistently committed to GitHub, you can see the commits and code changes in a lot of detail. It helped me keep track of progress, and noted my successes in areas of the app, as well as what I failed to implement- photo end points and extension 2.

I aimed to work on the project most days, but for no longer than 3 hours in a row (Except the odd occasion when you're on a roll!), as I believe the attention span begins to falter after too many hours. This is evident in the commits of my GitHub project.

Link - <https://github.com/lewisf20/ChittrApp/graphs/commit-activity>

The figure below shows the commits on my busiest week, but more details are in the link above, where you can break down each commit, and the changes made to each file.

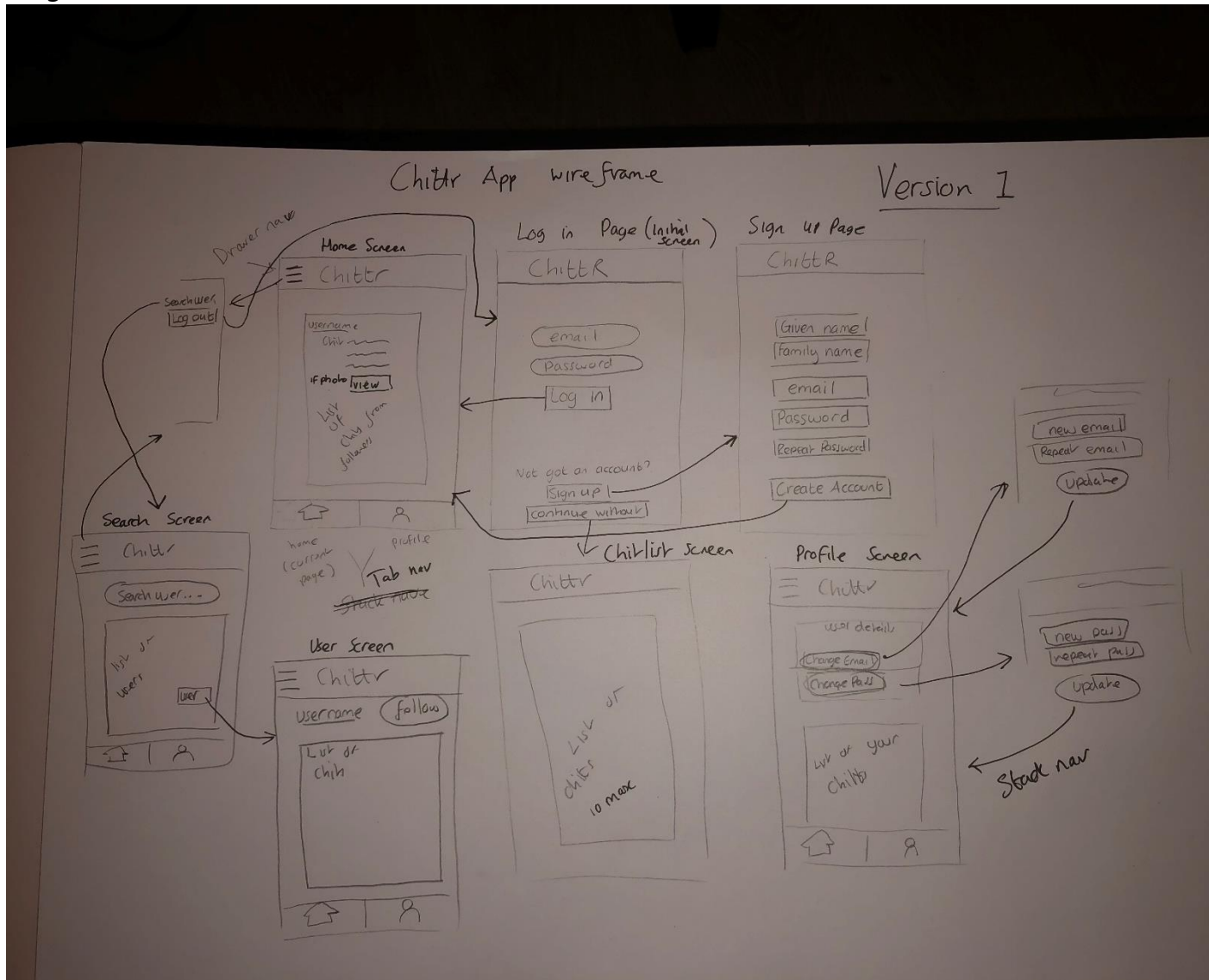


Wireframing

The design of my app was sketched on paper. I believe this is the easiest way to get ideas across quickly, pencil to paper, simple and straightforward.

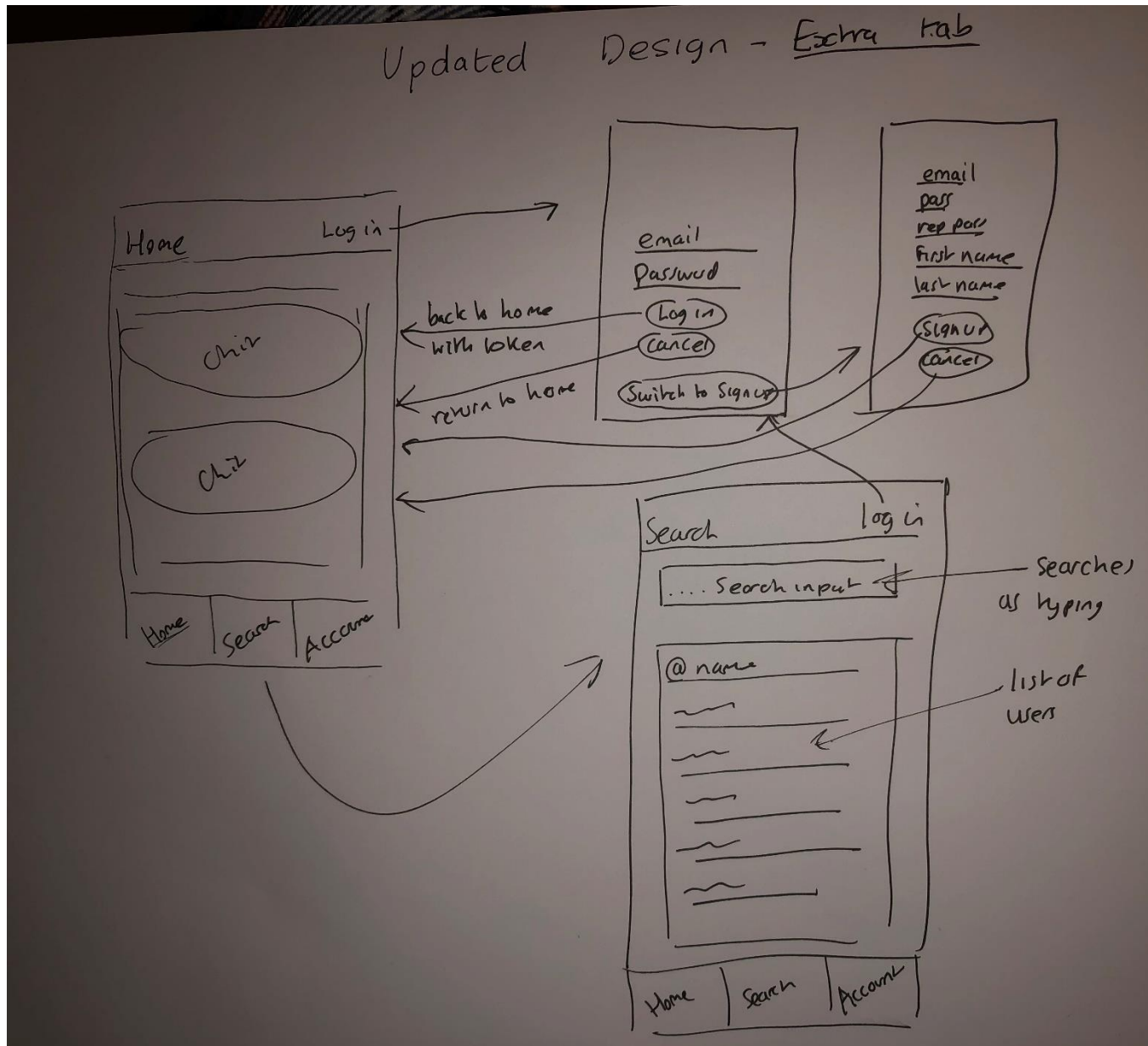
I initially had a two-tab design, however, as I went forward with the application, I quickly realised an extra tab would be beneficial, instead of having two deep stack navigators, I had 3 stack navigators where the screens are distributed evenly.

Diagram below:



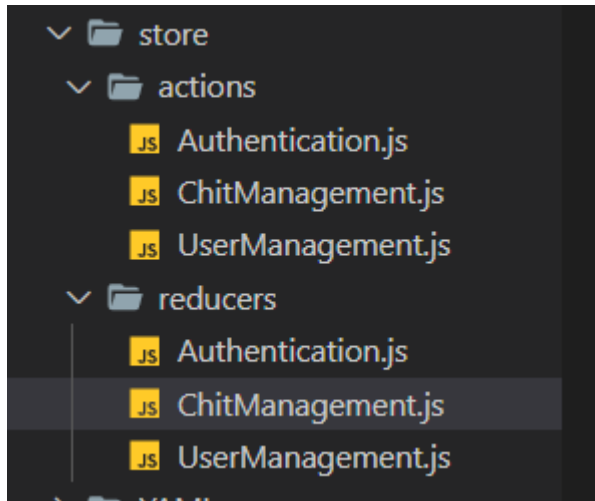
Although styling has been changed slightly within the app, this became the main navigation structure I was using, the only addition on the next diagram, is the inclusion of a search tab, to search users.

The additional tab which was added is shown below.

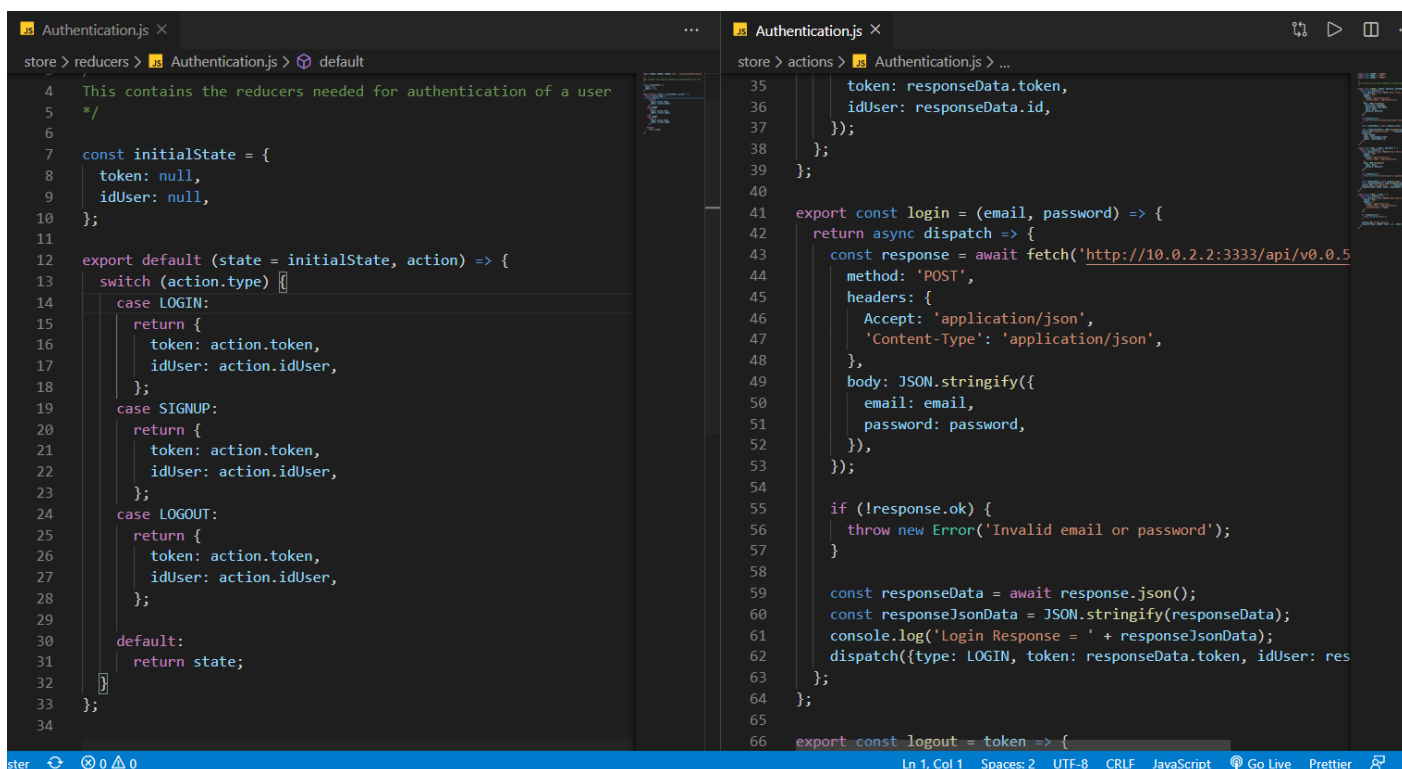


Design, structure and flow

This application was designed with the API end points in mind, with relation to **chit management, authentication and follower management**. I used Redux, which is a state container that wraps around the whole application to store important information, such as the token received when logging in. Redux helped keep all my code which contacts the server in one place making it much easier to manage.



The initial state of token and user id is null, when a login action is dispatched (*which is used in the authentication screen for this example*), the global state store is updated, this is where it holds the token. I realise there were simple alternatives like AsyncStorage to do this, but I wanted to practice and implement the Redux framework as I believe it's very valuable in the industry.

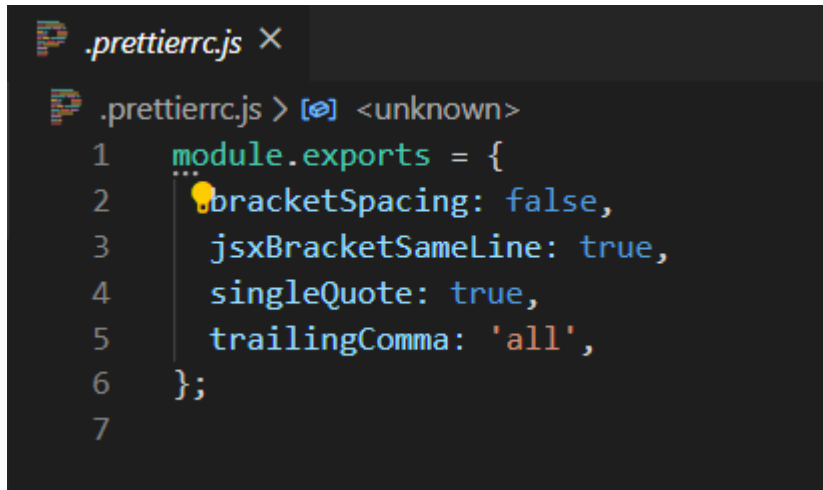


The redux store wraps around the full application in the App.js file, this enables every screen to be able to access the state and dispatch actions using this redux state store. Redux-Thunk was required to be able to dispatch async functions, this benefits functionality which contacts a server, waiting for a response. This helps with the performance of the application.

```
JS App.js  X
JS App.js > ...
6  import {Provider} from 'react-redux';
7  import ReduxThunk from 'redux-thunk';
8
9  //import reducers
10 import authenticationReducer from './store/reducers/Authentication';
11 import userManagementReducer from './store/reducers/UserManagement';
12 import chitManagementReducer from './store/reducers/ChitManagement';
13
14 import MainNavigation from './navigation/MainNavigation';
15
16 const rootReducer = combineReducers({
17   authentication: authenticationReducer,
18   userManagement: userManagementReducer,
19   chitManagement: chitManagementReducer,
20 });
21
22 //Redux store - stores all state to be used app wide
23 const store = createStore(rootReducer, applyMiddleware(ReduxThunk));
24
25 export default function App() {
26   return (
27     <Provider store={store}>
28       <MainNavigation />
29     </Provider>
30   );
31 }
```

Code quality

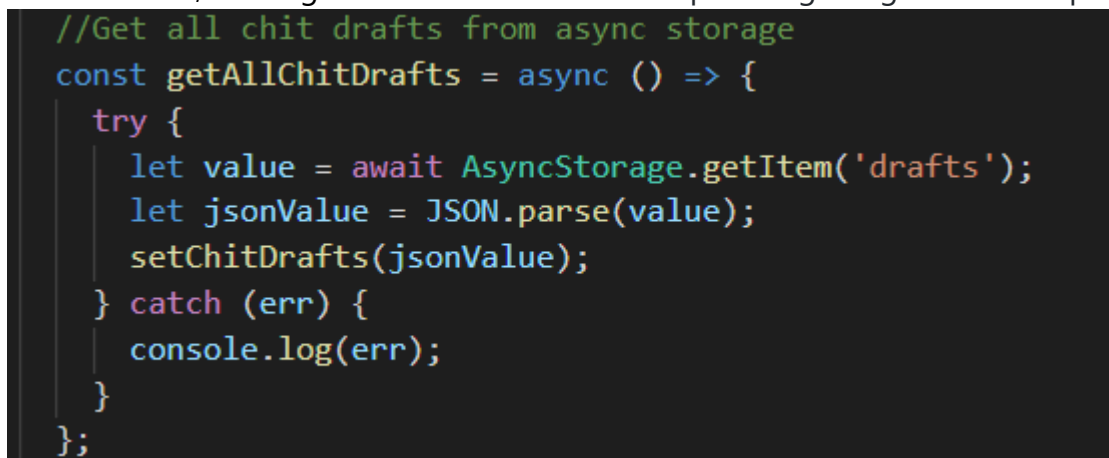
It's essential to have neat and readable code, using prettier I was able to keep my code tidy and consistent. Using single quotes in the pure JS code and double quotes in the JSX. Upon saving my work, prettier would automatically tidy my code up, making it consistent throughout, I was trying to be consistent myself, but computers are amazing and prettier helped me to keep it looking good.

A screenshot of a code editor showing a file named .prettierrc.js. The code is written in JavaScript and defines a configuration object for Prettier. The code is as follows:

```
.prettierrc.js <unknown>
1  module.exports = {
2    bracketSpacing: false,
3    jsxBracketSameLine: true,
4    singleQuote: true,
5    trailingComma: 'all',
6  };
7
```

I followed the **airbnb** styling on most of its recommendations to the best of my ability. Some examples –

No use of var, sticking to **const** and **let** to avoid polluting the global namespace.

A screenshot of a code editor showing a code snippet. The code is as follows:

```
//Get all chit drafts from async storage
const getAllChitDrafts = async () => {
  try {
    let value = await AsyncStorage.getItem('drafts');
    let jsonValue = JSON.parse(value);
    setChitDrafts(jsonValue);
  } catch (err) {
    console.log(err);
  }
};
```

Also using the array spreader to combine arrays/objects as instructed in the airbnb documentation.

```
const ChitItem = props => {
  return (
    <Card style={{...styles.container, ...props.style}}>
      <View style={styles.contentInfo}>
        <Text style={styles.name}>{props.username} chittr'd:</Text>
      </View>
      <View style={styles.content}>
        <Text style={{...styles.text, ...props.text}}>
          {props.item.chit_content}
        </Text>
      </View>
      <Text>{new Date(props.item.timestamp).toString()}</Text>
    </Card>
  );
};
```

I have also followed the airbnb styling on a number of other things such as: arrow functions for anonymous functions, multiline block statements, comments and much more.

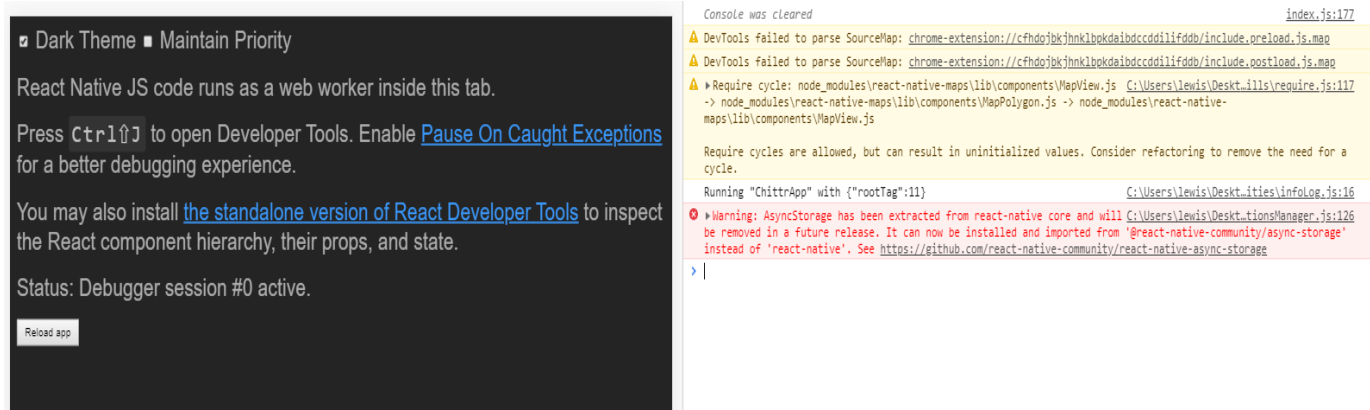
```
72 //Updates feed when user clicks on home tab - when home screen
73 //is in focus
74 useEffect(() => {
75   const onFocusListener = props.navigation.addListener(
76     'didFocus',
77     payload => {
78       chitHandler();
79     },
80   );
81   return () => {
82     onFocusListener.remove();
83   };
84 });
85
```


Testing, debugging and validation

I incorporated validation checks on many important functions when it comes to the user. For example, password checking when signing up -

```
<Btn
  style={styles.cardButton}
  title="Sign Up"
  onPress={() => {
    //Check if passwords match
    if (password !== repeatPassword) {
      Alert.alert('Passwords do not match!');
      //Clear password input fields by resetting state
      setPassword('');
      setRepeatPassword('');
    } else {
      authHandler();
    }
  }}
/>
```

Much of the debugging when things went wrong was solved using `console.log`, and the built-in **android debugging tool**. Its incredibly useful when making sure certain variables are containing the required information. Most of these debugging statements are commented out when I solved the problem I was having.



Dark Theme ■ Maintain Priority

React Native JS code runs as a web worker inside this tab.

Press `Ctrl+J` to open Developer Tools. Enable [Pause On Caught Exceptions](#) for a better debugging experience.

You may also install [the standalone version of React Developer Tools](#) to inspect the React component hierarchy, their props, and state.

Status: Debugger session #0 active.

Reload app

Console was cleared index.js:177

- ⚠ DevTools failed to parse SourceMap: [chrome-extension://cfhdojfkjhnkibokdaibdcddilifddb/include_reload.js.map](#)
- ⚠ DevTools failed to parse SourceMap: [chrome-extension://cfhdojfkjhnkibokdaibdcddilifddb/include_reload.js.map](#)
- ⚠ Require cycle: node_modules/react-native-maps/lib/components/MapView.js [C:\Users\lewis\Desktop\itiles\infoLog.js:16](#)
-> node_modules/react-native-maps/lib/components/MapPolygon.js -> node_modules/react-native-maps/lib/components/MapView.js

Require cycles are allowed, but can result in uninitialized values. Consider refactoring to remove the need for a cycle.

Running "ChittrApp" with {"rootTag":11}

⚠ Warning: AsyncStorage has been extracted from react-native core and will be removed in a future release. It can now be installed and imported from 'react-native-community/async-storage' instead of 'react-native'. See <https://github.com/react-native-community/react-native-async-storage>

Conclusion/End Notes

I immensely enjoyed this assignment, I failed to do the photo end points and extension task 2, but I really enjoyed the challenge, I really felt I was making an app for a client. It will help a lot with my confidence and understanding, as well as allowing to assess where I need to improve my own skills.

