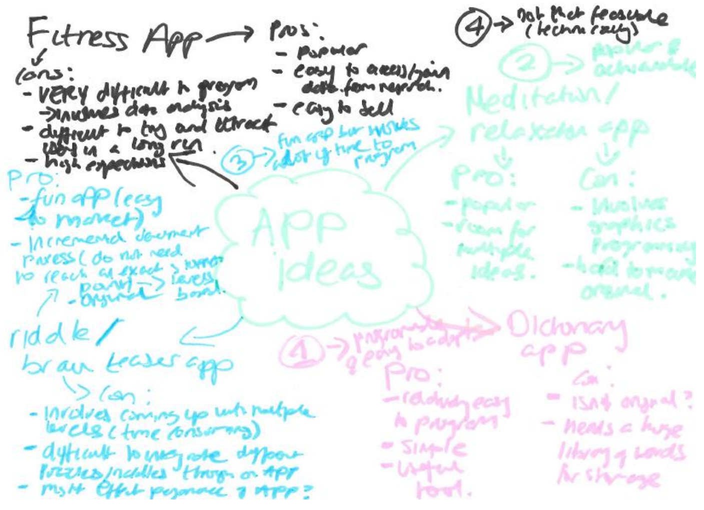
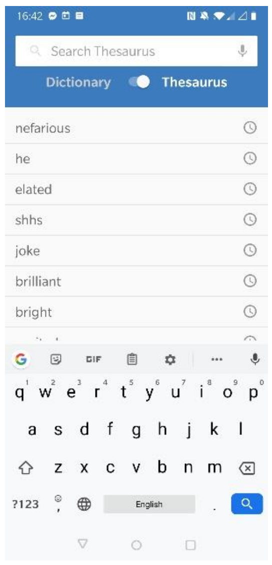
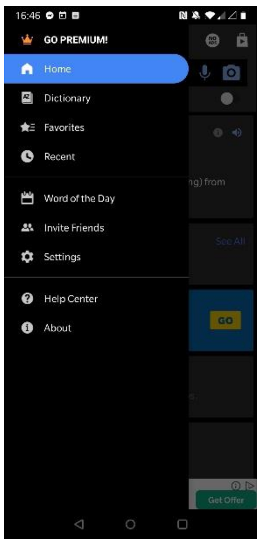
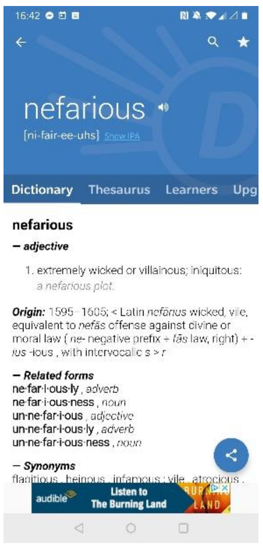
Design Log

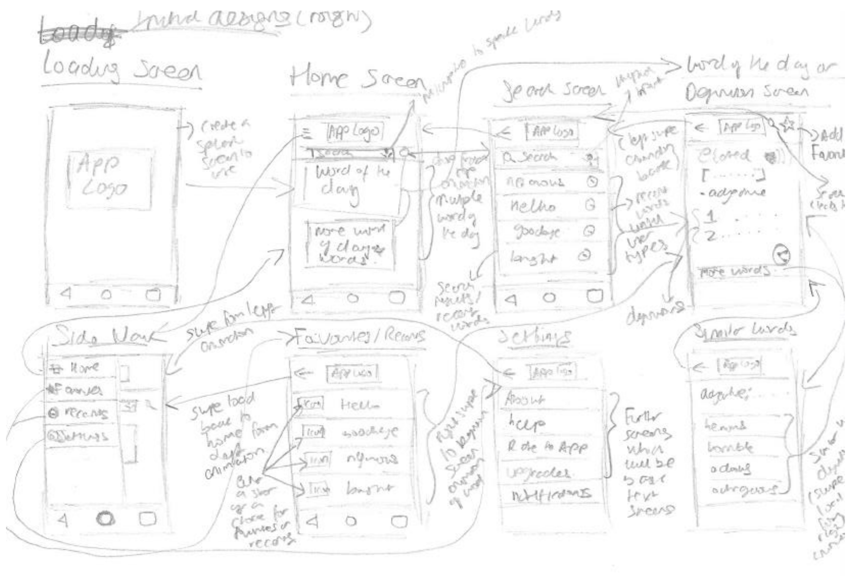
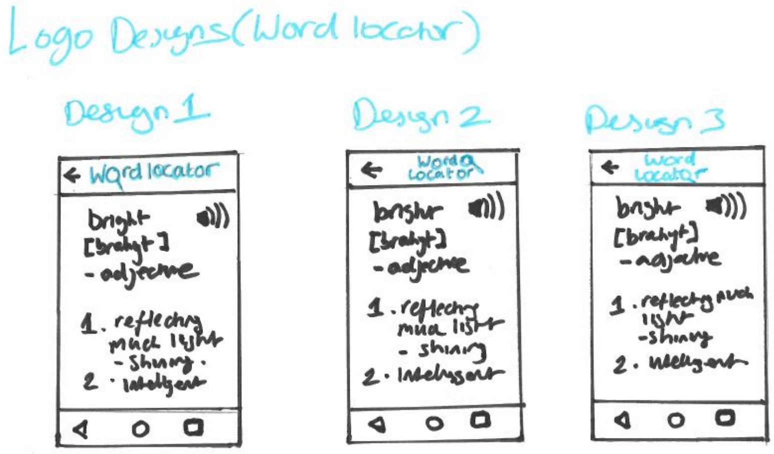
Date	Time	Title	Notes
28/01/20	20:00 – 21:00	Generating App Ideas	Created a mind map for brainstorming 4 app ideas and concepts  App Ideas Fitness App Pros: - popular - easy to access/gain data from report. - easy to sell Cons: - VERY difficult to program → involves data analysis - difficult to try and attract users in a long run. - high expectations ④ → not that feasible (technically) Meditation/ relaxation app ② → popular & achievable Pro: - popular - room for multiple ideas. Con: - involves graphics programming - hard to make original. riddle/ brain teaser app ③ → fun app but involves a lot of time to program Pro: - fun app (easy to market) - incremental development process (do not need to reach an exact point → more levels, bonus) - original Con: - involves coming up with multiple levels (time consuming) - difficult to integrate different puzzles/riddles through an app - might affect performance of app? Dictionary app ① → programming & easy to code Pro: - relatively easy to program - simple - useful tool. Con: - isn't original? - needs a huge library of words for storage Made pro's and con lists for each idea and thought about how each idea could be different to existing apps, then ranked the ideas from best concept to worst concept, in order to choose the best app idea.
28/01/20	21:00 – 22:00	Deciding on programming language/ tools and platform for development (e.g. Android vs IOS)	Listed the different programming tools (paired with which programming language) and database storage providers I would consider using to develop the app, creating pro's and con's lists for each of them. Comparing and deciding that I would use React Native and Firebase.

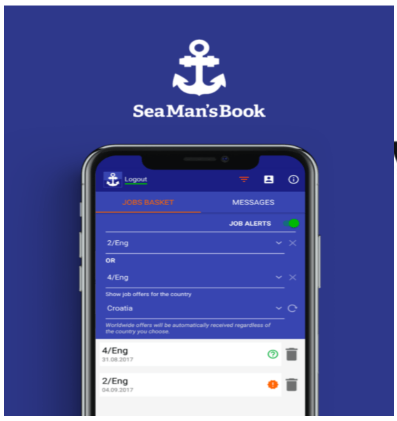
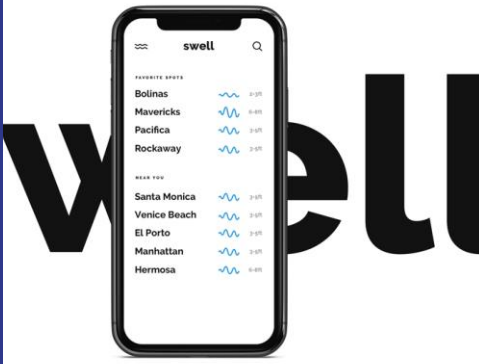
React Native (JavaScript)	**Java (Using Android Studio)**	**Swift (Objective C)**
Pros: • Many open source UI component libraries available • Can create dynamic web pages • Specifically designed to cater to mobile development • Can be cross-platform • Fast performance • Ability to preview App Cons: • Difficult to learn to program • View-orientedness due to 'Model' and 'Controller' is constricting	Pros: • Familiar IDE and programming language, therefore, may be quicker to program • Plenty of docs • Hardware independent • Ability to preview App Cons: • Slow to compile and run • From previous experience can be very buggy using libraries	Pros: • Fast performance • Open source and has docs • Promotes safe and consistent code • Easier to read/maintain • Scalable • Cross- device support • Manages memory automatically Cons: • Do not have an apple device therefore will need to use an emulate • Not overly familiar using C, Objective C or C++

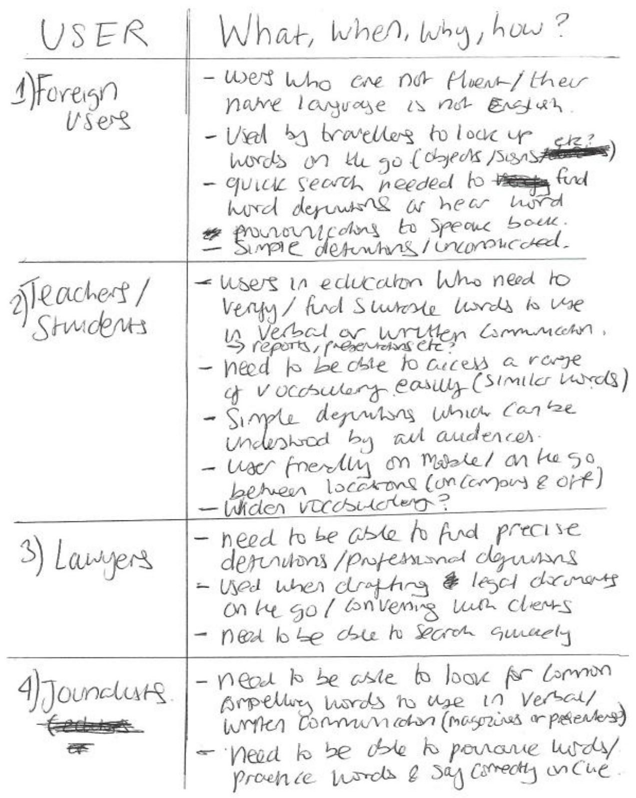
Firebase	**MySQL**	**MongoDB**
Pros: • Free (up to a high limit) • NoSQL database (better performance/is faster and easier to scale) • Google Analytics (for crash and performance monitoring)	Pros: • Familiar to querying relational data • Quick management • Able to integreate with different development tools (e.g. Android Studio)	Pros: • Flexible Database (schema-less database) • Auto-partioning into smaller databases if necessary (self-managed)

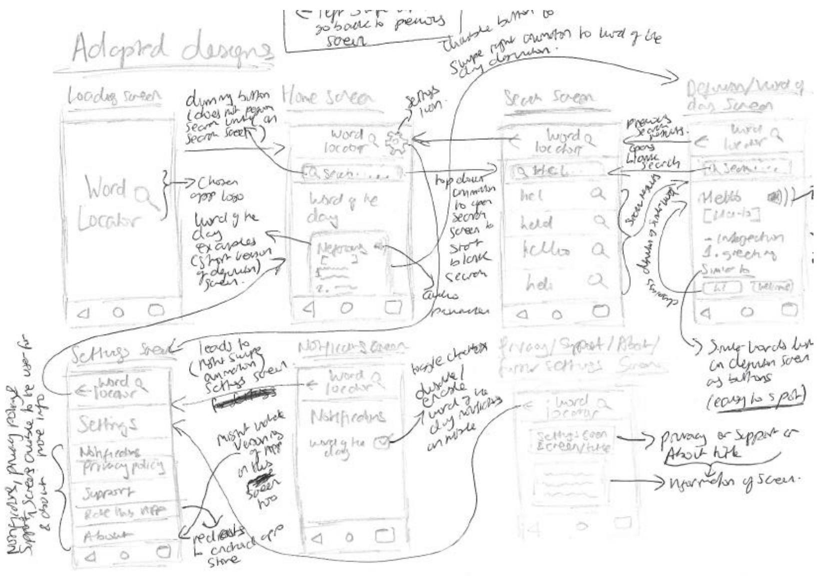
			• Database and authentication is easy to use and start up from out of the box Cons: • Uses JSON storage formatting which may make it more difficult to query for specific items • Costly past a certain limit (can only setup one RT DB in a project)	Cons: • Hard to find support for issues • Slightly buggy • Overall, slightly sluggish performance	• High availability/ scalability • Free (for non-commercial use) • Cons: • Difficult to use and program • High memory usage
29/01/20	13:00 -14:00	Word Locator Idea brainstorm	This was mainly putting myself in the seat of what the users would possibly want in the app, listing the key features of competitor apps (dictionary.com[1] and oxford dictionary[2]). Plus coming up with the name of the app(Word locator) in relation to this.  Search Thesaurus Dictionary Thesaurus nefarious he elated shhs joke brilliant bright English  GO PREMIUM! Home Dictionary Favorites Recent Word of the Day Invite Friends Settings Help Center About GO Get Offer  nefarious [ni-fair-ee-uhs] Show IPA Dictionary Thesaurus Learners Upg nefarious – *adjective* 1. extremely wicked or villainous; iniquitous: *a nefarious plot.* ***Origin:*** 1595–1605; < Latin *nefārius* wicked, vile, equivalent to *nefās* offense against divine or moral law (*ne-* negative prefix + *fās* law, right) + *-ius* -ious , with intervocalic s > *r* ***– Related forms*** ne·far·i·ous·ly , *adverb* ne·far·i·ous·ness , *noun* un·ne·far·i·ous , *adjective* un·ne·far·i·ous·ly , *adverb* un·ne·far·i·ous·ness , *noun* ***– Synonyms*** flagitious , heinous , infamous ; vile , atrocious . audible Listen to The Burning Land Key Features of both dictionary.com and oxford dictionary: • Word definitions • Related words (thesaurus) • Search functionality • Recently searched words • Word pronunciations (audio) • Favourite words list • Word of the day + notifications • Settings/about/help • Social media sharing of words • Voice search for words		

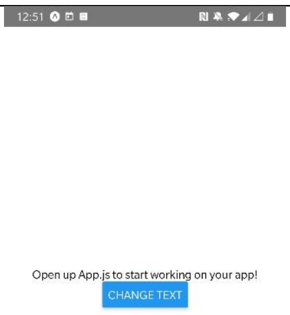
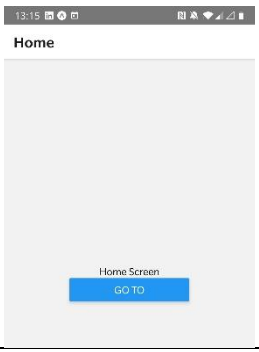
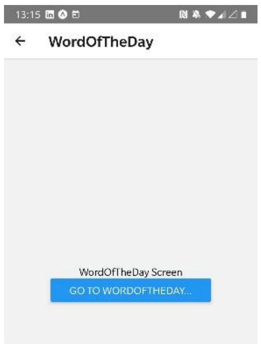
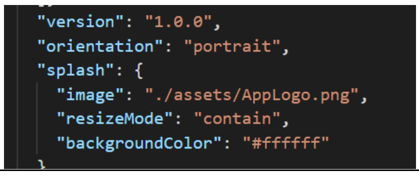
Samantha Ling, Student ID: 160720604, CSC3122 Design Log

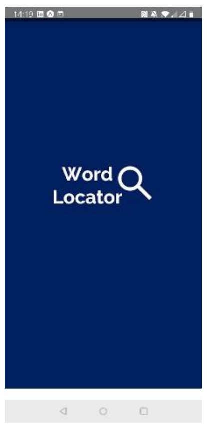
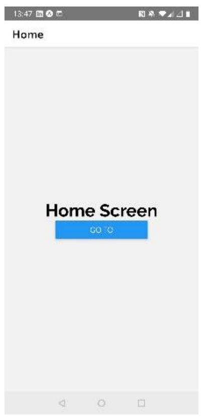
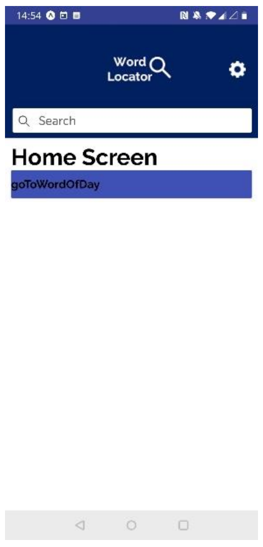
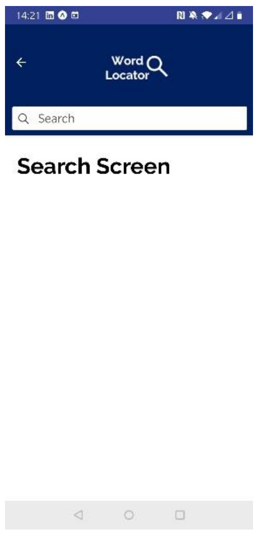
29/01/20	14:00 – 15:00	Moodboard creation	Creating a mood board based off the app idea and looking at the key features of competitor apps, web searching and choosing images and a colour scheme based off most associated images and colours to the theme/idea of the app.
01/02/20	13:00 – 14:00	Initial Sketches	Creating and annotating initial designs of the app (including logo/branding) based off moodboard and research of key features of competitor apps(I realised that the app design will most likely change in the future depending on features chosen to implement as currently its involves a heavy number of features which is not accomplishable in the time period).  Initial designs (rough) Loading Screen Home Screen Search Screen Side Nav Favourites / Recents Settings Similar Words App Logo Word of the day  Logo Designs(Word locator) Design 1 Design 2 Design 3 Word locator bright [brahyt] - adjective 1. reflecting much light - shining. 2. intelligent This also helped me understand, plan and construct a flow of which a user will be able to easily navigate through the application and the different screens.
01/02/20	14:00 – 15:00	Fontboard Creation	Started off with researching the different default font types that are available in the programming language and platform that I would be using (Android) as well as briefly looking at other existing mobile apps and the font types used.

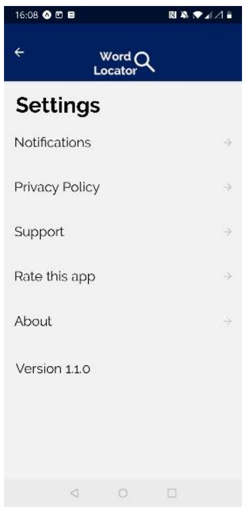
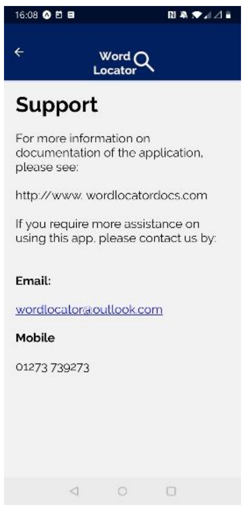
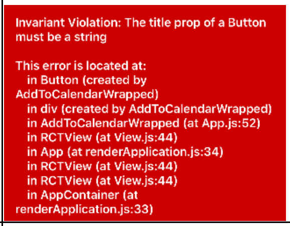
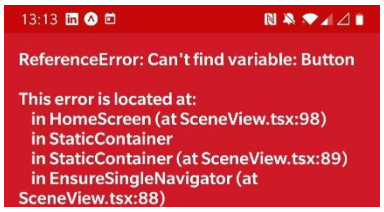
			Some of React Native fonts (Available for android): [3] [4] THIN Roboto BLACK SMALL CAPS SUNGLASSES ITALIC Self-driving robot ice cream truck BOLD Fudgesicles only 25¢ BOLD CONDENSED ICE CREAM MEDIUM Marshmallows & almonds LIGHT #9876543210 monospace monospace monospace mkdir a b c d e f g h j k l i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 a Other existing mobile app fonts: **Futura: [5]**  SeaMan'sBook Logout JOBS BASKET MESSAGES JOB ALERTS  swell Bolinas Mavericks Pacifica Rockaway Santa Monica Venice Beach El Porto Manhattan Hermosa **Raleway: [6]** Using a mobile phone (Android) template I found online, I started to construct in PowerPoint using PowerPoint tools, a basic UI screen of one of the main screens (the word definition screen) from my initial sketches and incorporated the font type using possible forms of text which might be used within the app (to envision what it might look like on a real mobile). Then picked the most appealing font-type (Raleway). From there I briefly searched how to import external fonts.
05/02/20	20:00 – 22:00	Business Case	I started to develop my business case or "pitch" for a client, at first researching different business cases to understand key points to include. Reviewed the existing apps that were similar to my app idea and analysing what user issues my app would need to overcome to improve on existing apps and therefore be worth buying, Listing my implementation strategy that I had chosen as well as taking into account other factors such as cost, timeline and other risks and contingencies of which could come into play during app development.
09/02/20	14:00 – 16:00	Personas and User Stories	I started off with first looking at what functionalities my app could provide then made a list of the most common types of possible users (mainly focusing on their occupations) the app would serve, thinking about when they would be using the app, most importantly what would they be using the app for and the reasons why as well as how (see below image).

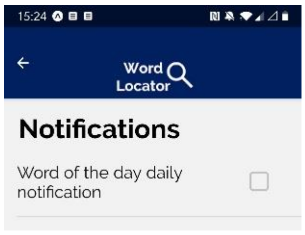
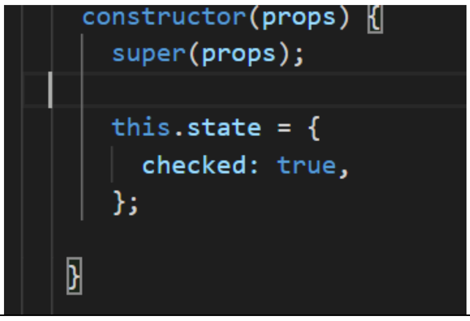
			The user stories and personas gave me insight on my own app from a different perspective, rather than just writing about my app features, I feel as though it, it helped me plan and discuss my design decisions before implementation by taking into account different user needs and prioritising app features/capabilities in order to meet the overall goals of the app. This is commonly used in agile based projects.  USER | What, when, why, how? 1) Foreign users | - users who are not fluent/their native language is not English. - Used by travellers to look up words on the go (objects/signs ~~[illegible]~~ etc?) - quick search needed to ~~[illegible]~~ find word definitions or hear word pronunciations to speak back. - Simple definitions/uncomplicated. 2) Teachers/Students | - users in education who need to verify/find suitable words to use in verbal or written communication. → reports, presentations etc? - need to be able to access a range of vocabulary easily (similar words) - Simple definitions which can be understood by all audiences. - user friendly on mobile/on the go between locations (on campus & off) - Widen vocabulary? 3) Lawyers | - need to be able to find precise definitions/professional definitions - Used when drafting ~~[illegible]~~ legal documents on the go/conversing with clients - need to be able to search quickly 4) Journalists ~~[illegible]~~ | - need to be able to look for common compelling words to use in verbal/written communication (magazines or presenters) - need to be able to pronounce words/practice words & say correctly on cue.
10/02/20	14:00 – 16:00	User stories (continued)	Adding more user stories (in order to have a range to discuss and make design decisions from)
10/02/20	20:00-21:00	Choosing app key features and re-edit of business case and Font-board	From the personas and user stories, I was able to understand the key features that my app would need to include and how it would benefit from a business point of view, deciding on the following 5 key features to implement (due to time limitations): • Easy access and simple word definitions (in an easy to read format – not overcrowded with information) • Easy access to similar words (on the word definition page) • Search functionality • Sound/audio pronunciations of words • Word of the day Editing business case and font-board to reflect the key features chosen.

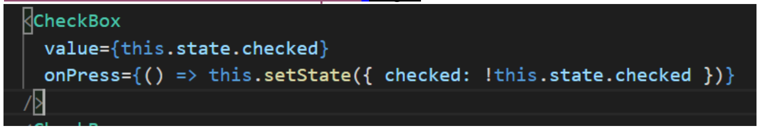
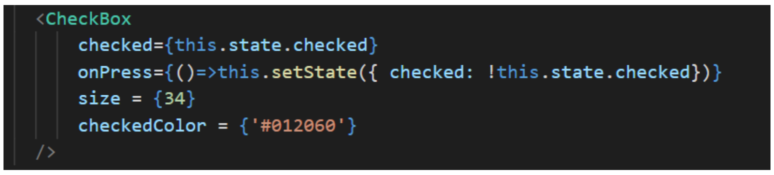
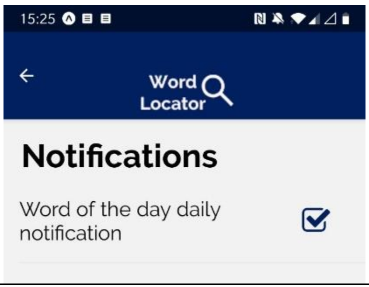
11/02/20	11:00 – 17:00	Storyboard (application diagram and individual screens)	Developing from my initial sketches and after narrowing down the features of my application I again created a mock-up of all the UI screens through sketches and then when I was happy with it transferred it digitally onto a mobile outline using PowerPoint tools (like the Footboard).  Adopted designs Loading screen Home screen Search screen Definition/word of day screen Word Locator Settings screen Notifications screen Privacy/Support/About/more settings screen I restructured the screens based on the flow I had decided within the above sketches in order to create the high-level application diagram of the storyboard. This will help to be able to reference and possibly change if necessary, the different screens during development.
11/02/20	20:00 – 23:00	Storyboard (further development to individual screens)	I delved deeper looking at the user interactions on each screen (referring back to competitor applications), the different components that are visible on the screen and how the user will get from screen to screen, plus the transitions that will need to be planned/integrated. Noting down the design and descriptions of each screen on the storyboard. This again will help during development as a reference.
12/02/20	13:00 – 15:00	Getting started on using React Native and Expo	Had a brief look at react native basics (seeing how it differs from react which I have had previous experience in). https://reactnative.dev/docs/getting-started [7]. Figuring out the best way to get stuck into creating a react native app, coming past expo; an open source platform which allows easy setup and preview of react native applications. Learning to create, build and deploy a blank react native application/project using Expo through CLI. Installing all the necessary dependencies and packages into the project. https://expo.io/ [8] basic expo managed project created and previewed on my phone through QR code scanning:

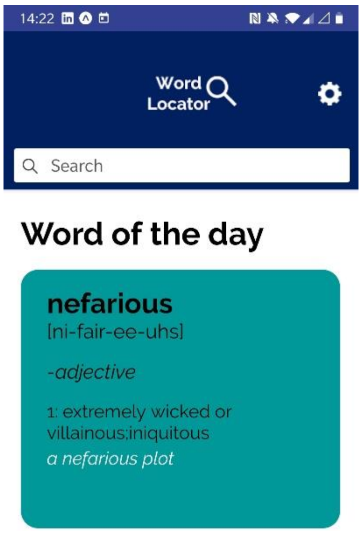
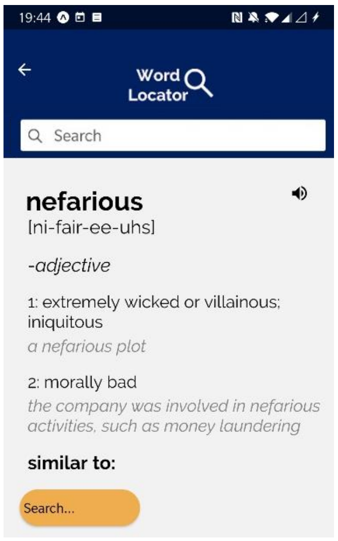
			
13/02/20	20:00 – 22:00	Understanding React Navigation using react native packages	After creating a few basic screens in react native using expo, I started looking into / searching on how to implement a basic navigator to navigate between screens, coming past react navigation, which uses a basic stack navigator. React Navigation Official Tutorial Website: https://reactnavigation.org/docs/en/getting-started.html [9]
14/02/20	13:00 – 16:00	Understanding React Navigation using react native packages (Continued)	Continued using react navigation to successfully be able to transition between screens using buttons (on the app). Creating a basic (without styling) home screen, word of the day screen, search screen as well as settings' screens   WordOfTheDay WordOfTheDay Screen GO TO WORDOFTHEDAY...
16/02/20	13:00 – 16:00	Styling loading screen(expo) + importing fonts	Worked on styling the default loading screen created by Expo, by creating a separate image with the chosen app logo (created using GIMP) to replace the default image and editing the app.JSON script (tutorial here: https://medium.com/faun/customize-your-splash-screen-with-expo-for-your-react-native-app-5aca4fc667). [10]  `"version": "1.0.0",` `"orientation": "portrait",` `"splash": {` `"image": "./assets/AppLogo.png",` `"resizeMode": "contain",` `"backgroundColor": "#ffffff"`

			  In addition to this, imported external fonts (Raleway – see above screenshot on home screen) into the project, using the following tutorial: https://medium.com/@mehran.khan/ultimate-guide-to-use-custom-fonts-in-react-native-77fcdf859cf4 [11]
20/02/20	13:00 – 19:00	Styling home screen, search screen, word of the day screen and Settings' screens	Installing and Importing UI libraries like Native base (https://docs.nativebase.io/docs/GetStarted.html [12]) and React Native Elements (https://react-native-elements.github.io/react-native-elements/docs/getting_started.html[13]) into the project, looking at implementing + styling various components, starting off with changing the header styling of the stack navigator, using the following link as a guide to the parameter values that were available: https://reactnavigation.org/docs/headers [14]   Once a header was created for all screens, text and buttons was also added to the settings screens to mimic the storyboard created.

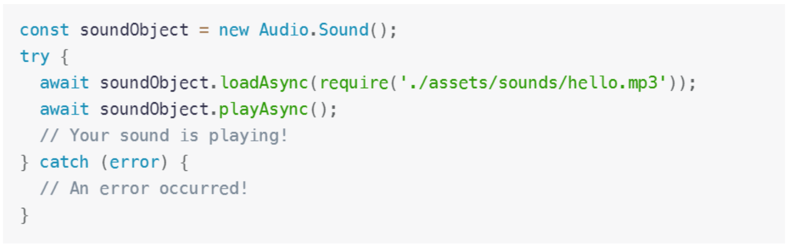
			 16:08
			Word Locator
			Settings
			Notifications
			Privacy Policy
			Support
			Rate this app
			About
			Version 1.1.0
			 16:08
			Word Locator
			Support
			For more information on documentation of the application, please see:
			http://www. wordlocatordocs.com
			If you require more assistance on using this app, please contact us by:
			Email:
			wordlocator@outlook.com
			Mobile
			01273 739273
			
			This was a lot of fiddling around with sizing and padding of various components which ate up a lot of time. also encountered some minor issues occasionally with the styling of components from different libraries which looking at the error logs , documentation and a few search links was able to quickly solve.
			
			 `Invariant Violation: The title prop of a Button must be a string`
			`This error is located at:`
			`in Button (created by AddToCalendarWrapped)`
			`in div (created by AddToCalendarWrapped)`
			`in AddToCalendarWrapped (at App.js:52)`
			`in RCTView (at View.js:44)`
			`in App (at renderApplication.js:34)`
			`in RCTView (at View.js:44)`
			`in RCTView (at View.js:44)`
			`in AppContainer (at renderApplication.js:33)`
			 13:13
			`ReferenceError: Can't find variable: Button`
			`This error is located at:`
			`in HomeScreen (at SceneView.tsx:98)`
			`in StaticContainer`
			`in StaticContainer (at SceneView.tsx:89)`
			`in EnsureSingleNavigator (at SceneView.tsx:88)`

22/02/20	13:00 – 16:00	Fixing Notifications Screen issue	Originally setup the UI Checkbox component (from react native elements) on the notifications screen previously, however, did not implement the checkbox till today.
			 15:24
			Word Locator
			Notifications
			Word of the day daily notification ☐
			
			I first manually set the state of the checkbox (using the checked property) in react to false, then true, to test if the checkbox state did change and this passed.
			 `constructor(props) {`
			`  super(props);`
			
			`  this.state = {`
			`    checked: true,`
			`  };`
			
			`}`

			However, upon setting the checkbox to change state when pressed (following this link: https://stackoverflow.com/questions/47362123/is-there-a-react-native-checkbox-example [15]), did not do so when run.
			
			`<CheckBox`
			`  value={this.state.checked}`
			`  onPress={() => this.setState({ checked: !this.state.checked })}`
			`/>`
			
			Reviewed the comments on the above example of which did not solve the issue. Ended up searching for the implementation of a react native elements checkbox and realised that the checkbox component only accepted certain parameters: https://snack.expo.io/@deivanai/react-native-element-checkbox [16]
			
			
			`<CheckBox`
			`    checked={this.state.checked}`
			`    onPress={()=>this.setState({ checked: !this.state.checked})}`
			`    size = {34}`
			`    checkedColor = {'#012060'}`
			`/>`
			
			Of which fixed the issue.
			
			
			 15:25
			Word Locator
			Notifications
			Word of the day daily notification

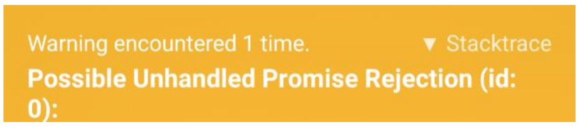
23/02/20	20:00 – 23:00	Creating a mockup of the word of the day screen (this will be removed later when the definition screen is added)	Implemented the most important UI parts of the app first, so that edits can be made later as the app develops. Editing the home screen to add the word of the day, using the storyboard as inspiration (creating a large button with the brief text definition (not yet fetching the data from backend)), that navigates to a word of the day screen (this will later be the general word definition screen)).  14:22 Word Locator Search Word of the day nefarious [ni-fair-ee-uhs] -adjective 1: extremely wicked or villainous;iniquitous a nefarious plot  19:44 Word Locator Search nefarious [ni-fair-ee-uhs] -adjective 1: extremely wicked or villainous; iniquitous a nefarious plot 2: morally bad the company was involved in nefarious activities, such as money laundering similar to: Search...

			Text had been added to the word of the day screen as a mock-up, however, will need to be generalised once the definition screen is added and backend (firebase) is incorporated.
28/02/20	13:00 – 18:00	Adding audio using expo-av	Continuing with the word of the day screen mockup, I made a start to adding audio to the screen so that when the user clicks on the sound icon, the word pronunciation would be played in English. Creating a sandbox expo project to test this in first before adding to the app later. First located a website of which allows open sourcing downloading of mp3 audio pronunciation of various English words. https://www.techvigil.com/tips-tricks/347/mp3-pronunciation-files/ [17] Downloaded the audio pronunciation of the chosen word ('nefarious') used in the word of the day mockup screen and imported this into the react native project. I was able to find Expo documentation on how to play audio in Expo projects so installed the expo dependencies needed. https://docs.expo.io/versions/latest/sdk/audio/ [18] And attempted to follow the documentation on loading and playing a sound object.



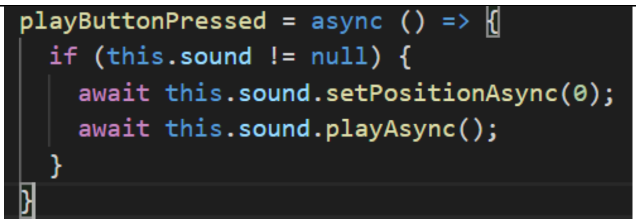
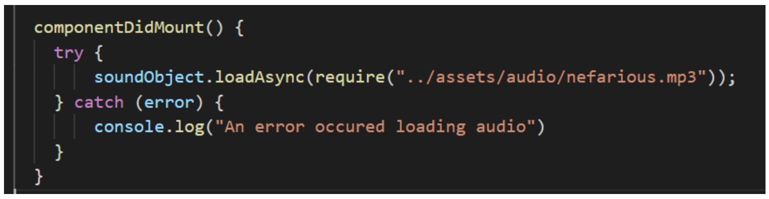
```
const soundObject = new Audio.Sound();
try {
  await soundObject.loadAsync(require('./assets/sounds/hello.mp3'));
  await soundObject.playAsync();
  // Your sound is playing!
} catch (error) {
  // An error occurred!
}
```

However, had an unhandled promise rejection issue when I clicked on the sound icon/button and no sound was produced due to the sound object not being loaded:

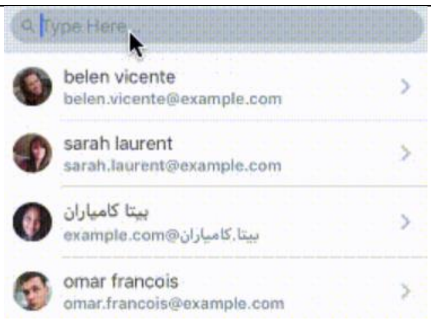


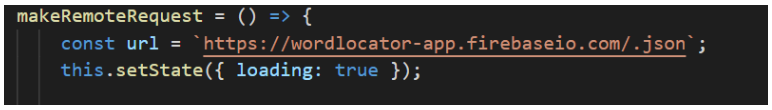
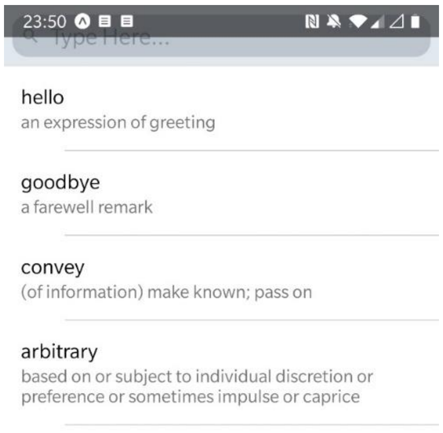
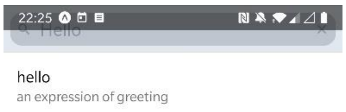
```
Warning encountered 1 time.          ▼ Stacktrace
Possible Unhandled Promise Rejection (id:
0):
```

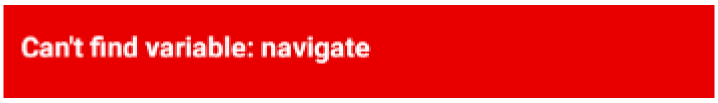
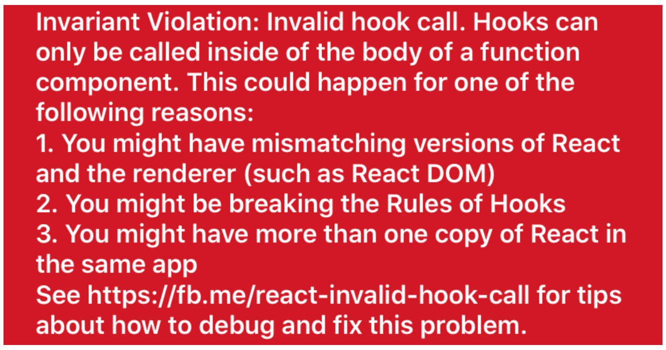
Found a related issue: https://github.com/expo/expo/issues/146 [19]
And tried different approaches of loading the audio from the comments such as setting the position of the audio and creating a separate async method to play audio.

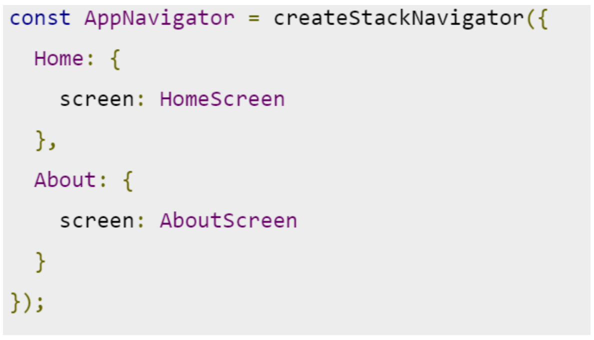
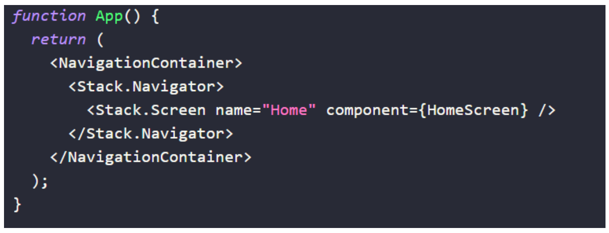
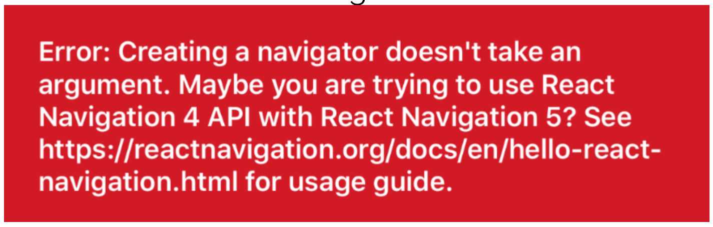
			
			`playButtonPressed = async () => {`
			`  if (this.sound != null) {`
			`    await this.sound.setPositionAsync(0);`
			`    await this.sound.playAsync();`
			`  }`
			`}`
			
			Spent time fiddling with the statements, placing the loading statement in various areas of app.js. Eventually in placing the loading statement when app.js mounts, it gave the component enough time to play the audio pronunciation of the word.
			
			
			`componentDidMount() {`
			`  try {`
			`    soundObject.loadAsync(require("../assets/audio/nefarious.mp3"));`
			`  } catch (error) {`
			`    console.log("An error occured loading audio")`
			`  }`
			`}`

01/03/20	13:00 – 17:00	Creating a search functionality	Once I had a mockup of the word of day/word definition screen created, I moved onto one of the main key functionalities of the app; implementing search to get to the word definition screen. I was aware that implementing a search functionality is incredibly hard to accomplish in any application, therefore created another separate dummy/sandbox expo project in order to edit and test search in React Native. As I had the UI component of search, I wanted to utilise that to implement the search bar. At first, I looked for guidance on where to start through a web search and found that most results were on using a Flat list and a search function of a sort to display the search results. Found the following link which not only had a simple step by step process of implementing search in a class in react but also fetched JSON data by URL(which remembering from my experience in using Firebase was how data was fetched). https://www.freecodecamp.org/news/how-to-build-a-react-native-flatlist-with-realtime-searching-ability-81ad100f6699/ [20] Was able to successfully create the example from the above link in the sandbox project. This example allowed searching for contacts in a phone book, however I will be able to use this a guideline for implementing search within my own project.

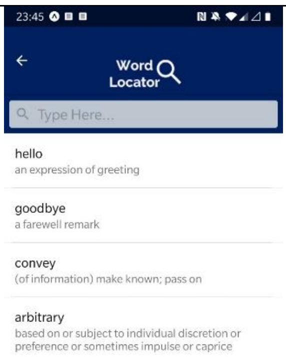
			 Type Here belen vicente belen.vicente@example.com sarah laurent sarah.laurent@example.com بيتا كامياران بيتا.كامياران@example.com omar francois omar.francois@example.com

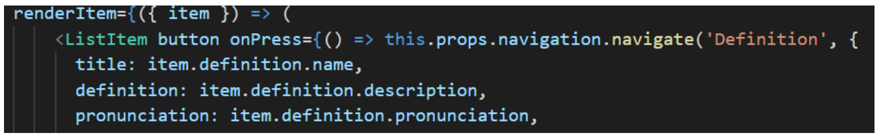
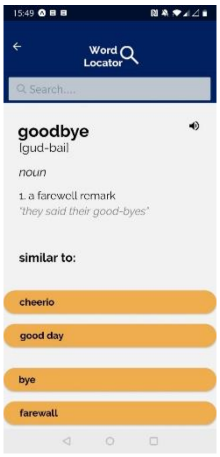
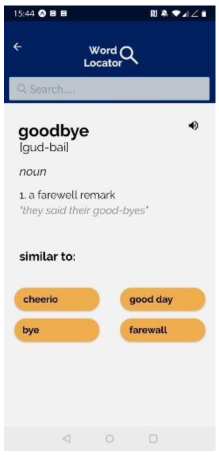
04/03/20	14:00
	–
	18:00
		Setting up Google Firebase	Continuing from creating the example contacts list in a sandbox project on 01/02/20, I will utilise this example to be able to adapt this for searching for word definitions. However, first needed to setup a new firebase project(with a real-time database) on my account for this app(I had previous experience with this, so it didn't take me long). When this was created, I then was able to import the WordNet package [21] of word definitions as a JSON script into the project. (Due to the enormity of word definitions reducing the performance of firebase and the app I unfortunately had to cut down the definitions to be just under 500).
			When this data was confirmed to be in the project (using the firebase console), I then added firebase dependencies and config to the sandbox project in order to test.
			https://docs.expo.io/versions/latest/guides/using-firebase/ [22]
			
			Swapping the URL and search parameters in the search class to fetch the firebase JSON script.
			
			
			`makeRemoteRequest = () => {`
			`    const url = ` `https://wordlocator-app.firebaseio.com/.json` `;`
			`    this.setState({ loading: true });`
			
			After refreshing, the expo project was successful in fetching and displaying the data from firebase in a Flatlist. Also, tested the functionality of the search function in the class and was overjoyed to see that it still worked.
			
			23:50
			Type Here...
			hello
			an expression of greeting
			goodbye
			a farewell remark
			convey
			(of information) make known; pass on
			arbitrary
			based on or subject to individual discretion or preference or sometimes impulse or caprice
			
			22:25
			Hello
			hello
			an expression of greeting
			
			As I was happy with the search functionality, I decided to move back to my original react native expo project to implement the search functionality again however make tweaks to the UI display.

06/03/20	13:00 – 20:00	Finding the Navigation Issue	I realised that upon returning to my original project and creating a react class for the search functionality, that I had implemented mobile screens through functions in App.js, and not classes up to this point as given by react navigation examples: https://reactnavigation.org/docs/hello-react-navigation [23] Therefore, wasn't sure on how to be able to access the react navigation prop in another class, and had to figure a way round it.  Can't find variable: navigate Tried using 'use navigation' (https://reactnavigation.org/docs/use-navigation/ [24]) to pass navigation prop to the class, but kept getting the above issue on expo during run. I attempted searching for alternatives like importing the search screen/function from App.js , trying the following link examples: • https://stackoverflow.com/questions/43648440/how-to-call-a-function-from-another-class-in-react-native [25] • https://stackoverflow.com/questions/53128479/react-native-importing-function-from-another-file [26] Then encountered a different issue:  Invariant Violation: Invalid hook call. Hooks can only be called inside of the body of a function component. This could happen for one of the following reasons: 1. You might have mismatching versions of React and the renderer (such as React DOM) 2. You might be breaking the Rules of Hooks 3. You might have more than one copy of React in the same app See https://fb.me/react-invalid-hook-call for tips about how to debug and fix this problem. And upon searching the issue, realised that I couldn't pass the navigation prop to the class as a nested function because of it being an invalid hook call in react. https://stackoverflow.com/questions/59701558/react-native-invalid-hook-call [27] Upon several hours of trying to fix the issue, I decided that I would just try and find an alternative way to navigate between screens using classes instead of functions another day.
08/03/20	13:00 – 19:00	Solving the navigation issue	Due to the inability of using the navigation prop in separate classes, I decided to find an alternative solution. Searching to see how other developers have approached this issue. Most of the search links I came across involved defining the stack navigator in the following format:

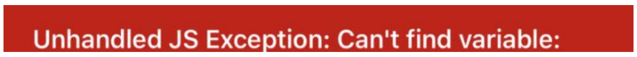
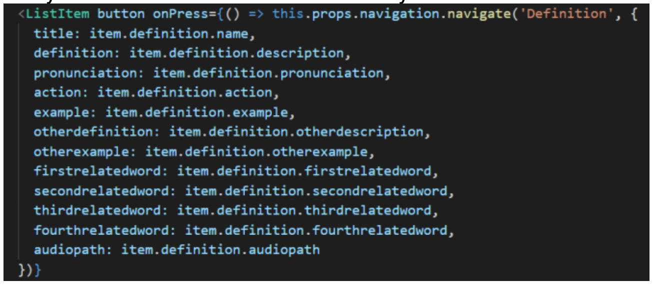
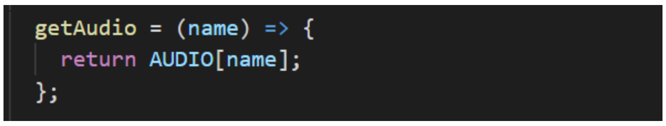
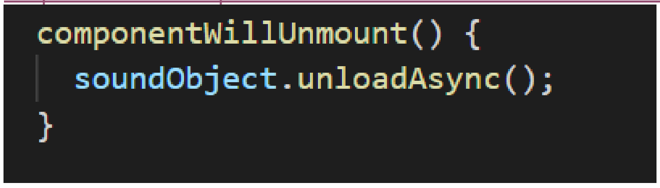
			
			`const AppNavigator = createStackNavigator({`
			`  Home: {`
			`    screen: HomeScreen`
			`  },`
			`  About: {`
			`    screen: AboutScreen`
			`  }`
			`});`
			
			Rather than how I approached it as stated on the react navigation website:
			
			`function App() {`
			`  return (`
			`    <NavigationContainer>`
			`      <Stack.Navigator>`
			`        <Stack.Screen name="Home" component={HomeScreen} />`
			`      </Stack.Navigator>`
			`    </NavigationContainer>`
			`  );`
			`}`
			
			So, I decided to test this in my project by following a basic tutorial: https://blog.logrocket.com/navigating-react-native-apps-using-react-navigation/ [28]
			
			and received the following issue:
			
			Error: Creating a navigator doesn't take an argument. Maybe you are trying to use React Navigation 4 API with React Navigation 5? See https://reactnavigation.org/docs/en/hello-react-navigation.html for usage guide.
			
			Realizing that I was trying to use a react navigation 4.0 navigator instead of 5.0 (current version which I was using).
			Unfortunately, upon searching I couldn't seem to find any alternatives to using a navigator in react to navigate different classes, so decided to try and bump down the version of the react navigation package:
			
			https://stackoverflow.com/questions/51306148/how-to-downgrade-react-native-from-0-56-0-to-0-55-4-version [29]
			
			This finally fixed the issue and I was able to navigate between classes; however, this now will be cost of now having to migrate all the separate functions/screens already implemented into new React classes.

13/03/20	14:00 – 18:00	Migrating functions to classes	This is continuing from where I left off with restructuring the navigation of my app by migrating screens from functions in app.js to classes.

14/03/20	13:00 – 16:00	Changing styling in search screen	 Word Locator Type Here... hello an expression of greeting goodbye a farewell remark convey (of information) make known; pass on arbitrary based on or subject to individual discretion or preference or sometimes impulse or caprice Once all screens were migrated from functions in app.js to separate classes, I then returned to the search screen and finished implementation of the search screen, including styling.

21/03/20	13:00 – 19:00	Creating the definition screen	After the completion of the search screen, I started to create the actual definition screen template (using the word of the day screen UI as a starting point) of which allows users to view different word definitions. Figuring out how to pass the selected word name, definition and pronounciation (from firebase) to the definition page when clicked on the search screen.
			
			Luckily react navigation had a way to pass parameters between classes:
			https://reactnavigation.org/docs/params [30]
			
			
			`renderItem={({ item }) => (`
			`<ListItem button onPress={() => this.props.navigation.navigate('Definition', {`
			`title: item.definition.name,`
			`definition: item.definition.description,`
			`pronunciation: item.definition.pronunciation,`
			Was able to test this and found it did indeed pass the selected word data to the defintion screen, so passed the rest of the data and styled the screen from there.
			 Word Locator
			Search...
			goodbye
			[gud-bail]
			noun
			1. a farewell remark
			'they said their good-byes'
			similar to:
			cheerio
			good day
			bye
			farewall Word Locator
			Search...
			goodbye
			[gud-bail]
			noun
			1. a farewell remark
			'they said their good-byes'
			similar to:
			cheerio
			good day
			bye
			farewall
			Had a few issues with button placement of similar words as I wanted to style the buttons to be side by side. After searching and trying a couple of solutions was able to find a solution that was correct:
			
			https://stackoverflow.com/questions/42864827/two-buttons-with-equal-width-horizontally-fill-the-screen-in-react-native/42881304 [31]

22/03/20	20:00 – 23:00	Fixing the word of the day (+ home screen)	After completing the search and definition screen, I was able to revisit the word of the day mockup screen/class and remove this from the project, then re-implement the word of the day functionality within the home screen class (based off the definition screen) to randomly choose a item/word in firebase and display the definition (a shorter version) on the home screen.

04/04/20	11:00 – 18:00	Fixing the audio on different word definitions.	Upon starting the app, I doubled checked the search functionality on various words and it seemed to display correctly, however upon clicking the sound icon/button came up with the following issue on the path of the audio:  Unhandled JS Exception: Can't find variable:
			
			The audio path was passed as a parameter to the definition screen similar to the other parameters, and was programmed in a similar way as on the word of the day screen on the defintion screen.  `<ListItem button onPress={() => this.props.navigation.navigate('Definition', {`
			`  title: item.definition.name,`
			`  definition: item.definition.description,`
			`  pronunciation: item.definition.pronunciation,`
			`  action: item.definition.action,`
			`  example: item.definition.example,`
			`  otherdefinition: item.definition.otherdescription,`
			`  otherexample: item.definition.otherexample,`
			`  firstrelatedword: item.definition.firstrelatedword,`
			`  secondrelatedword: item.definition.secondrelatedword,`
			`  thirdrelatedword: item.definition.thirdrelatedword,`
			`  fourthrelatedword: item.definition.fourthrelatedword,`
			`  audiopath: item.definition.audiopath`
			`})}`
			The only difference I spotted was that the path was being passed as a parameter to the class instead of being hard coded, so tried a hard coded path and this fixed the issue. I didn't understand why this was an issue so searched for the cause:
			https://github.com/expo/expo/issues/4332 [32]
			
			and found out that I couldn't dynamically pass the audio path using require(''). So manually moved the audio paths from firebase into the project as constants in constants.js and referenced this in the definition screen class.  `getAudio = (name) => {`
			`  return AUDIO[name];`
			`};`
			After fixing the above issue, I tested that the audio was working when visiting different words, however for some reason it was replaying the word pronunciation of the first word pronunciation clicked on when the app started. I restarted the app and it still had the same issue.
			
			Looking at the documentation, I was able to find out why very easily and understood that I didn't unload the audio when leaving the screen of a word definition (i.e. unmounting), so it kept replaying.
			
			https://docs.expo.io/versions/latest/sdk/audio/ [18]  `componentWillUnmount() {`
			`  soundObject.unloadAsync();`
			`}`

References

[1] Dictionary.com: Find Definitions for English Words. Dictionary.com. (2020). Available: https://play.google.com/store/apps/details?id=com.dictionary&hl=en_GB. *Last Accessed: April 2020.*

[2] Oxford Dictionary of English : Free. MobiSystems. (2019). Available: https://play.google.com/store/apps/details?id=com.mobisystems.msdict.embedded.wireless.oxford.dictionaryofenglish&hl=en_GB. *Last Accessed: February 2020.*

[3] Download Roboto Font that Google Made for Android. A.Agarwal. (2012). Available: https://www.labnol.org/software/download-google-roboto-font/21007/. *Last Accessed: February 2020.*

[4] Monospace font pairings. Figma.com. (2019). Available: https://www.figma.com/font-types/monospace-fonts/. *Last Accessed: January 2020.*

[5] seamans-book portfolio. itcraftApps.com. (2019). Available: https://itcraftapps.com/portfolio/seamans-book/. *Last Accessed: April 2020.*

[6] Swell Surf App. D. Salgado. (2018). Available: https://dribbble.com/shots/5426463-Swell-Surf-App. *Last Accessed: March 2020.*

[7] Introduction. React Native 0.62. (2020). Available: https://reactnative.dev/docs/getting-started. *Last Accessed: April 2020.*

[8] Expo. (2020). Available: https://expo.io/. *Last Accessed: November 2019.*

[9] Getting started. React Navigation 5.0. (2020). Available: https://reactnavigation.org/docs/getting-started. *Last Accessed: March 2020.*

[10] Customize your splash screen with Expo for your React Native app. Y. Henni. (2019). Available: https://medium.com/faun/customize-your-splash-screen-with-expo-for-your-react-native-app-5aca4fc667. *Last Accessed: January 2019.*

[11] Ultimate guide to use custom fonts in react native. M.Khan. (2019). Available: https://medium.com/@mehran.khan/ultimate-guide-to-use-custom-fonts-in-react-native-77fcdf859cf4. *Last Accessed: January 2019.*

[12] Getting Started. Native Base v2 13.0. (2019). Available: https://docs.nativebase.io/docs/GetStarted.html. *Last Accessed: February 2019.*

[13] Getting Started. React Native Elements 1.2.0. (2020). Available: https://react-native-elements.github.io/react-native-elements/docs/getting_started.html. *Last Accessed: February 2020.*

[14] Configuring the header bar. React Navigation 5.0. (2020). Available: https://reactnavigation.org/docs/headers/. *Last Accessed: April 2020.*

[15] react-native checkbox example. K.Harrison. (2019). Available: https://stackoverflow.com/questions/47362123/is-there-a-react-native-checkbox-example. *Last Accessed: November 2019.*

[16] React Native Element Checkbox. Deiveani. (2019). Available: https://snack.expo.io/@deivanai/react-native-element-checkbox. *Last Accessed: January 2020.*

[17] Get MP3 Audio Pronunciation File of Any Dictionary Word. TechVigil. (2012). Available: https://www.techvigil.com/tips-tricks/347/mp3-pronunciation-files/. *Last Accessed. February 2013.*

[18] Audio. Expo. (2020). Available: https://docs.expo.io/versions/latest/sdk/audio/. *Last Accessed: April 2020.*

[19] Playing audio on Android error. (2017). Available: https://github.com/expo/expo/issues/146. *Last Accessed: May 2017.*

[20] How to build a React Native FlatList with realtime searching ability. freeCodeCamp. (2018). Available: https://www.freecodecamp.org/news/how-to-build-a-react-native-flatlist-with-realtime-searching-ability-81ad100f6699/. *Last Accessed: August 2018.*

[21] Downloading WordNet and associated packages and tools. WordNet. (2005). Available: https://wordnet.princeton.edu/download. *Last Accessed: December 2006.*

[22] Using Firebase. Expo. (2020). Available: https://docs.expo.io/guides/using-firebase/?redirected. *Last Accessed: January 2020.*

[23] Hello React Navigation. React Navigation 5.0. (2020). Available: https://reactnavigation.org/docs/hello-react-navigation/. *Last Accessed: April 2020.*

[24] useNavigation. React Navigation 5.0. (2020). Available: https://reactnavigation.org/docs/use-navigation/. *Last Accessed: April 2020.*

[25] How to call a function from another class in React-Native?. (2017). Available: https://stackoverflow.com/questions/43648440/how-to-call-a-function-from-another-class-in-react-native. *Last Accessed: April 2017.*

[26] React Native importing function from another file. (2018). Available: https://stackoverflow.com/questions/53128479/react-native-importing-function-from-another-file. *Last Accessed: November 2018.*

[27] React Native: Invalid Hook Call. S. Kabir. (2020). Available: https://stackoverflow.com/questions/59701558/react-native-invalid-hook-call. *Last Accessed: January 2020.*

[28] Navigating React Native apps using React Navigation. E.Yasufu. (2019). Available: https://blog.logrocket.com/navigating-react-native-apps-using-react-navigation/. *Last Accessed: June 2019.*

[29] How to downgrade react-native from 0.56.0 to 0.55.4 version?. (2018). Available: https://stackoverflow.com/questions/51306148/how-to-downgrade-react-native-from-0-56-0-to-0-55-4-version. *Last Accessed: July 2018.*

[30] Passing parameters to routes. React Navigation 5.0. (2020). Available: https://reactnavigation.org/docs/params/. *Last Accessed: April 2020.*

[31] Two buttons with equal width horizontally fill the screen in React Native. V. Jalan. (2017). Available: https://stackoverflow.com/questions/42864827/two-buttons-with-equal-width-horizontally-fill-the-screen-in-react-native/42881304. *Last Accessed: March 2017.*

[32] expo audio load multiple audio file. (2019). Available: https://github.com/expo/expo/issues/4332. *Last Accessed: May 2019.*