James Duggan

06357628

Tourpal

A Walking Tour App Final Report

Table of Contents

[Declaration 0](#_Toc101108463)

[1 Introduction 1](#_Toc101108464)

[1.1 Background to the Problem 1](#_Toc101108465)

[1.2 Description of Project Aims 1](#_Toc101108466)

[2 Technologies 2](#_Toc101108467)

[2.1 Software 2](#_Toc101108468)

[2.1.1 React 2](#_Toc101108469)

[2.1.2 React Native 2](#_Toc101108470)

[2.1.3 Expo 2](#_Toc101108471)

[2.1.4 Leaflet 2](#_Toc101108472)

[2.1.5 React Native Maps 3](#_Toc101108473)

[2.1.6 Firebase 3](#_Toc101108474)

[3 System and Analysis Design 4](#_Toc101108475)

[3.1 User Stories 4](#_Toc101108476)

[3.2 Class Diagrams 5](#_Toc101108477)

[3.3 Functional Requirements 5](#_Toc101108478)

[3.3.1 Core Requirements 5](#_Toc101108479)

[3.3.2 Non-functional Requirements 6](#_Toc101108480)

[3.3.3 Stretch Goals 6](#_Toc101108481)

[3.4 User Interface Design 6](#_Toc101108482)

[3.4.1 Web App 6](#_Toc101108483)

[3.4.2 Mobile App 7](#_Toc101108484)

[4 Methodology 8](#_Toc101108485)

[4.1 Methodology Selection 8](#_Toc101108486)

[4.2 Process Outline 8](#_Toc101108487)

[4.2.1 Phase 1: Define and finalize project requirements 8](#_Toc101108488)

[4.2.2 Phase 2: Begin building prototypes 8](#_Toc101108489)

[4.2.3 Phase 3: Gather user feedback 9](#_Toc101108490)

[4.2.4 Phase 4: Testing 9](#_Toc101108491)

[4.2.5 Phase 5: Present your system 9](#_Toc101108492)

[4.3 Project Schedule 9](#_Toc101108493)

[5 Implementation 10](#_Toc101108494)

[5.1 Web App 10](#_Toc101108495)

[5.1.1 Phase 1: App Creation and Login 10](#_Toc101108496)

[5.1.2 Phase 2: Navigation 11](#_Toc101108497)

[5.1.3 Phase 3: Tour Card and Tour Card List 12](#_Toc101108498)

[5.1.4 Phase 4: Maps 14](#_Toc101108499)

[5.1.5 Phase 4: Firebase Realtime Database 15](#_Toc101108500)

[5.1.6 Phase 5: Firebase Storage 17](#_Toc101108501)

[5.1.7 Phase 6: Media Player 18](#_Toc101108502)

[5.1.8 Phase 7: Further CRUD Implementation 20](#_Toc101108503)

[5.2 Mobile App 23](#_Toc101108504)

[5.2.1 Phase 1: App Creation on Expo, Navigation, and Firebase Auth 23](#_Toc101108505)

[5.2.2 Phase 2: Accessing Realtime Database 25](#_Toc101108506)

[5.2.3 Phase 3: Maps 26](#_Toc101108507)

[5.2.4 Phase 4: Media Player 28](#_Toc101108508)

[5.2.5 Phase 5: Geolocation 30](#_Toc101108509)

[5.2.6 Phase 6: User Model 31](#_Toc101108510)

[6 Evaluation 34](#_Toc101108511)

[7 Conclusion 35](#_Toc101108512)

[Table of Figures 36](#_Toc101108513)

[Bibliography 37](#_Toc101108514)

# Declaration

I declare that the work that follows is my own, and that any quotations from any sources (e.g. books, journals, the internet) are clearly identified as such. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is in the bibliography. I have not submitted the work presented in this report in any other course of study leading to an academic reward.

Student: Text, letter

Description automatically generated Date: 17/04/2022

# Introduction

## Background to the Problem

In the current pandemic, face to face interaction and moving in groups is being avoided as people want to feel safe. Because of this, and the simple fact that people aren’t travelling, activities like tours have taken a hit over the past two years.

## Description of Project Aims

The tourism industry is important, not just from a revenue perspective but also in terms of the learning that it can provide. For my project I want to create a walking tour app that can be used by anyone, anywhere in the world to create an audio tour that, in turn, anyone can take on their mobile phone. I think an app like mine would be of great use to people who are wary about travelling in a group, as most walking tours would be, but also to the usual hosts of walking tours themselves. Someone could create a tour and have people taking it while they are still running in-person tours. They could even use the app as a kind of sample tour and mention in the audio that if someone would be interested in taking a more in-depth tour to contact them.

My project will include the following:

* A web app where users can create tours
* Users will need to be able to create a tour with different Points of Interest (POI) or Destinations and associate an audio recording with them
* A mobile app where users can take tours
* It should be easy for users taking a tour to find one in their location, possibly through use of a search bar

# Technologies

## Software

### React

React is a JavaScript library for building user interfaces. I will be using this to create my web app which a Tour Guide user will interact with to create apps. There are several reasons why I chose React to build my app in the end. Firstly, I am more confident with JavaScript than any other programming language. Also React is one of the most popular JS libraries in use today, being used by large sites such as Facebook, Instagram, and Reddit for their front-end development needs (Patel, 2022).

React is also component driven, which will assist greatly in the type of app I will be developing (Pandit, 2021). The main data types I will be using for the app will be Places of Interest (or Destinations) and Tours. These can be displayed in a similar fashion, i.e. with a map and then some data in a text format. As such, reusable components that can work off of either data set would be ideal.

### React Native

React Native is an open source UI software framework developed by Meta Inc that can be used to develop apps for both Android and iOS. It allows the developer to create an app in JavaScript that can run on both Android and iOS.

Being able to write my code for the mobile app in JavaScript will greatly reduce my workload as a lot of my code for the web app should carry over.

### Expo

Expo is an open-source platform for making universal native apps for Android, iOS, and the web with JavaScript and React. It is essentially a bundle with tools and services that allows a developer to create a mobile app in React Native with relative ease and ensures that you do not need to work with native Android or iOS code to get you project up and running (Ravichandran, 2020).

For someone just starting with React Native, Expo is a natural choice. As well as this, Expo has a mobile app that can allows the developer to view their app on their mobile device during development by scanning a QR code.

### Leaflet

Leaflet is an open source JavaScript library for creating interactive maps in web apps. I have decided to use this for my web app to keep prospective costs low.

### React Native Maps

React Native Maps is a library for React Native that assists in creating interactive maps for mobile apps built in react native. This library is used in conjunction with Google maps so an API key will be required.

### Firebase

Firebase is a backend service that helps developers build real-time apps for iOS, Android and the web that can store and sync data instantly (Lardinois, 2014). I will be using it to handle the majority of my backend functionality. This will include storage for images and audio files, a real-time database for storing data for tours, destinations, and user information, and user creation and authentication.

# System and Analysis Design

## User Stories

The below table shows the user stories for my apps, i.e. the functionality that each type of user requires and why this is. The users will be broken into three categories, Administrators and Tour Guides, who will interact mainly with the web app, and Tourists who will interact exclusively with the mobile app.

|  |  |  |  |
| --- | --- | --- | --- |
| # | As a | I want to be able to | So that |
| US1 | Tour Guide | Create an account | I can create tours |
| US1.1 | Tour Guide | Create tours | Customers can view and take them |
| US1.2 | Tour Guide | Create destinations | I can group them together to make a tour |
| US1.3 | Tour Guide | Edit tours | Maintain my tours |
| US1.4 | Tour Guide | Delete tours | Maintain my tours |
| US1.5 | Tour Guide | Edit destinations | Maintain my destinations |
| US1.6 | Tour Guide | Delete destinations | Maintain my destinations |
| US2 | Tourist | Create an account | I can take tours |
| US2.1 | Tourist | View all tours | I can make a decision on what tour I like |
| US2.2 | Tourist | Assign a tour to my account | I can take the tour now or later |
| US2.3 | Tourist | Remove tour from my account | I can maintain my tour list |
| US2.4 | Tourist | Search tours | I can find nearby tours more easily |
| US3 | Administrator | Remove users from the system | To remove any troublesome/abusive users |
| US3.1 | Administrator | Remove tours | To remove any offensive content |
| US3.2 | Administrator | Remove destinations | To remove any offensive content |

Figure : User Stories

## Class Diagrams

The below diagram shows how the different classes will be built and how they will interact with each other.

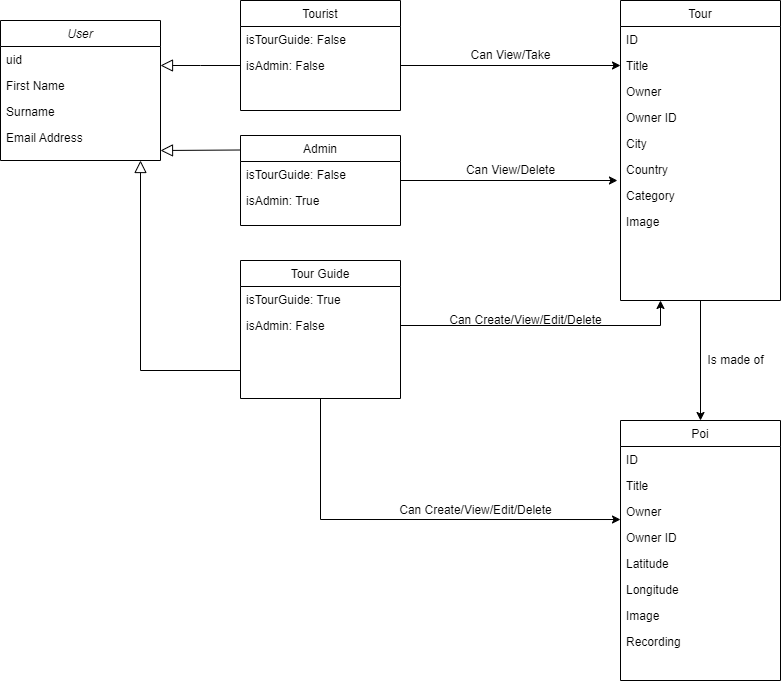


Figure : Class models for TourPal

## Functional Requirements

### Core Requirements

Users will need to be able to, firstly, create an account and log in. Once logged in they will need to be able to create destinations that will make up their tours. Both tours and destinations will need to be persistent so the data will be stored in a Firebase Realtime Database.

Storage will also be required as each destination will have an associated image and recording, and each tour will have an associated image. Users will need to be able to upload the files to online storage, again handled by Firebase, then the URLs for these files will need to be saved to the Tour or Destination model.

### Non-functional Requirements

In terms of security, this will generally be handled by Firebase Auth, which uses its own password hashing. If I were to ever look at giving Tour Guides the ability to charge for their tours, then further security measures will need to be looked into. I will also need to implement validation for my forms.

### Stretch Goals

The first stretch goal I would like to aim for once the base apps are complete, would be to give users the ability to create an orienteering style tour. With this type of tour, markers will not appear on the map and the tourist users will use recorded clues to find the next destination.

Following this, I would also like to include some more social functionality such as being able to like or dislike a tour, or a star based review system. The ability to comment on tours would also be a useful feature.

## User Interface Design

### Web App

The below diagram gives an idea of what the UI of the web app will look like and what screens users will navigate to.

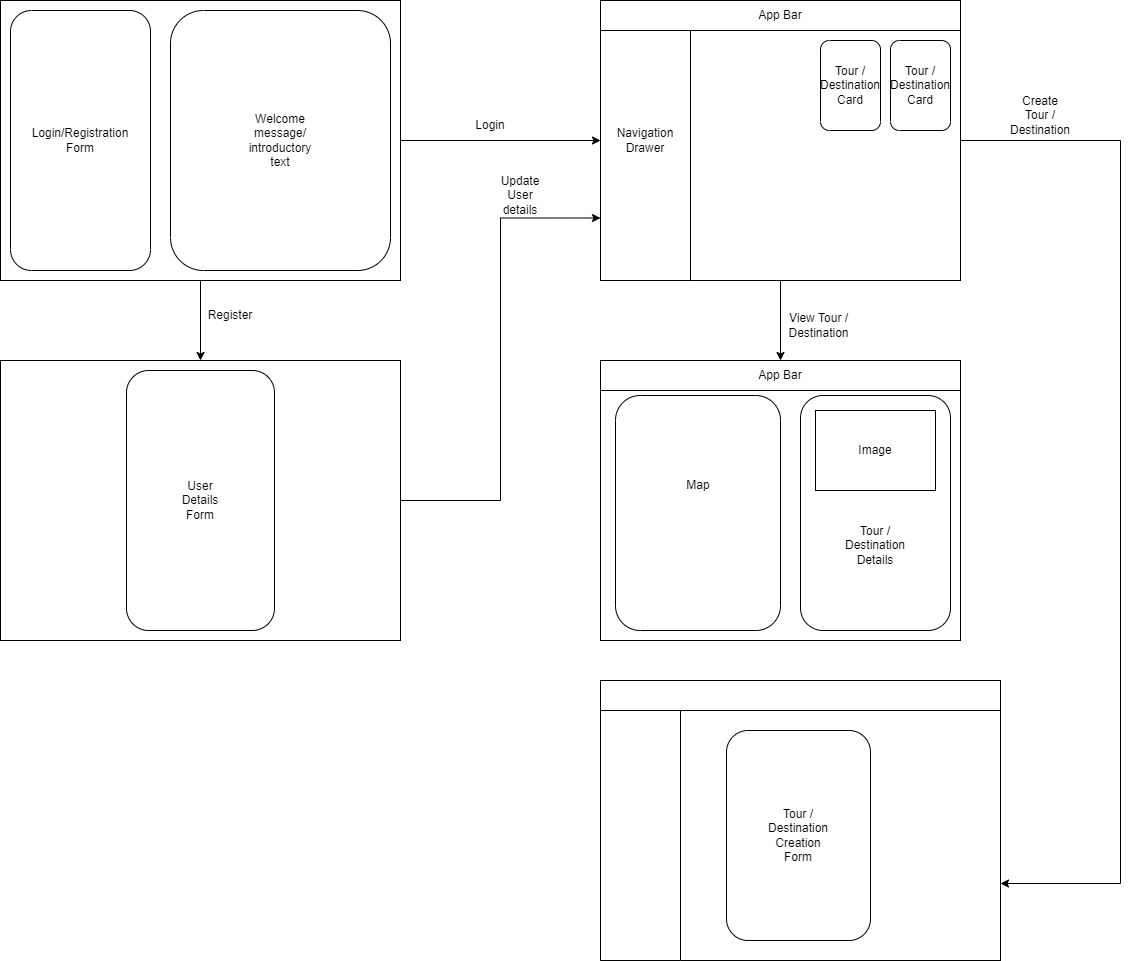


Figure : Web App UI and Navigation

### Mobile App

For my mobile app, I took a look at other apps to model from. I had decided to use React Native Paper to help with the design and feel of the app also so it would need to compatible with this also. In the end I looked to that app that I would use most that also incorporates a map, namely Strava, the running app. The main design note I took from this was to use a single vibrant colour to associate with the app and a white background.

# Methodology

## Methodology Selection

After looking at a few different application development methodologies, I have decided to go with a version of the rapid application development model (RAD). The RAD approach is a form of Agile software development emphasizing rapid prototype releases and iterations (Cox, 2019).

There are many benefits to the RAD method that makes it the ideal working ethos to apply to my own project. First is flexibility as developers can make adjustments to their code quickly during the development process. As I am working with a clear deadline, the speed of delivery when working using this method is also an advantage.

The greatest advantage of the method in relation to my own project, however, is the encouragement of code reuse (Cox, 2019). This is a natural fit working with component based library such as React as I will be creating components and then re-using them for different pages/purposes within the web app. Considering I am then moving on to create the mobile app in React Native, where a lot of my code from the web app will carry over, something along the lines of the RAD method is the obvious choice.

The RAD method consists of 5 distinct phases, which I will be tweaking to fit my current situation, i.e. working by myself on a college project as opposed to as part of a Team.

## Process Outline

### Phase 1: Define and finalize project requirements

During this phase, stakeholders in the project will meet to define and finalize project requirements such as project goals, expectations, timelines, and budget (Cox, 2019). At a macro-level, the stakeholders are myself and my supervisor and we will be having regular meetings to discuss the project but in terms of what the project is and what it needs to do, this has been decided early on. I will also be applying this phase, and all phases, on a micro level. This would mean on a component by component basis. I will be planning what the component needs to do, what it needs to look like, etc. At this level, I am the only stakeholder.

### Phase 2: Begin building prototypes

In this phase, designers and developers will work closely with clients to create and improve upon working prototypes until the final product is ready (Cox, 2019).

For this phase, and subsequent phases. I will be approaching it on a component by component basis. As I am, in a way, the client, the developer, and the designer, this phase will largely involve drawing up design for a component, keeping in mind what its purpose is, and then reproducing this idea in code.

### Phase 3: Gather user feedback

Once the components are put together as a page, one would generally gather user feedback. In this case the feedback would be coming from myself so if I am not happy with how the project looks and functions, it will be back to the drawing board.

### Phase 4: Testing

This process will involve testing the product, both as individual components and together as a whole project.

### Phase 5: Present your system

Under the RAD method this phase will involve data conversion, user training, and (possibly) more testing (Cox, 2019). For me, this will mainly involve setting up usable user accounts and tours that can be taken with these accounts. It will also, hopefully, include deploying both apps so that it can be used by external users.

## Project Schedule

|  |  |  |
| --- | --- | --- |
| Week | Date | Development Work |
| Week 1 | 17 January | Work on web app |
| Week 2 | 24 January | Work on web app |
| Week 3 | 31 January | Have basic web app completed |
| Week 4 | 7 February | Work on interim report/Start mobile app |
| Midterm | 14 February | Continue mobile app |
| Week 5 | 21 February | Continue mobile app |
| Week 6 | 28 February | Have basic mobile app completed |
| Week 7 | 7 March | Work on UI/Stretch goals |
| Week 8 | 14 March | Stretch goals |
| Week 9 | 21 March | Work on Final Report |
| Week 10 | 28 March | Project Submission |
| Week 12 | 04 April | Work on Video and Presentation |

Figure : Project Schedule

# Implementation

## Web App

### Phase 1: App Creation and Login

#### Planning

During the planning phase, I decided that the best way to get started with the app was to create the app and then go straight into creating the Login Page and implementing Firebase Authentication. My reasoning for this was to avoid backtracking. The Login Page is the first page users will come into contact with so it should be the first thing I work on. I also wanted to get Firebase Auth up and running as this would mean, once I started working on the User Model, a lot of the base information will already be there and be functional, e.g. the uid, email, etc.

#### Review

I began by running Create React App which does a lot of the heavy lifting with regard to creating a React web app, creating a modern build setup with no configuration. Once this was done and the app was created, I created a test page that I could use to test the different components I would be creating as I went along and then began work on setting up Firebase Auth and putting together a login form.

I had previously set up Firebase Auth on a Kotlin app but this was my first time doing so with React so I consulted a YouTube video for a step by step guide on how to implement (Simplified, 2020). Once I had set up a project on the Firebase site, I then went on to create an .env file which would hold the API key and various other vital values for firebase to be operational on my app. This was followed by creating a firebase.js file which would handle communication with and imports from Firebase. This can be seen below.

Text

Description automatically generated

Figure : Firebase.js file

I then created a separate contexts folder in my application and a JavaScript file to house my authentication methods, which I then implemented into my login form.

After testing the form on my test page to ensure both registration and login functions worked, I created a login page which I then put the finished form into. The process was fairly straight forward and I was then able to create a new users for my web app.

Graphical user interface

Description automatically generated

Figure : Firebase Auth

#### Evaluation

All in all, implementing authentication through Firebase is straightforward with a good deal of supporting material online to assist. One issue arising from this is how to approach setting up the user model. The model will require the uid but this is not created until the user is created on Firebase Auth. As such, I will need to implement a two-step registration process down the line.

#### Reflection

These initial steps have made me more confident in my choices of technology. React and Firebase work together fairly well and the setup was not very intensive.

### Phase 2: Navigation

#### Planning

With regard to planning my navigation, I decided to use React Router to handle this as this is the most common practice. My plan was to have a Login Page act as the landing page, which would then lead to several protected pages depending on if the user logged in or registered. A diagram outlining the navigation can be seen above.

#### Implementation

We had previously covered navigation with React in the *ICT Skills 2* module so I was confident that putting together navigation for the app would be relatively straightforward, and that I could base my Private Routes off of previous assignments. However, in v6 of React Router the implementation of Private Routes has changed completely. Luckily, I was able to find a comprehensive guide on a new method to implement private routing (Luca, 2021).

I began by creating a Private Route component. The code for this can be seen below, with the code of the previous method of creating a Private Route commented out:

Text

Description automatically generated

Figure : Code to implement private routes in React JS

As you can see the code has been simplified greatly, going from multiple lines of code to just two.

#### Evaluation

Having read up on the new process for creating Private Routes, it was easier to implement than the initial method that I had learned.

#### Reflection

This process has really highlighted just how quickly things can change in the world of Web Development. It was interesting to see that within less than a year a method that we had learned was completely obsolete and would not work at all.

### Phase 3: Tour Card and Tour Card List

#### Planning

The next thing I wanted to work on was how individual tours and destinations would display for users to select them. I decided to display each tour or destination on a card with an image, name. location (made up of the city and country), and buttons depending on the context they are viewed. I would use Material UI to achieve this design (MUI, 2022). To display the cards as a list I would create a tour list file that would be largely based on code from the *ICT Skills 2* module.

#### Implementation

The main challenge when approaching creating the tourCard.js was implementing it to handle the context of what the user is meant to be viewing at the time. To do this, I decided to add a poi Boolean to the destinations. If this Boolean is present, then the desired button set would be displayed. You can see this in the UsableButtons function below:

Text

Description automatically generated

Figure : Use of poi Boolean to implement different button sets

#### Evaluation

Implementing the tourCard.js and tourList.js proved to be an interesting challenge that had me thinking about how React actually works, i.e. passing props to different components. This proved to be extremely useful and once I had the Firebase Realtime Database up and running I was able to come back to my tourCard.js file and code it to take a third context into account.

If a user wants to add destinations to a tour, then a similar page is displayed with cards showing the destinations but this time the buttons change to allow them to add the destination to the tour. This is achieved using the useParams hook from React Router Dom. If there is a uid from a tour in the url, then than tour is called from the database and the useState hook used to set it as an activeTour. If an activeTour is present, then another set of buttons is displayed:

Text

Description automatically generated

Figure : Use of activeTour to determine button set

#### Reflection

I found implementing the cards and lists quite rewarding and educational. The fact that I could use the same component for three different use cases, and do it with relative ease, shows how useful the React library can be.

Graphical user interface

Description automatically generated

Figure : All Tours screen showing three tour cards

### Phase 4: Maps

#### Planning

I decided to use Leaflet JS to implement maps on the web app to keep costs at a minimum. It was also at this point that I created a Test Page which I would use to first build my components before building the actual page that they would appear on. I watched a step by step video on the use of Leaflet JS with React in preparation for this (LogRocket, 2020). Due to the nature of implementing markers for the map, I decided it would be simplest to have two map components, one for tours and one for destinations.

#### Implementation

I started by installing the Leaflet JS library and then created a mapViewPoi.js file. This would be the map component for single destinations. The map centres on the location indicated by the latitude and longitude values of a destination and places a marker there. When the user clicks on marker, the destination is set as the activePoi and a popup is displayed.

The mapViewTour.js works in a similar fashion but the associated destinations from a tour are added to an array. The latitude and longitude of these are then mapped to markers and the map is centred on the first destination of the array. Starting the app I used hardcoded models but once Realtime Firebase was set up for the app, I could then pull the associated destination from the “pois” database and put them in the array. Again, an activePoi is set when the marker is clicked and a popup is displayed for the given marker.

Map

Description automatically generated

Figure : Map View for a tour

#### Evaluation

Leaflet JS proved the correct choice for my map functionality for the web app. There is a good deal of documentation on the library and it is relatively easy to implement. Using an open source library for one of the apps would greatly decrease costs should I ever choose to launch the app as a service to the public.

#### Reflection

The choice of how to implement mapping proved to highlight the importance of extensive research for me in relation to this project. I looked into a few different choices such as OpenStreetMap, Google Maps, and Mapbox and Leaflet really did seem to be the best choice both in terms of cost and compatibility with React.

### Phase 4: Firebase Realtime Database

#### Planning

When planning how to handle my data, I decided to implement Firebase Realtime Database as we had previously covered it in the Mobile App Development module for Kotlin. I watched another video to get to grips on how to get this working in React (Coding, 2020). The data I would need to save in the database would be that for the individual destinations (POIs), the tours, and the users. Users would be added at a later date.

#### Implementation

I returned to my firebase.js file and imported database functionality from Firebase and exported the firebase.database() function as a const, db, that could be called by different components. I then imported db, as well as the set and ref methods from Firebase, into new form components for creating tours and destinations.

As I am using React, I needed to use the useState hook in order to get the form working in terms of inputting data and assigning it to a variable. Then I need to create a handleCreateTourClick method for when the user clicked the button to submit the data. This method used db, and the set and ref methods to create a path to the required dataset (if none exists), and adds the data.

Text

Description automatically generated

Figure : handleCreateTourClick method

As you can see in the above image, a uid is used to separate the different items in each section of the database. This is created by calling uuid4. The above is the finished method, originally I did not have an image associated with a tour or POI as storage had not yet been set up.

#### Evaluation

Having Firebase Auth already set up for the project made coding to set up the Realtime Database relatively easy.

#### Reflection

I think my biggest takeaway from implementing Realtime Database is how versatile it is in structuring the database, using set and ref to create a pathway. If I had more time to plan, I would probably have come up with a more feasible structure for the database. At the moment is divided into users, tours, and POIs (destinations) and this is fine for the current state of the app. However, moving forward, a possible option would be to store both POIs and tours within the user section of the database. This would make it far easier to find a tour or destination from a Customer Service standpoint if there were a large number of users, tours, and destinations.

### Phase 5: Firebase Storage

#### Planning

The next step was to implement storage which, again, I used Firebase for. Again, I watched another video on implementing Firebase Storage with React to get to grips with it (Simplified, 2021).

#### Implementation

As with the Realtime Database, I imported the functionality into my firebase.js file and exported from there as storage. This was them imported to the required components as well as the ref function from Firebase Storage. I then went on to use these to create an uploadFiles function.

A screenshot of a computer

Description automatically generated with medium confidence

Figure : uploadFiles method

If you look at the uploadFiles function above, it actually serves 3 purposes. Firstly, it uploads the file, in this case an image file for a tour, to the storage bucket. It also sets the progress of the upload as a percentage. This is then used to give a visual que to the user of the progress of the upload. Finally, once the file is uploaded, it gets the url associated with it and assigns it to the image variable. This is then associated with the tour when the user clicks to create a tour.

#### Evaluation

As with the implementation of Realtime Database, setting up Firebase Storage was relatively straightforward. One issue arose from testing and that is that if a file with the same name is uploaded, it will replace the one in the bucket, but change the url. This means that the image or recording associated with a previous tour or destination will break. As you can see from the code above, I have fixed that problem by including the time and date of upload as well as the file name when uploading.

#### Reflection

One takeaway that I had from the experience of implementing not just Firebase Storage but Firebase Auth and Realtime database as well, was how much easier coding can be when you are using the same service for these backend functions. The syntax for storage and the databases are quite similar and I could have a lot of the functionality, APIs, and other required authentication all called in the one firebase.js file which I could then pull from.

### Phase 6: Media Player

#### Planning

Originally I had planned to use a library such as react-native-media-player to handle my media playback but I could not find any decent documentation and that particular library has not been updated in four years. In the end, I managed to find a video that showed how to create a media player using the audio HTML tag and the useRef hook in React to create a functional media player (ua, 2020).

#### Implementation

Once I had designed the look of my media player I inserted a HTML audio tag that used the recording associated with a destination as its source and set the ref to player, which I had previously called with the useRef hook. The useRef hook can be used to store a mutable value that does not trigger a re-render when updated (Pavlutin, 2021). With this player attribute I could manipulate the playback of the audio.

I also used the useState hook to create a Boolean called isPlaying that could be used to tell if the audio was playing at a given time. I used this so that I could create s single Play/Pause button whose icon would change if audio was playing or not.

Text

Description automatically generated

Figure : Various methods to handle audio playback

Once the Media Player component was created, I called it in the popup of the marker for the destination.

Diagram

Description automatically generated

Figure : Media Player

#### Evaluation

The method used for my media player showed the usefulness of hooks such as useRef in React. With it I was able to make a custom media player which looks much better and is more in keeping with the visual theme of the web app than the standard HTML controls. Also, the imports required were more aesthetic based than requiring any libraries to operate the audio playback.

#### Reflection

This was my first time attempting any kind of audio playback in my code so I had never even used the audio tag in HTML. The implementation of this component has shown me that you do not need to stick completely to the language you are using, in my case JavaScript, to achieve your goals and that html tags integrate quite well with React.

### Phase 7: Further CRUD Implementation

#### Planning

The final step in creating my web app, besides general cleanup and fixing some theming for the UI, was to go back to my implementation of Realtime Database and Storage and ensure that users could delete and edit tours in the app. To make the addition of an update feature for destinations and tours a bit easier, I decided to make a separate update form component for reach, rather than refactoring the existing creation forms. This would give me a bit more freedom in experimenting with my coding without risking breaking the initial create forms.

#### Implementation

I started by creating update forms that were largely based on the creation forms with some key differences. These forms take in the tour or destination and the attributes that need to be changed are then taken as props. These then show in the text input fields so that the user does not have to re-enter anything if they are only changing one value.

I then used the useState hook for each value, with the initial value set to the original. These update as the user types in new data. The method used to update the tour or destination is much the same as the one to create, just taking in the new values and using the same uid.

Text

Description automatically generated

Figure : handleUpdatePoiClick function

Implementing the deletion of tours and destinations was simply a matter of writing a method that would find the correct tour or destination and then calling the .remove() method for it in Firebase. I also needed to ensure that any related files for images or audio were also deleted to save on space in my storage bucket.

Text

Description automatically generated

Figure : handleDeleteTour method

It was also at this point that I created a list of destinations that would appear on the Tour Details page that would allow the user to remove a destination from the tour.

Graphical user interface, text, application, email

Description automatically generated

Figure 18: Tour Destinations List on Tour Details page

#### Evaluation

With that the web app was more or less complete. Again, this process showed just how easy Firebase is to work with and I was able to re-use a lot of the code I had used previously to implement the changes here, particularly when it came to creating the update forms.

#### Reflection

On reflection, one thing I could have done differently would be to have attempted using the createTourForm and createPoiForm components for updating also. Had I gotten this to work, there would have been just two files to maintain as opposed to four.

## Mobile App

### Phase 1: App Creation on Expo, Navigation, and Firebase Auth

#### Planning

Starting the mobile app seemed like a daunting task at first as I had not worked with React Native before. In the end, I came across a very helpful video for React Native beginners that also recommended using Expo when creating your React Native App (Mosh, 2020). I decided to approach the mobile app in a similar fashion to how I had handled the web app, by starting with the Login page and setting up Firebase Auth. I would also need to implement some form of navigation to ensure that this worked.

As well as all of this, I also decided to use React Native Paper for the design of my app. This is a standard-compliant Material Design library that is similar to Material UI (React Native Paper, 2021).

#### Implementation

Firstly, I needed to set up and Expo account at <https://expo.dev/> and then install Expo CLI by running “npm I -g expo-cli”. Once everything was installed, I ran “expo init walking-tour-mobile” to create the app.

I then had to add my new mobile app to my existing Firebase project through the Firebase console. Once this was done, I copied the code from the firebase.js file in my web app to a new file of the same name in the mobile app and updated the values in the firebase.initializeApp method accordingly . Unfortunately, I was unable to get a .env file working with the mobile app so the actual values are currently housed in the firebase.js file. This is an obvious security issue that will need addressing down the line. I also copied across my AuthContext.js file.

Next, I worked on putting together a Login Form component and a Login Screen to house it. For the Login Form, I simply copied across most of the code from my web app again and changed some of the JSX to be compatible with React Native and React Native Paper.

Finally, I needed to edit my App.js file to view and test these components. To set up navigation, I needed to install React Navigation and then import NavigationContainer this would be wrapped around all components of the app. I also decided to implement Stack Navigation for now. This provides a way for your app to transition between screens where each new screen is placed on top of a stack (React Navigation, 2020).

I also imported my AuthProvider from AuthContext and PaperProvider from React Native Paper. Once I had also imported my screens, the App.js file looked something like this:

Text

Description automatically generated

Figure : App.js

I would later go on to implement Tab Navigation also, so that I could include bottom tab navigation. The finished app utilizes a combination of both navigation styles.

#### Evaluation

Upon running the app, everything seemed to work fine. Existing users could log in to the mobile app and I could also register new users with it. The whole process was quite straight forward, just with some minor issues. The syntax of both apps are quite similar both being JavaScript but there are some slight differences between React and React Native that can take a bit of getting used to, namely the use of the navigator in React Native vs routing in React.

#### Reflection

These first steps in creating the mobile app have shown that creating the app in React Native was, indeed, the best choice. A lot of the code from my web app has carried over with little to no editing which has definitely sped up the process of app creation.

### Phase 2: Accessing Realtime Database

#### Planning

Next I needed to work on setting up Realtime Database on the mobile app in order to display the different tours. As much of the code would carry over, the largest portion of the work would be planning on how this data would be displayed. I decided to utilize cards again for this.

#### Implementation

As with the web app, I created a separate component for the actual Tour Cards, the list of the cards, and the screen that they would be displayed on. I used the useEffect hook to access the Realtime Database when the All Tours screen is initialized and then pass all tours to an array. This array is then passes as a prop to the Tour List component.

Text

Description automatically generated

Figure : useEffect method accessing Realtime Database

The Tour List component then maps all tours to the individual Tour Cards, which are displayed a list that looks like this:

Graphical user interface, application

Description automatically generated

Figure : All Tours screen with two Tour Cards

#### Evaluation

This also proved to be straightforward to implement as much of my code carried over from the web app. Using React Native Paper also made the creation of the cards manageable and allowed the finished product to look quite well.

#### Reflection

From these first two sections, I am confident in saying the React Native was the correct choice for my mobile app. As much of my code carried over, I was able to make some decent headway on the app relatively quickly. It was after these initial steps that building the app became a bit more challenging.

### Phase 3: Maps

#### Planning

For the map component of the app, I put a lot of effort into researching what API to use. From the start, it became clear that, as I had chosen to build the app on Expo, I was locked out of using many map APIs. I was still trying to keep costs down so I initially tried to use react-native-webview-leaflet which is a Leaflet map component with no native code for React Native applications (reggie3, 2020). This proved fruitless, however, and I was forced to use React Native Maps. This provides a map component that is compatible with Expo but requires a Google Maps API Key to function on Android (Expo, 2022).

#### Implementation

I first tried to create a Map Card as a separate component and the feed the data from a tour to it as props. This Map Card would be passed a tour as a prop and would use the associated POI uids to pull the required POIs from the database and add them to an array. The latitude and longitude from each POI in this array was then mapped to a marker on the map.

Text

Description automatically generatedText

Description automatically generated

Figure : Destinations being assigned to markers in Map View

As with the web app before it, clicking on the marker sets the destination as the activePoi and causes a popup to appear, this time over the marker as opposed to in its place.

#### Evaluation

This was one of the most time consuming parts of the project, largely due to the fact that before settling on React Native Map, I did put a lot of time into researching other free map APIs and even trying to get some of the work with the app. In the end, React Native Maps was straightforward to implement and looks and works quite well with the app.

Graphical user interface, application, map

Description automatically generated

Figure : Tour being taken

#### Reflection

The process of implementing the maps has taught me that sometimes the most straightforward solution is often the best. During my research it was becoming clear that though some map APIs may work with React Native, even less work with Expo and in the end, I went with the option that the Expo docs recommended. This was after two to three days of trin to get another to work.

### Phase 4: Media Player

#### Planning

For the media player I had hoped that I could simply copy across the code from the web app and refactor to change from syntax compatible with React and Material UI to React Native and React Native Paper. This proved not to work.

However, some quick research showed that Expo has its own library for handling audio, Expo AV (Expo, 2022).

#### Implementation

I installed Expo AV on my app and using that was able to initiate playback on the recording associated with a destination. As with the web app, I used an isPlaying Boolean to control what icon appears on the Play/Pause button.

Text

Description automatically generated

Figure : Functions handling audio playback of mobile app

I had initially wanted to display the media player in the popup for the marker, as with the web app, but could not get this to work so instead I created a modal that will display when the user presses the popup.

Graphical user interface, application

Description automatically generated

Figure : Media player for mobile app

### Phase 5: Geolocation

#### Planning

After looking into how to have my app use the user’s device location to tell if they were close to a destination, I discovered that Expo also has a location library, Expo Location (Expo, 2022).

#### Implementation

Once I had installed Expo Location on the app, I took the code from the docs to get the current location of the user’s device and ask them permission to do so and put it into a useEffect hook.

Text

Description automatically generated

Figure : Getting user location on Tour Taking screen startup

This then set the user’s latitude and longitude as the value for currentLat and currentLng which would then be passed into the Map Card and centre the camera on that location and create a marker indicating the user’s location. However, this is where I ran into a problem.

This Map Card component did work initially and displayed markers for each destination on the tour. However, I needed a map that would continuously update with the user’s position which would mean updating the props being fed into the Map Card component regularly. Unfortunately I could not get this to work and instead had to combine most components used on the Tour Taking screen into the one file. The goes against the concept of separation of concerns but allowed for the updates that I needed.

I could now tackle giving the user the ability to update their location at the touch of a button. Once they were then in the vicinity of a destination on a tour, the modal with the media player would be displayed and they could listen to the audio for that location. I achieved this with my getLocation method.

Text

Description automatically generated

Figure : getLocation method to update user location

As you can see above, if the user is within 0.0002 of the latitude and 0.0002 of the longitude, then the modal will pop up with the recording for that given location. I came to the value of 0.0002 by a process of trial and error, creating a tour just outside of my house and seeing how close or far away from a point I had to be for different values to work.

### Phase 6: User Model

#### Planning

Finally, I needed to go back and implement the user model that I had planned for the app. This would mean creating a user database in Realtime Database first of all, then allowing users to assign themselves tours and giving different permissions to different types of users.

#### Implementation

The first thing to address was how to create a user in the database. As someone would need to already have set up their account using Firebase Auth so that they had a uid, I would need a separate registration form that is displayed after the user registers on the Login Screen. I created this for both the web app and mobile app. The difference between both forms is that when the user creates their account using the mobile app the tourGuide value is set to false while on the web app it is set to true.

Text

Description automatically generated

Figure : Creating a user in Realtime Database

I then went back to the mobile app and on the Tour Details Card I added a method allowing users to add a tour to their account. How this works is by adding the uid of then tour to the user’s entry in the database.

Text

Description automatically generated

Figure : Add tour to user account

This is then used to call all of the tours on the user’s account when on the new My Tours screen. This screen is essentially the same as the All Tours screen but only calls the tours that the user has assigned to their account. This is now the only way to take a tour as I removed the Take Tour button from the cards on the All Tours screen. My reasoning for this was that if I were to ever release the app as a paid service, this would help to facilitate it as a user would need to pay to have an app added to their account. Also, if there were a large number of tours on the app, it would make for a better user experience to have only the tours that they are interested on one page. Users can also remove tours from their account.

#### Evaluation

Implementing the user model was relatively easy thanks to all of the work I had done so far with Realtime Database. Also, using conditional rendering it would be a simple enough task to keep users with different permissions from seeing screens that you would not want them to.

#### Reflection

My only issue with implementing the user model in my database was how long I left to do it. With better planning I could have implemented it earlier and saved myself a lot of going back through my code and refactoring.

# Evaluation

In the end I believe that this project was a success in the sense that I have built a working app suite that allows users to create and take tours anywhere in the world, which is what I set out to do. I managed to meet most of my core goals.

Using the technologies and libraries available to me I managed to create a web app that allows users to create an account first of all. These users are then considered to be Tour Guides. These Tour Guides can then go on to create, edit, and delete locations. These locations have associated images and audio recordings. The user can then use these locations to create workable walking tours, which can also be edited and deleted.

Users using the mobile app are considered to be Tourists and they can see all tours, assign the ones they want to take to their account and then take the tour. When nearing a destination on the tour, they press a button to get their current location and when near the destination and a modal will appear that has the tour’s image and allows them to play the audio for the tour. If their location is not displayed correctly, they also have the option to click on the destination to listen to the audio. They can then remove tours from their account when they are done. This is exactly how I had planned the app to work.

Of course I did run into some issues with the app which needed to be resolved. Most of these relate to myself being overly ambitious with the project. The greatest example of this is the amount of time I spent trying to find a working free map API that would be compatible with the technologies I had chosen to create my mobile app with. In the end, I just had to go with the recommended API and this would likely incur costs down the line should I ever choose to launch the app. All in all though, I am happy with how the project turned out.

# Conclusion

In conclusion, I hope this report shows that I have achieved a great amount in building both of my apps. Most goals were achieved and the app is a workable service.

However, what has become clear to me from this process is that there are areas that I need to work on. The first, and most important of these, is planning. I did the planning myself and largely wrote down what I wished to do on pen and paper and did not avail of service like Trello to outline my goals. This was probably a mistake and I would have likely reached my stretch goals if I had taken a more structured approach that this would have provided.

I also need to work on design. I think both of the apps could look better but in particular the web app. As the web app would be what the users that could be considered “professionals” would be using, a sleeker design would certainly be required to attract more users.

As for the future of the project, as I have indicated. I would definitely like to work on my design skills and create a better looking app suite. I would also like to work on the orienteering stretch goal that I had set for myself as I believe that this would be one function that would set my app suite apart from its competitors.

# Table of Figures

[Figure 1: Class models for TourPal 7](#_Toc101101155)

[Figure 2: Web App UI and Navigation 9](#_Toc101101156)

[Figure 3: Firebase.js file 12](#_Toc101101157)

[Figure 4: Firebase Auth 13](#_Toc101101158)

[Figure 5: Code to implement private routes in React JS 14](#_Toc101101159)

[Figure 6: Use of poi Boolean to implement different button sets 15](#_Toc101101160)

[Figure 7: Use of activeTour to determine button set 16](#_Toc101101161)

[Figure 8: All Tours screen showing three tour cards 16](#_Toc101101162)

[Figure 9: Map View for a tour 17](#_Toc101101163)

[Figure 10: handleCreateTourClick method 19](#_Toc101101164)

[Figure 11: uploadFiles method 20](#_Toc101101165)

[Figure 12: Various methods to handle audio playback 22](#_Toc101101166)

[Figure 13: Media Player 22](#_Toc101101167)

[Figure 14: handleUpdatePoiClick function 24](#_Toc101101168)

[Figure 15: handleDeleteTour method 25](#_Toc101101169)

[Figure 16: Tour Destinations list on Tour Details page 25](#_Toc101101170)

[Figure 17: App.js 27](#_Toc101101171)

[Figure 18: useEffect method accessing Realtime Database 28](#_Toc101101172)

[Figure 19: All Tours screen with two Tour Cards 29](#_Toc101101173)

[Figure 20: Destinations being assigned to markers in Map View 30](#_Toc101101174)

[Figure 21: Tour being taken 31](#_Toc101101175)

[Figure 22: Functions handling audio playback of mobile app 32](#_Toc101101176)

[Figure 23: Media player for mobile app 33](#_Toc101101177)

[Figure 24: Getting user location on Tour Taking screen startup 34](#_Toc101101178)

[Figure 25: getLocation method to update user location 35](#_Toc101101179)

[Figure 26: Creating a user in Realtime Database 36](#_Toc101101180)

[Figure 27: Add tour to user account 37](#_Toc101101181)

# Bibliography

Coding, D. W., 2020. *YouTube - Crud operation with react and firebase realtime database..* [Online]   
Available at: https://www.youtube.com/watch?v=v0TKYSkZ2tI  
[Accessed 26 January 2022].

Cox, T., 2019. *Capterra.* [Online]   
Available at: https://blog.capterra.com/what-is-rapid-application-development/#definition  
[Accessed 12 February 2022].

Expo, 2022. *Audio.* [Online]   
Available at: https://docs.expo.dev/versions/latest/sdk/audio/  
[Accessed 20 February 2022].

Expo, 2022. *Location.* [Online]   
Available at: https://docs.expo.dev/versions/latest/sdk/location/  
[Accessed 3 March 2022].

Expo, 2022. *Mapview.* [Online]   
Available at: https://docs.expo.dev/versions/latest/sdk/map-view/  
[Accessed 12 February 2022].

Lardinois, F., 2014. *TechCrunch.* [Online]   
Available at: https://techcrunch.com/2014/10/21/google-acquires-firebase-to-help-developers-build-better-realtime-apps/?guccounter=1&guce\_referrer=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnLw&guce\_referrer\_sig=AQAAAGeodZJSMgS3exGPN2pxnED2gAprUT3i32Y\_b0uBPCZg4Grl01QN-Prc06WhZNanH  
[Accessed 13 February 2022].

LogRocket, 2020. *YouTube.* [Online]   
Available at: https://www.youtube.com/watch?v=290VgjkLong  
[Accessed 21 January 2022].

Luca, A., 2021. *DEV.* [Online]   
Available at: https://dev.to/iamandrewluca/private-route-in-react-router-v6-lg5  
[Accessed 16 January 2022].

Mosh, P. w., 2020. *YouTube - React Native Tutorial for Beginners - Build a React Native App.* [Online]   
Available at: https://www.youtube.com/watch?v=0-S5a0eXPoc&t=563s  
[Accessed 29 January 2022].

MUI, 2022. *MUI Core, Material UI, Card.* [Online]   
Available at: https://mui.com/material-ui/react-card/  
[Accessed 16 January 2022].

Pandit, N., 2021. *C#Corner.* [Online]   
Available at: https://www.c-sharpcorner.com/article/what-and-why-reactjs/  
[Accessed 06 February 2022].

Patel, J., 2022. *Monocubed.* [Online]   
Available at: https://www.monocubed.com/why-use-react/  
[Accessed 06 February 2022].

Pavlutin, D., 2021. *The Complete Guide to useRef() and Refs in React.* [Online]   
Available at: https://dmitripavlutin.com/react-useref-guide/  
[Accessed 28 January 2022].

Ravichandran, A., 2020. *Adhithi Ravichandran.* [Online]   
Available at: https://adhithiravi.medium.com/building-react-native-apps-expo-or-not-d49770d1f5b8#:~:text=With%20Expo%2C%20you%20can%20build,iOS%20and%20Android%20with%20ease.&text=With%20Expo%2C%20you%20will%20never,while%20building%20apps%20using%20Expo.  
[Accessed 6 February 2022].

React Native Paper, 2021. *React Native Paper.* [Online]   
Available at: https://reactnativepaper.com/  
[Accessed 29 January 2022].

React Navigation, 2020. *Stack Navigator.* [Online]   
Available at: https://reactnavigation.org/docs/stack-navigator/  
[Accessed 29 January 2022].

reggie3, 2020. *GitHub - React Native Webview Leaflet V5.* [Online]   
Available at: https://github.com/reggie3/react-native-webview-leaflet  
[Accessed 10 February 2022].

Simplified, F., 2021. *YouTube - Firebase v9 Storage in React | Upload Files to Cloud.* [Online]   
Available at: https://www.youtube.com/watch?v=pJ8LykeBDo4  
[Accessed 28 January 2022].

Simplified, W. D., 2020. *YouTube.* [Online]   
Available at: https://www.youtube.com/watch?v=PKwu15ldZ7k  
[Accessed 15 January 2022].

ua, j., 2020. *YouTube - React Audio Player - Part 2.* [Online]   
Available at: https://www.youtube.com/watch?v=BZibbA9sTqQ  
[Accessed 28 January 2022].