Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Final Year Project Report
Final Report

Ringforts of Ireland

David Corrigan (20091410)

Higher Diploma in Computer Science

Supervisor: Peter Windle

Department of Computing & Mathematics

Waterford Institute of Technology

Declaration

I declare that the work which follows is my own, and that any quotations from any sources (e.g., Books, journals, the internet) are clearly identified as such by the use of 'single quotation marks', for shorter excerpt and identified italics for longer quotations. All quotations and paraphrases are accompanied by (date, author) in the text and a fuller citation is the bibliography. I have not submitted the work represented in this report in any other course of study leading to an academic award.

Student..... Date..11/4/2022.

Work Place Mentor..... Date.....

Acknowledgements

This course would not have been possible without the support and patience of my family as I locked myself away in the office most evenings for 2 years. Hopefully it will show my kids that learning can and should be lifelong.

I would like to thank my supervisor Peter Windle who supported me through this project over the last 3 months and provided invaluable advice and direction throughout.

I would also like to say a big thank you to all the lecturers on the course who delivered great content in their respective modules. I would especially like to thank Eamonn de Leastar and Colin Dunphy who coordinated this fantastic course.

Finally, I would like to thank my local historian, who wished to remain nameless in this report. He made himself available every time I asked, and provided invaluable ideas and feedback throughout the project.

Table of Contents

Table of Figures

Table of Tables

Glossary of Terms

Agile	A iterative project delivery methodology where the project is broken into sprints of usually 2 weeks that produce tested features as output.
API	Application Programming Interface An API is a set of definitions and protocols for building and integrating application software.
BaaS	Backend as a Service is a cloud computing service model that serves as the middleware that provides developers with ways to connect their Web and mobile applications to cloud services via API and SDK.
CSS	Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML
Hapi Framework	Hapi is a rich node based framework for building applications and services.
HTML	Hypertext Markup Language, a standardized system for tagging text files to achieve font, colour, graphic, and hyperlink effects on World Wide Web pages
IOS	An acronym for iPhone Operating System, is a Unix-derived operating system powering all of Apple's mobile devices.
JSX	A syntax extension to JavaScript used with React to describe the UI layout and design.
Node.js	An open source development platform for executing JavaScript code server-side.
NMS	National Monument Services
REST	Representational state transfer is a set of architectural constraints
Ringforts	Circular Structures built as protective enclosures around farmsteads.
SDK	Software Development Kit providing a set of tools, libraries, relevant documentation, code samples, processes, and or guides that allow developers to create software applications on a specific platform.
User Story	A user story is the smallest unit of work in an agile framework. It's an end goal, not a feature, expressed from the software user's perspective.
Waterfall	A linear project delivery methodology where each phase starts when the previous phase has been completed and signed-off.
WebView	WebView is an app component that display web pages inside your application as native apps.

1 Introduction

1.1 Introduction To Document

The overall purpose of this document is to showcase the skills and learning from the Higher Diploma in Computer Science from Waterford Institute of Technology (WIT). The major technology demonstrated in the project was not covered in the course curriculum, but others were, like Firebase and Google Maps. I believe this project demonstrates other important skills learned during the course, the ability to objectively evaluate the various tools and languages available to determine which are most suited to the task at hand and secondly, the ability to independently learn any new technologies as required.

This project also demonstrates other non-technical skills learned and practiced during the course; particularly during the assignments. These were design and planning skills, which are required to execute any project successfully.

This report documents the development methodology which I used throughout the project and shows the progress made and re-planning required at different points. It also shows how I evaluated the various technologies and the reason I had for choosing what I did.

Finally, I have outlined the software architecture design which I employed when building the application and how it feeds into readable and easy maintainable code.

1.2 Project Background

According to (Hendicott, 2017), there are estimated to be 60,000 ringforts, sometimes referred to as fairy forts or Ráths, dotted around the Island of Ireland. These were mostly built during the early Christian period c. 500 – 1100AD and were erected as protective enclosures around farmsteads (Manning, 2004). Most of these are in rural Ireland, predominantly on farms where the farmer is the custodian. It is though that you are never more than a few kilometres from a Ringfort, yet most people are unaware of their presence.

My project aimed to build a mobile application which would make the location of these sites more accessible to the public. Importantly, it also allows for previously unmapped ringforts to be recorded. The app allows these new ringforts to be recorded in three ways: firstly, people who discover them locally can visit and record them on site; secondly people who want to search google satellite maps and discover them visually can record them this way. Finally, I have provided access to known locations from the National Monument Service database, which can be updated and saved to the app's live database of Ringforts.

As part of the requirements gathering process for the project, I asked for assistance from a local historian (who has a keen interest in Ringforts), with screen design and flow. From this point forward he will be referred to as the 'knowledge expert'. He provided a different prospective, and highlighted areas which he deemed important from the point of view of a potential user of this type of App.

1.3 Project Objectives

The objectives of this project are:

- To build a mobile application which is intuitive to use and provides users with important information on the Ringforts around Ireland.

- To allow crowd sourcing of new Ringfort locations by users who know of them personally and by users who like to discover new ones by searching satellite map imagery.

- To investigate the feasibility of using existing records from the National Monument Service database archives to enable known locations of existing sites to be used as a discovery tool. (Department of Housing Local Government and Heritage, 2019)

- To investigate the current tools and frameworks which allow for cross platform mobile app development using a single code base.

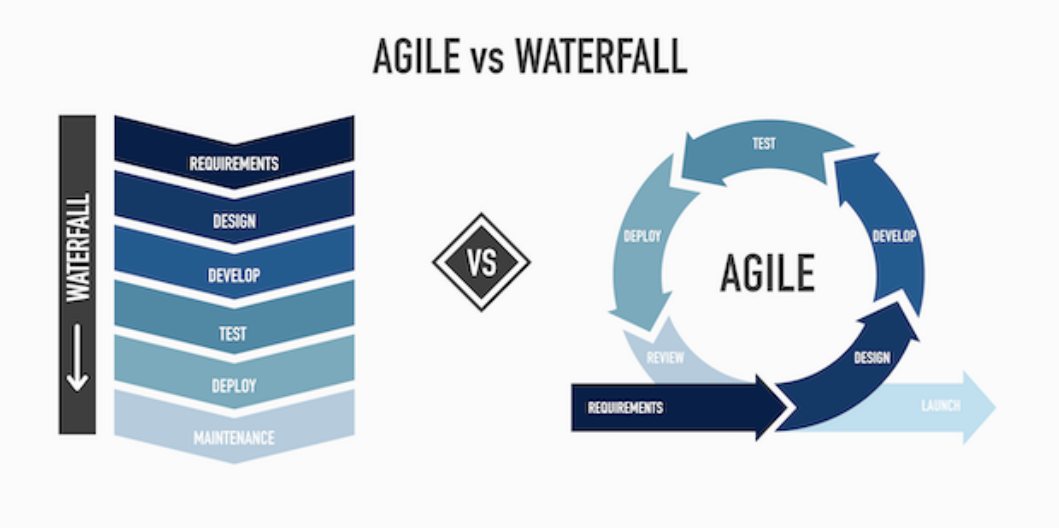
- To become proficient in a new technology and/or programming language not previously taught during the course.

1.4 Project Methodology

Before deciding on the methodology to use for the project, I explored the waterfall and agile/prototyping approaches. Waterfall is linear approach to software development where each phase is fully completed and signed off before the next phase is started. Therefore requirements would be fully gathered and signed off before design starts, and development would not start until design has completed.

Agile is an iterative approach where development is divided into small iterations called sprints. The output of each sprint should be a set of tested features which could be implemented to the live system. Each sprint should have a feedback loop into the next sprint so requirement changes can be easily absorbed into the development process. Prototyping fits well with the agile approach in that there is a working prototype of new features at the end of each sprint. Therefore, the knowledge experts can confirm early that what is being delivered meets their expectations.
Source: (Blake, 2021)

Figure 1-1 Waterfall versus Agile Process



Source: (Blake, 2021)

I chose Agile and Prototyping as my project methodology because they were best suited to delivering a solution within a limited timescale whilst allowing for changing requirements as the project progressed. I employed the following Agile practices:

- Writing requirements as User Stories based on conversations with the knowledge expert. The approach I used here is explored in section 2.

- Prototyping using both hand drawn screen layouts and wireframe designs created using an online mock-up tool.

- Breaking the project timeline down into the following 9 sprints over a period of 15 weeks.

    - Part 1 – This is three sprints of 2 weeks each. **(04.01.2022 – 13.02.2022)**
        - Sprint 1 – Preparatory Work
        - Sprint 2 – Analysis
        - Sprint 3 – User Stories, Development, and Interim Report

    - Part 2 – This again is three sprints of 2 weeks each. **(14.02.2022 – 27.03.2022)**
        - Sprint 4 – Development/Design
        - Sprint 5 – Development
        - Sprint 6 – Development

    - Part 3 – This is three sprints of 1 week each. **(28.03.2022 – 17.04.2022)**
        - Sprint 7 – Completing Build and Final Report
        - Sprint 8 – Prepare Demo Video
        - Sprint 9 – Prepare Presentation

- Incrementally building small testable features in each sprint which were demoed with the knowledge expert with the aim of getting feedback. This feedback was taken on-board in the subsequent sprints. I took highest priority stories into the next sprint, along with feedback from the knowledge expert about the previous sprint.

- See Appendix D for details of the original Sprint Plan which evolved as the sprints progressed based on re-planning which was carried out.

1.5 Conclusion

This chaptered has outlined the aims for the overall report. It then talks about the project objectives, and the project methodology used to achieved those objectives. It lists the Agile practices which I have employed in the project.

The next section will take a deep dive into the project requirements and how these were gathered and documented.

2 Project Requirements

2.1 Introduction to Requirements

After discussing my initial ideas about the mobile application with the knowledge expert and getting their thoughts and feedback, I put together some manually drawn screen layouts for the Mobile App. These hand drawings were then used during further discussions with the knowledge expert where they were tweaked as we discussed them. These prototype drawings enabled the knowledge expert to visualise the flow through the application and to better understand the data displayed and required at each point. See Appendix A for final version of the hand drawn screen prototypes. I then used an online tool from www.mockplus.com to build a wireframe prototype of the mobile app screens. See Appendix B .

The output from my discussions with the knowledge expert were used as a basis to write the functional requirements as user stories. User stories outcomes were written very much from the perspective of the user of the system. These stories split the apps functionality into discrete testable features. With the help of the knowledge expert, I was able to prioritise those in order of importance. See Appendix C for User Story details.

2.2 Functional Requirements

The following are the list of user stories which were specified at the start of the build, with estimates given in hours They are organised in priority sequence below, See Appendix C for User Story details including expected results.

Table 2-1 Functional Requirements

Title	Estimate
(1) Add a New Ringfort	7
(2) Add New Ringfort Location	5
(3) Add New Ringfort Images	4
(4) List all Ringforts	5
(5) Edit Ringfort Details	5
(6) Delete Ringfort Details	3
(7) Make Data Available to All Users	8
(8) Allow User to Signup	7
(9) Allow User to Login	4
(10) Allow User to Logout	2
(11) Limit Access until User Logged-in	5
(12) Add Multiple Images per Ringfort	8
(13) Searching the Ringfort List	8
(14) Searching the Ringfort Map	5
(15) Add a Ringfort as a Favourite	7
(16) Make Photos Available to all	5
(17) Set up Approval System	10
(18) Load and Display National Monument Data	12
Total	110

2.3 Non-Functional Requirements

These were the non-functional requirements which I deemed as important to the project and to the delivery of a successful mobile application experience:

- Performance - The application must be responsive such that the user is waiting for a minimal time.
    - As part of this, an initial splash screen was considered to distract the user while the application initially loads.
    - Responsiveness during application usage is key to its success, and caching of data was considered to improve this performance, especially network images.
- Scalability – The ability of the application to grow both its user base and the number of ringforts stored is a key part of the success criteria. Some important factors are:
    - Choosing the correct backend which can scale while keeping costs to a minimum.
    - Choosing a mapping solution which can handle growth in the user base while keeping costs at an acceptable level.
- Availability – Making sure that the application is available when the user expects is important. It must also be available on the device of their choice. Some of the areas considered were:
    - Making the application available on the 2 main mobile platforms, android and IOS. This was crucial to being able to see the project as a success.
    - Choosing a backend which had a high availability score was important.
- Security – Security is very important to users. Accordingly, keeping their login details secure was a must have. This was one of the reasons that Firebase Authentication was chosen.

3 Analysis of Technology Options

3.1 Introduction

This section looked at various options available across the technology stack for building cross-platform mobile applications. I compared them objectively in order to determine which would be best suited to this project. I broke the analysis into sections as follows:

1. Frontend technology
2. Backend/Database
3. Mapping Options

3.2 Frontend Technology Options

3.2.1 Introduction

The frontend technology choice was a hugely important decision and had a big impact on the direction of the project. One of the objectives was to make the mobile experience available on the main two mobile platforms; IOS and Android. Because of this, it was clear from the beginning that building native apps was not a viable option. This was mainly because of the duplicity of coding required and the learning curve of IOS development. I therefore investigated solutions which could be used to deploy native standard applications to both platforms from a single code base. I have compared the main contenders in this area: Flutter; React Native and Ionic. Source of comparison: (Strapi, 2021)

3.2.2 Flutter

Flutter, released by Google in 2018 is a comprehensive app SDK complete with widgets and tools which enables cross-platform development. It uses its own rendering engine and so controls every pixel on the screen. *To the underlying operating systems, Flutter applications are packaged in the same way as any other native app (Flutter, 2021).* By default it uses the material design screen style favoured by Google in Android apps. However, it also can be coded to use Cupertino design which would deliver an iOS style app or both.

The advantages:

- Faster development and Delivery – for example tools like “hot reload”.
- Quality documentation. https://flutter.dev/
- Feature Rich user interfaces
- Backward compatibility
- Comprehensive library of plugins. https://pub.dev/
- Very good performance, closest to Native performance

The disadvantages:

- Framework age means the online community support is not as mature.
- Project size – project files occupy more space than other tools.
- Required learning a new language – Dart.

3.2.3 React Native

React Native is an open-source app framework launched in 2015 by Facebook and can also be used to build cross platform apps using a single codebase. It is based on React so has basic concepts like JSX, Components, State and Props. But it uses native UI components rather than web components. These native components compile directly to Android views and IOS UIViews. So, a React Native app will look differently depending on the platform. *React Native communicates with the native APIs via a JavaScript bridge. The JavaScript bridge doesn't perform well for all development requirements (Suranga, 2021).*
The advantages:

- Platform specific Native UI compiled components
- Web and ReactXP libraries for possible web version of the app
- Almost on Par performance with Native apps
- Large community of users to assist with issues encountered

The disadvantages:

- Poor documentation
- Requires a lot of external 3rd party libraries
- Requires React Native Bridge to communicate between JSX and the native app layers which can sometimes slow performance.

3.2.4 Ionic

Ionic was first introduced in 2013 and is an open-source SDK for building cross-platform mobile apps. Ionic uses traditional web languages like HTML, CSS and JavaScript to build mobile apps. It is wrapped by the Cordova framework which will then render the app in a native WebView. Cordova plugins are also used to access native features like the camera or sensors. Examples of Apps are MarketWatch, Pacifica and Sworkit.
Source: (Strapi, 2021)

The advantages:

- Great for web developers who already know HTML, CSS and JavaScript.
- Library of component and plugins to handle UI, app icons, and device features like Bluetooth, GPS or the camera.
- Any frontend framework can be used to create the app starting from Ionic 4.

The disadvantages:

- Using WebView for large applications can result in a significant reduction in performance.
- Can lack some native plugin support.
- Security – developers will need to consider web securities as well as Native App security concerns.

3.2.5 Head-To-Head Comparison

The diagram below compares the technologies across several categories. (Rykowski, 2021)

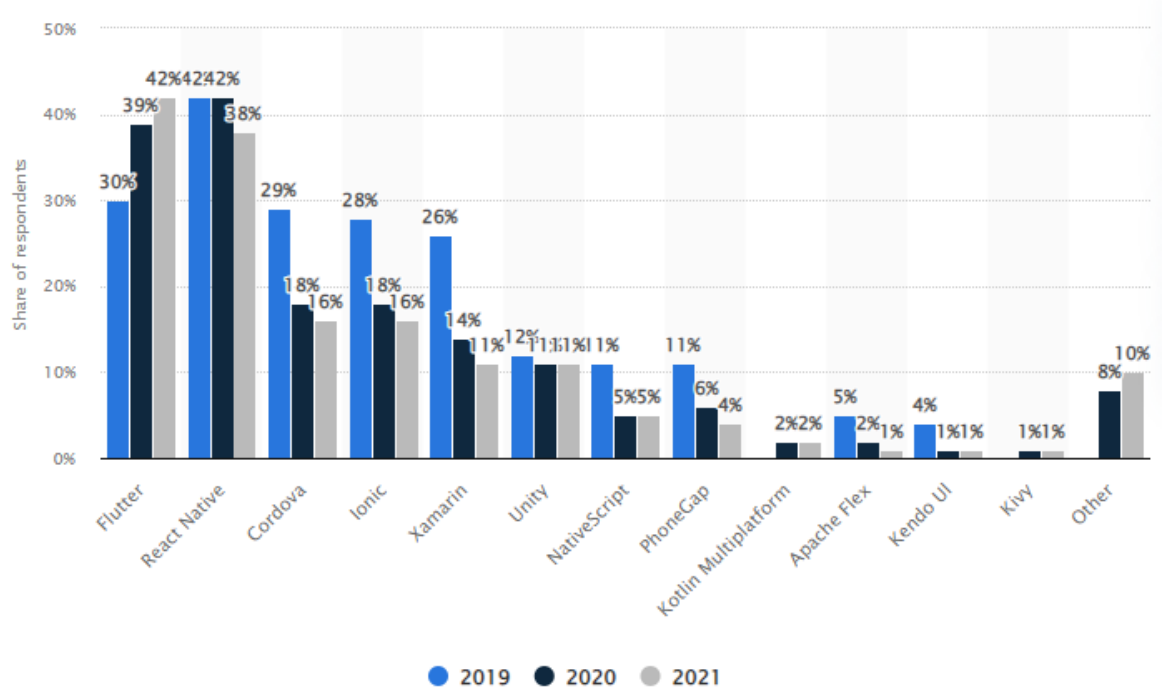
Table 3-1 Comparison of Flutter, React Native and Ionic

Criteria	Flutter	React Native	Ionic
Created by	Google	Facebook	Ionic
Official Release	December 2018	March 2015	November 2013
Programming Language	Dart	JavaScript/Typescript	HTML, CSS, JavaScript
Compilation	Compiled Native App	Partly Compiled Native App. JavaScript not compiled but run using a bridge	Not compiled to Native, but Web App wrapped in Native app.
UI Display	Flutter renders the screen itself using the frameworks graphics engine. So, by default the Android and IOS apps will be consistent across platforms. But can be customised where required.	UI Elements compile to Native Widgets so you get Native looking applications as default.	Styled as in any Web app, not compiled to Android or IOS.
Performance	Flutter has near Native performance. It is compiled Ahead-of-time in native machine language.	UI components get compiled to Native components, but the JavaScript isn't and runs in its own JavaScript thread. So, any interaction between JavaScript and to native components is through a bridge which can impact performance.	Performs the worse of the 3 in almost all scenarios.
Popularity	135k stars GitHub Jan22	101k stars GitHub Jan22	46k stars GitHub Jan22
Example App	Google Ads, Alibaba, ULike, Hamilton	Instagram, Uber, Messenger	Accenture, NHS, UNTAPPD
Documentation	Comprehensive and easy to understand.	Documentation not easy to use.	A lot of documentation available

3.2.6 Cross-Platform Mobile Frameworks Used Worldwide 2019-2021

Here we can see the steady increase in the usage of Flutter in the last 3 years and the stagnation and then decline in the market share of React Native and Ionic. Flutter is now the most popular Cross-platform mobile framework among Software Developers. *Source: (Vailshery, 2022)*

Figure 3-1 Comparison of Flutter, React Native and Ionic by popularity



Source: (Vailshery, 2022)

3.2.7 Conclusion

I chose Flutter as my Front-End Development Framework for several reasons:

- Flutter gives a richer more consistent cross-platform UI experience, which is what I wanted for this app.

- The fast development environment was a huge positive given the timescales of the development.

- Flutter Applications give the closest to Native performance of any of the options compared.

- There is great documentation and resources available. And community hubs like Flutter Community and Flutter Awesome are great for finding solutions.

- I wanted the opportunity to learn a new framework and programming language, one which is becoming increasingly popular. This allowed me to demonstrate to myself that one of the most important skills learned on the Higher Diploma course is the ability and confidence to research and learn new technologies.

3.3 Backend Technology

3.3.1 Introduction

The decision about which backend technology to use was another major project decision. For me, the decision came down to two technologies which I covered during the course. The first was a custom REST API built on Node.js and the second option was using Firebase backend services.

3.3.2 Custom Node API (Application Programming Interface)

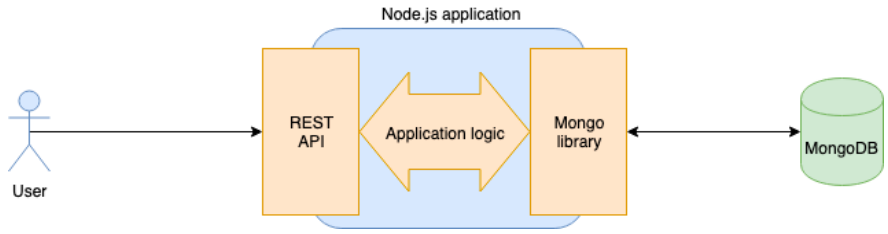
The first option, and the one I had initially planned to employ, was building a custom REST API on Node.js using the hapi framework. It would use a MongoDB NoSQL database with the help of the Mongoose library and use a service like Cloudinary to handle image storage. The advantages of this approach were - Source: (kella, 2021):

- The separation of the data service from the client allows for portability of the data service and re-use by other applications.
- The project has total control over reliability and scalability of the backend service.
- The data is cacheable which could be used to ease the burden on the server.

The disadvantages were - Source (kella, 2021):

- This requires a complete backend to be built from the outset including authentication of the endpoints. This is a large cost and time overhead for projects which wish to get off the ground quickly.
- Because reliability and scalability are project responsibilities, they will also become time and cost drains on the project.

Figure 3-2 Node JS API using MongoDB



Source: (Kamani, 2019)

3.3.3 Firebase Back End as a Service (BaaS)

The second option was to use the Googles Firebase suite of backend services, storing the App data on Cloud Firestore and images on Cloud Storage. This is a paid service but gives a free layer which should be enough for most apps initially. See Firebase Pricing in Appendix G (Firebase, 2019)
The advantages:

- Reliable and Extensive databases, including the Realtime Database and Cloud Firestone
- Free layer which might suffice for a lot of projects. See Appendix G.
- Google analytics works seamlessly with Firebase - unsurprisingly.
- Free Multi-Platform Authentication
- Very Scalable solution
- Great plugin and documentation available for Flutter to use the Firebase APIs
- Very quick to integrate into a new app.
- High Availability > 99.999 per month. (Google, 2020)
- Very secure.

Source: (Clark, 2021)

The disadvantages:

- No fixed price, rather a Pay as You Go approach which can be unpredictable
- Limited querying ability in the Realtime database which are resolved by Firestore DB.
- Doesn't provide tools for migration off Firebase to other platforms

Source: (Moqod, 2021)

3.3.4 Conclusion

I decided to proceed with the project using the Firebase BAAS both for my data needs and for the authentication for my application. This was based on the following factors:

- The integration with the Flutter SDK using the various plugins available and the documentation available both for Firebase and the Flutter plugins are excellent.

- The pricing plans make it very likely the application could exist without any cost using the Blaze plan (See Appendix G).

- The speed of development using Firebase services was a major factor in the decision. After the initial sprint planning (see Appendix D) it became apparent quite early that building the back end as a custom API was not feasible in the timeframe.

3.4 Mapping Technology

3.4.1 Introduction

This application relies on maps for part of its core functionality, so picking the best map solution was important. Here I evaluated some of the options available for mapping which could have been used. Comparison Source: (Dziuba, 2021)

3.4.2 MapBox

MapBox offers a set of comprehensive features to integrate into Mobile apps and websites. These include data visualization, navigation, and search functionality. It is used by giants like Facebook, Snapchat and Foursquare.
The advantages:

- Very customizable and flexible
- Fast load times and performance
- Offline mode in the API
- Open Source

The disadvantages:

- Inferior map coverage in certain areas
- Learning curve with MapBox API.
- Can be unpredictable pricing

3.4.3 OpenStreetMaps (OSM)

OpenStreetMaps is an open sourced, community-based project which supplied maps and data to many websites and apps. It's totally free but still can provide a high level of accuracy thanks to the efforts of volunteer map enthusiasts and developers who supply it with data.
The advantages:

- The API is free
- The is an open-source service.

The disadvantages:

- Limited number of queries. Excessive data exchange not welcome, and users may be blocked when making too many data requests.
- May require addition infrastructure services. Or it may be required to use MapBox which is based on OSM data.

3.4.4 Google Maps

Google Maps is the original pioneer of online mapping technology and boasts over 99% coverage of the world. It can leverage a fleet of satellites, Street View cars and Android devices as data sources.
The advantages:

- Very Recognisable on mobile and web
- Excellent global and local data quality
- Multi Language support

- Street View

The disadvantages:

- Less Customisation
- Not open source
- Can be unpredictable pricing

3.4.5 Conclusion

I chose Google Maps as the map solution for my App. This was based on the following:

- It provided all the functionality required in the App.

- I already had a google Map account with €200 euro credit a month which would cover my app in most scenarios. MapBox might have been slightly cheaper but both models have unpredictable pricing models.

- The Flutter plugin for Google maps is excellent and has very clear documentation.

4 Design/Analysis

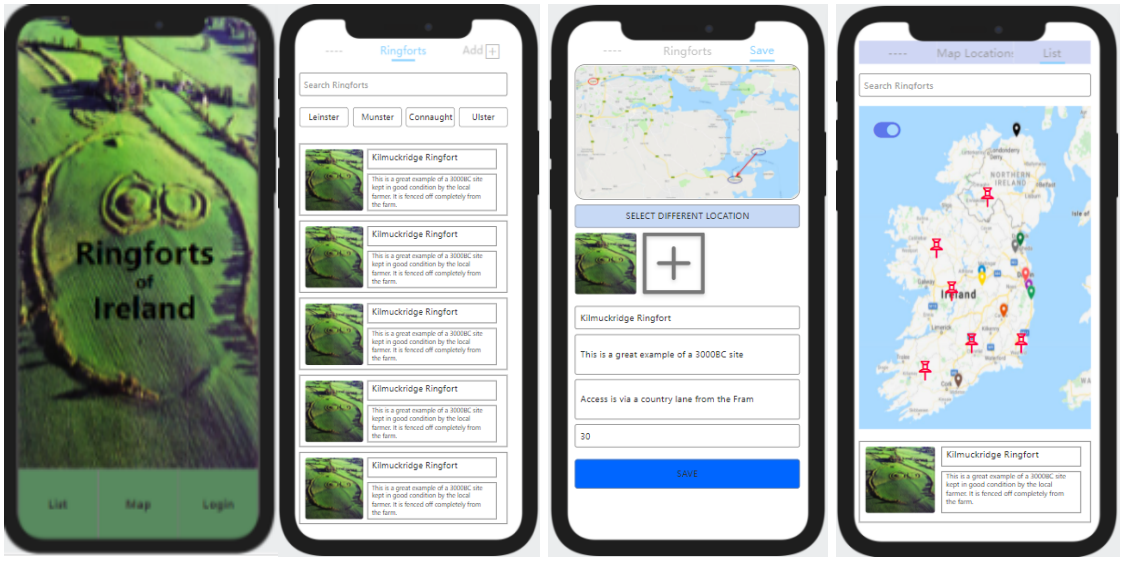
4.1 Introduction

The design of the screens was the area where the knowledge expert's feedback was of most benefit to me. I initially used hand drawn screens (see Appendix A) as an artifact to help draw out the requirements. These evolved into the wireframes below which were eventually agreed with the knowledge expert. I found that having something concrete like the screen drawings to use during discussions helped tease out what was important for a potential user of the application. It helped the expert more easily visualise how the app might work when built, and what data should be gathered and displayed.

4.2 Screen Mock-ups

These are the final wireframes agreed with the knowledge expert before the 1st development sprint. These allowed them to visualise what would be produced which lessened the probability of surprises after each sprint.

Figure 4.1 – Wireframes created before 1st sprint



4.3 Data Modelling

This section gives details of the data required by the application and how it will be modelled in the Firebase Firestore database (DB). There are the 4 Firestore collections being used by the application:

1. ***users*** *– This collection stores data about registered users of the system.*
2. ***historicSites*** *– This is a Firestore collection of the '**live'** ringforts on the system. This is the collection where the app stores the ringforts which users add to the system.*
3. ***historicStagingSites*** *– This is a collection of staging updates which are either awaiting approval by an admin user, or a history of those which have either been approved or rejected previously.*
4. ***NMS-Ringforts*** *– This collection stores ringfort records downloaded from the National Monument Services database of national Ringfort monuments. (National Monument Services, 2022)*

4.3.1 user Collection Model

This is the main 'user' data collection for the application. A new user document is created when a new user registers on the App. This document defines the user on the system. The favourites array will be used store the 'uid' of the users favourited ringfort sites. The adminUser Boolean field will indicate if the user has administrator rights on the data, with the ability to make instant updates to data and to approve/reject updates from other non-admin 'normal' users.

Figure 4.1 – user Model

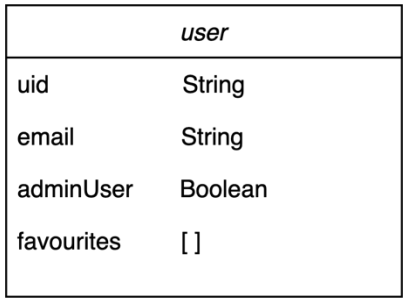
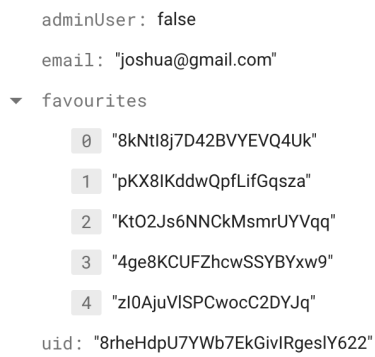


Figure 4-1 Sample user Collection Data



4.3.2 historicSites Collection Model

This is the main 'live' data collection for the application. It defines a ringfort in the application. When a user adds a new ringfort within the app it gets added to this collection. The siteAddress, siteProvince and siteCounty are fields which get generated automatically based on the latitude and longitude chosen. This collection is the data which drives the List Ringforts and Map Overview screens.

Figure 4-2 historicSite Model

historicSite	
uid	String
siteName	String
siteDesc	String
siteAddress	String
siteCounty	String
siteProvince	String
siteAccess	String
siteSize	double
latitude	double
longitude	double
image	String
lastUpdatedBy	String
createdBy	String

Figure 4-3 Sample historicSite Collection Data


```
address: "V6Q8+Q9 Incharue, County Kerry, Ireland"
county: "County Kerry"
createdBy: "uNH5vYQsqidNUaX23iglYjSQYrF3"
image: "https://firebasestorage.googleapis.com/v0/b/ringforts-
        ireland.appspot.com/o/images%2Fimage-0ML3vJHiFTy7GcjZrKAv.jpg?
        alt=media&token=82175dd6-42d4-4147-a977-49df9bf3c2fd"
lastUpdatedBy: "chHXJDpEXrTd4eLKq5387vaTWwT2"
latitude: 51.88939322459631
longitude: -9.784055426716805
province: "Munster"
siteAccess: "Yes"
siteDesc: "This is a great extract of a Ringfort in Kerry"
siteName: "Kerry Ringfort"
siteSize: 32
uid: "0ML3vJHiFTy7GcjZrKAv"
```

4.3.3 historicSitesStaging Collection Model

This is a staging collection for changes made by ‘normal’ users to ringfort records. When a non-admin user makes updates, i.e., adding a new Ringfort, deleting a Ringfort, or updating a Ringfort, the changes are stored in this collection with a status of ‘awaiting’. An admin user can then approve or reject this change at which point the changes get added to the ‘live’ collection historicSites if approved.

Figure 4-4 historicSitesStaging Model

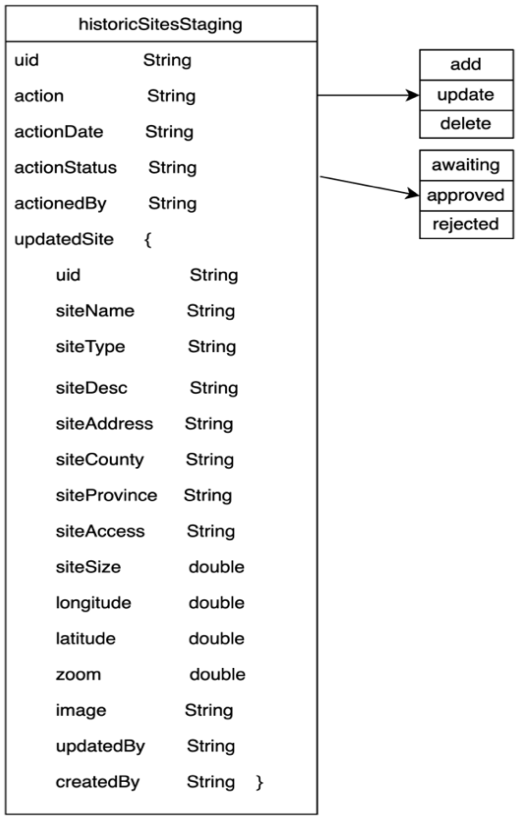


Figure 4-5 Sample historicSitesStaging Collection Data


```
action: "add"
actionDate: 26 February 2022 at 16:07:03 UTC
actionStatus: "approved"
actionedBy: "8rheHdpU7YWb7EkGivIRgeslY622"
uid: "9EuPaiSuAHNVW4TRqqvy"
updatedSite
  address: "FPRH+CP Killincooley Beg, County Wexford, Ireland"
  county: "County Wexford"
  createdBy: "8rheHdpU7YWb7EkGivIRgeslY622"
  image: "https://firebasestorage.googleapis.com/v0/b/ringforts-
      ireland.appspot.com/o/images%2Fimage-
      9EuPaiSuAHNVW4TRqqvy.jpg?alt=media&token=c8ce7760-ad10-
      4e90-b27c-53a658dbf05c"
  lastUpdatedBy: "8rheHdpU7YWb7EkGivIRgeslY622"
  latitude: 52.4910083
  longitude: -6.2707267
  province: "Leinster"
  siteAccess: "No"
  siteDesc: "Test Description is added here"

siteName: "Test"
siteSize: 32
uid: "9EuPaiSuAHNVW4TRqqvy"
```

4.3.4 NMS-Ringforts Collection Model

This is a data collection populated with an extract from the National Monument Service database of historical sites. The data has been filtered before being uploaded to include only ringforts sites. Approximately 24000 records have been added using an extract from NMS using the tools at this site: https://data.gov.ie/dataset/national-monuments-service-archaeological-survey-of-ireland , (Department of Housing Local Government and Heritage, 2019). They were loaded to this Firestore collection from a csv file using a service from FireFoo application. (FireFoo, 2022)

Figure 4-6 NMS-Ringforts Model

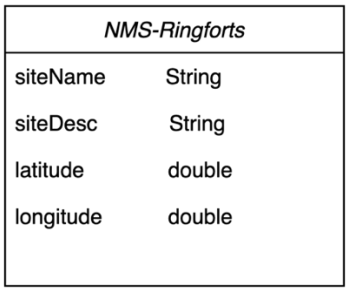


Figure 4-7 Sample NMS-Ringforts Collection Data


```
latitude: "52.392667"
longitude: "-6.586131"
siteDesc: "CORLICAN"
siteName: "Ringfort - rath"
```

4.4 Process And Data Flow

4.4.1 Introduction

This section shows the main process and data flows for the application. It outlines the different ways a user can add or update ringforts, and how the process is different depending on which route is taken within the application.

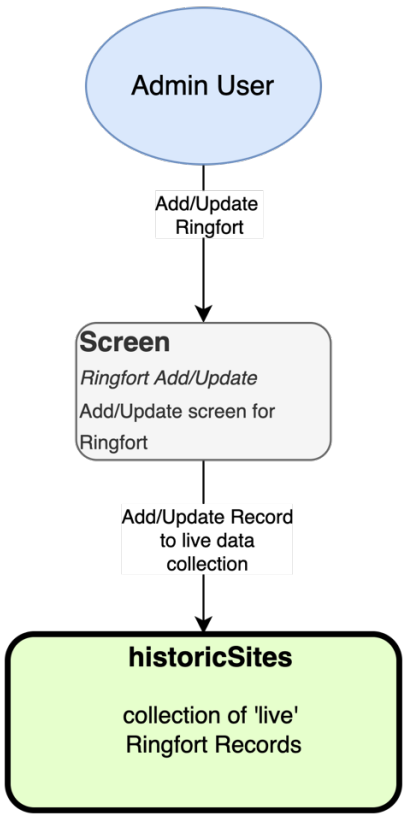
Scenarios

1. An Admin user adds or updates a Ringfort. No approval is required for this change as the user has admin rights.
2. A normal access user adds or updates a Ringfort. This needs to be approved by an Admin user.
3. The National Monument Service Data (NMS) map overview screen is used to select a ringfort to add to the 'live' system. This is done by a normal access user so requires approval.

4.4.2 Admin User Adds/Updates a Ringfort directly via the Add/Update Screen

This scenario shows how data flows when an *admin* user adds or updates a Ringfort via the Add or Update screen. This updated or new Ringfort data will get added directly to the live '*historicSites'* data collection. There is no approval or staging process required for admin users.

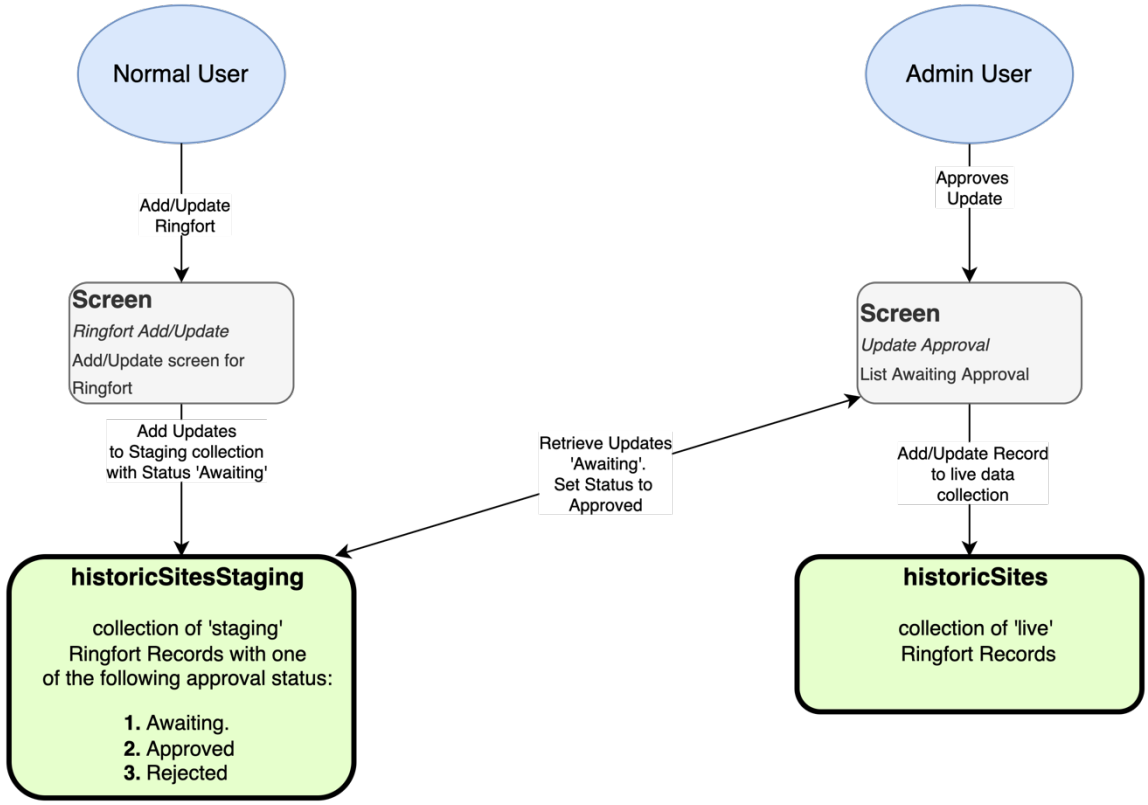
Figure 4-8 Admin User Add/Update Process Flow



4.4.3 Normal User Adds/Updates a Ringfort directly via the Add/Update Screen

This scenario shows how data flows when a *normal* user adds or updates a Ringfort directly via Add or Update screen. This updated or new Ringfort data will get added to the staging 'historicSitesStaging' collection with a status of 'awaiting' approval.
The *admin* user can use the approval screen to view all changes awaiting approval and to either approve or reject. Once approved the new or updated data is added or updated on the live '*historicSites'* data collection. The staging table gets its status updated from 'awaiting' to either 'approved' or 'rejected'. If rejected, then the data is not added to the 'live' data collection.

Figure 4-9 Normal User Adds Ringfort directly from the Add Screen



4.4.4 Normal User Adds a Ringfort via the NMS Map Screen

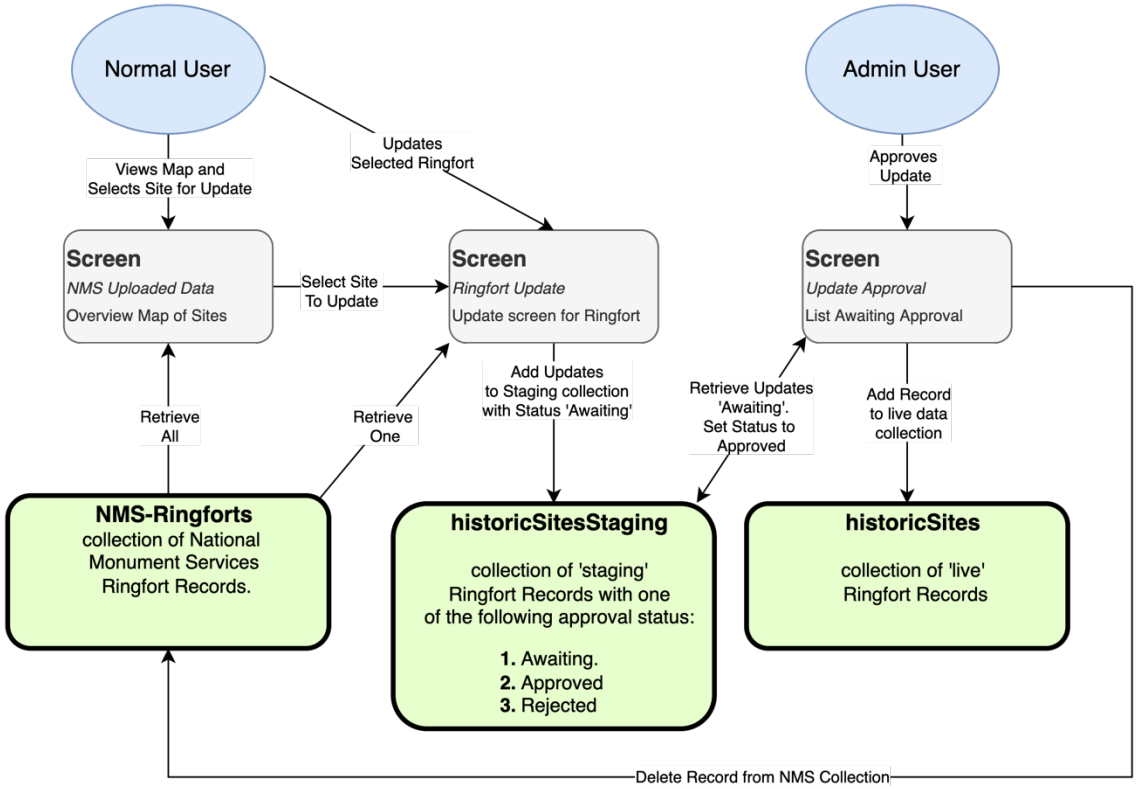
This scenario shows how data flows when a *normal* user adds a new Ringfort via the NMS data map overview screen. This screen shows ringfort locations on a map using data from the 'NMS-Ringforts' collection. This collection was populated from a download of data from the National Monument Services website.

On the map a user can then select the Ringfort they want to add. This triggers the update screen to launch with the selected Ringfort's location and some details pre-populated on it. The user then completes the updates and saves it. This triggers the data to be sent to a staging '*historicSitesStaging'* collection for approval by an admin user.

The *admin* user can use the approval screen to view all changes awaiting approval and to either approve or reject. Once approved the ringfort is added to the live '*historicSites'* data collection and deleted from the NMS data collection. The staging table gets its status updated from awaiting to either approved or rejected. If rejected, then the data is not added to the 'live' data collection.

Using this process users can systematically move data from the NMS-Ringforts collection to the apps 'live' historicSites collection while adding useful and consistent data.

Figure 4-10 Normal Users Adds Ringfort via NMS Map Screen



4.4.5 Overview of All Possible Screen flows.

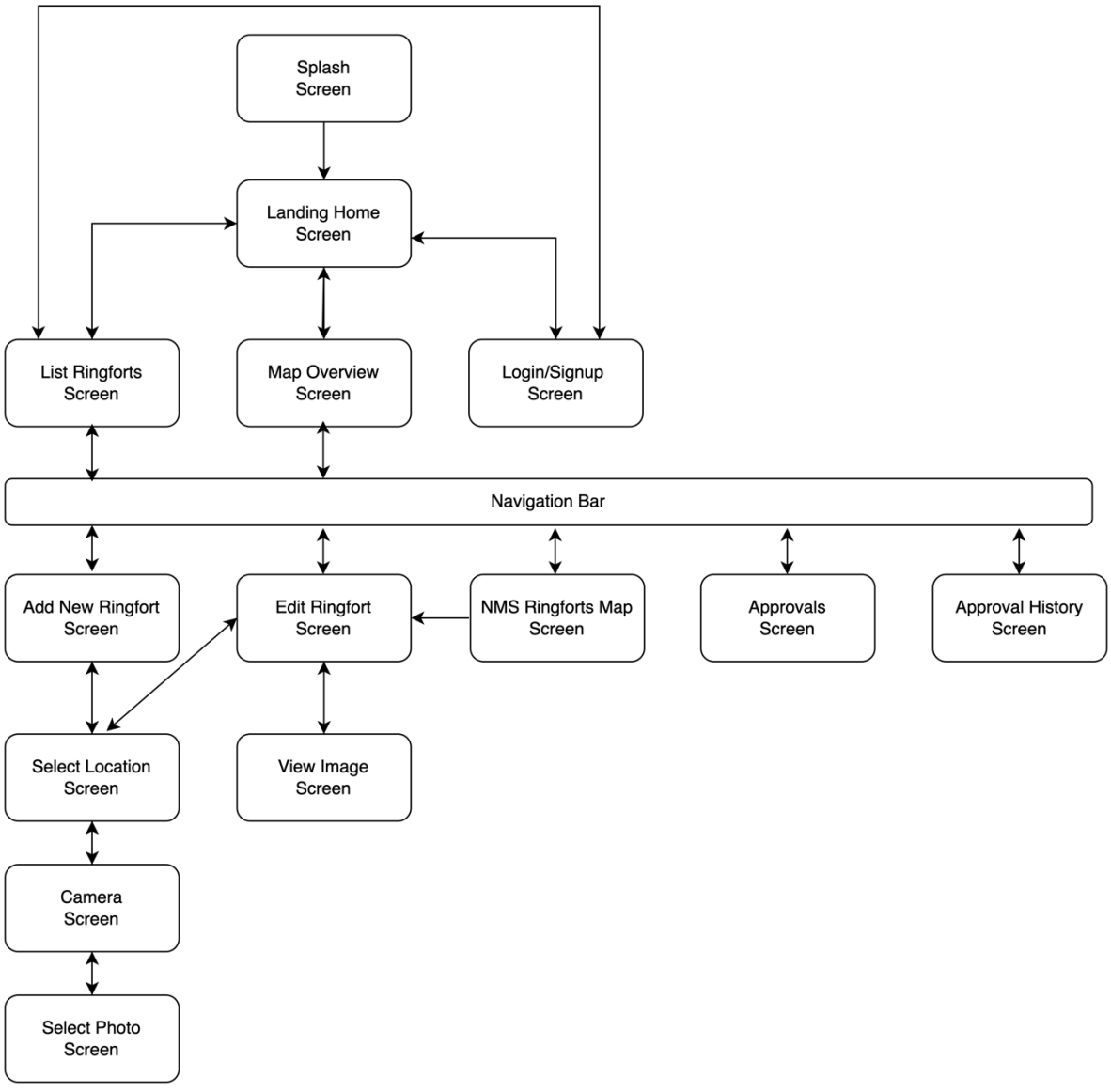
When the App initially starts, the splash screen will show briefly and then the app proceeds to the main landing home screen. From here three options are available: the List Ringfort Screen, the Map Overview Screen, and the Login/Signup Screen.

From List Ringforts and Map Overview screens the main functionality of the application is exposed, either from the Navigation Bar or the App Bar.

The Nav Bar provides access to: 1) the NMS Map Screen, 2) the Approvals screen for an Admin user 3) the Approval History Screen for the non-admin user and 4) the Add Ringfort Screen for both.

The Add and Edit screens provide access to further screens to enable selection of the Ringfort location, taking a picture with the camera or accessing images already on the device.

Figure 4-11 Screen Flow Overview

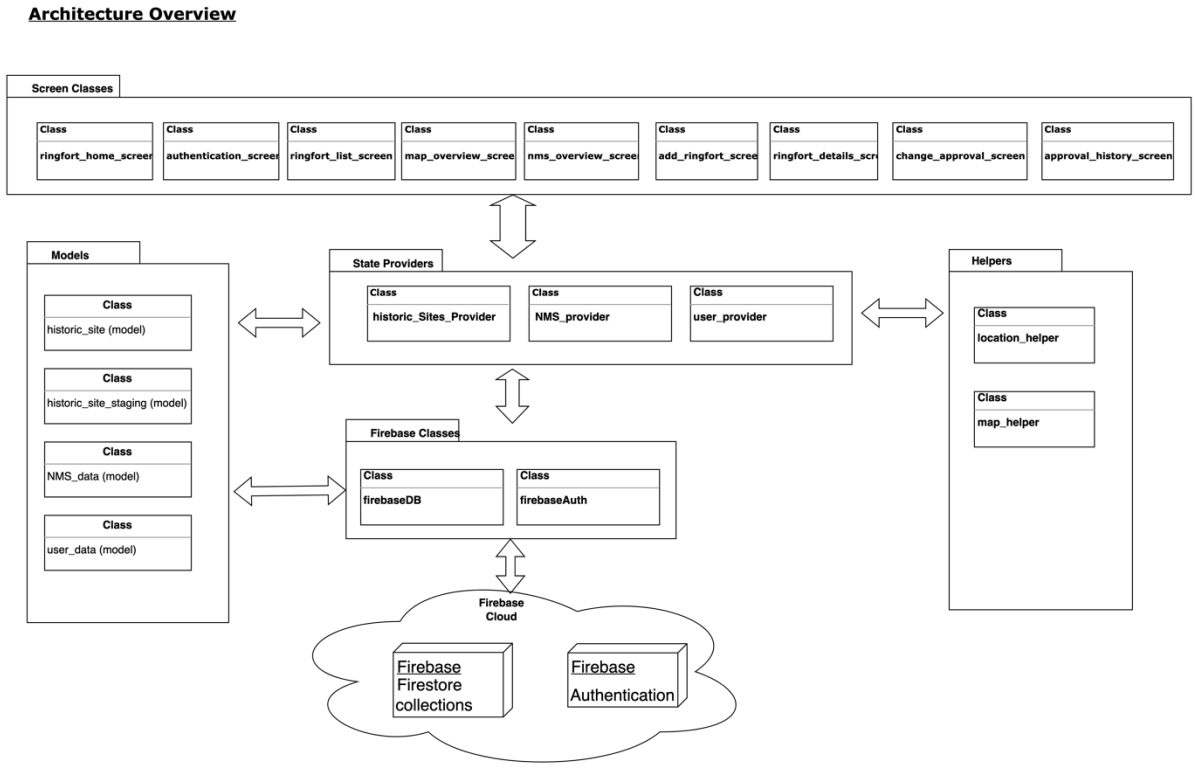


4.5 Source Code Structure

4.5.1 Introduction

I have broken the code into different libraries to help with maintainability and readability. I have kept as much logic out of the screen classes as possible and included the logic in the state management provider classes. Here is an overview of the architecture design of the application

Figure 4-12 Architecture Overview Diagram



4.5.2 Firebase Access Classes

The state management classes use these firebaseDB classes to give them access to the Firestore methods. The methods in the firebaseDB class provide add; update; delete and query access to the Firestore collections. The firebaseAuth class handles registration and authentication of users.

Figure 4-13 Firebase Classes

4.5.3 General Helper Classes

The location_helper class handles location tasks like getting the static satellite image for a given location. It also handles retrieving the current address for a chosen position, including the county and province for the location. While the map_helper class calculates the boundary needed by google maps to focus the maps to include all the selected markers.

Figure 4-14 Helper Classes

4.5.4 Data Model Classes

These classes define the data models for the Firestore collections. They define methods to convert data from dart classes to JSON maps and vice versa to help manage data to and from Firestore.

Figure 4-15 Model Classes

4.5.5 State Management Provider Classes

These classes use a package from Flutter called 'provider' to implement state management in the app. These classes are implemented using 'with ChangeNotifier' syntax which will provide access to methods from the 'provider' package's ChangeNotifier class without inheriting from it. Other classes can then listen for updates to fields in these provider classes.

Figure 4-16 State Management Provider Classes

For example, the 'historic_sites_provider' class has a field call _filteredSites which is a List of HistoricSites objects (Ringforts). The class also has a method setFilteredSites(), which updates this field based on a few parameters: search string, _showFavourites Boolean and _showLocal Boolean. The field and method can be used in the screen classes as required. The class variables can be listened to for updates, by the screen class. For example, when the favourites chip is selected to show the users favourite Ringforts, this method is called with _showFavourites = true. This will update _filteredSites to show the users favourite sites. The screen will automatically update as the screen class is listening to updates in this variable.

```
setFilteredSites(String searchQuery, bool _showFavourites,
      bool _showLocal, LatLng currentLocation, UserData userData)
```

4.5.6 Screen Classes

These classes define the screens in the application. Each of these classes has a build method which returns a Widget. The Widget returned is usually a hierarchical tree of Widgets. A very simple example might be that a **Text** widget is used to display text on the screen, and this might be wrapped in a **Padding** Widget in order to add padding, which might be wrapped in a **Container** Widget to define width and height and then wrapped in a **Scaffold** Widget which defines a screen.

Figure 4-17 Screen Classes

4.5.7 Widget Classes

To make the code more readable and maintainable, and to enable code re-use, it sometimes makes sense to define a part of the screen in its own class. For instance, below I defined what I want a favourite icon to look like in the *favourite_icon* class below. I defined input parameters and defined it as stateful so that it can react to user interaction on the screen. I can then use this like any Flutter defined widget in my screen classes or in other user defined widgets classes. I use this *favourite_icon* widget in my *ringfort_card* widget class. This is so that the icon appears in each ringfort card in the list and responds to being pressed by turning red or clear, and by calling a function to update the user's favourites on the database.

Figure 4-18 Widget Classes to define an UI element

Figure 4-19 Sample Code from the favourite_icon widget

```
Widget build(BuildContext context) {
  return Consumer<UserProvider>(
    builder: (context, userData, child) => GestureDetector(
      child: CircleAvatar(
        backgroundColor: Colors.white,
        child: userData.userFavourites == null
            ? Container()
            : userData.userFavourites.any((fav) => fav == widget.ringfortUID)
                ? Icon(
                    Icons.favorite,
                    color: Colors.red,
                  ) // Icon
                : Icon(
                    Icons.favorite_border,
                    color: Colors.red,
                  ), // Icon
      ), // CircleAvatar
      onTap: () async {
        if (userData.userFavourites.any((fav) => fav == widget.ringfortUID)) {
          await Provider.of<UserProvider>(context, listen: false)
              .removeFavouriteFromCurrentUser(widget.ringfortUID);
        } else {
          await Provider.of<UserProvider>(context, listen: false)
              .addFavouritetoCurrentUser(widget.ringfortUID);
        }
      },
    ), // GestureDetector
  ); // Consumer
}
```

Consumer – This listens for any changes in state management class UserProvider's field 'userData'.

GestureDetector generates a widget which can be used to detect user interaction like tapping. It's called by the method passed in the builder function of Consumer.

If userData's userFavourites List is null, then I don't display the heart icon

If the ringfort UID passed into this class is in the favourites List, then the heart icon is filled with red. Otherwise it's not filled.

The onTap callback will trigger the remove from favourites method if its already a favourite or will trigger the add to favourites if it's not already a favourite. These methods are available as part of the UserProvider state management class.

4.5.8 Main Class

The main dart class is where the app is controlled from, and where the set-up is defined.
I used a package from Flutter called MultiProvider to allow me to use multiple state management provider classes. I have used the following Provider State Management classes.

1. StreamProvider<User> - for the firebase authentication state of a user.
2. ChangeNotifierProvider<HistoricSitesProvider> - for list of Ringforts
3. ChangeNotifierProvider<NMSProvider> - for list of NMS Ringforts
4. ChangeNotifierProvider<UserProvider> - for details on current logged on user.

The Application 'home' is defined as the RingfortHomeScreen class which is the entry point to the app when it initially starts.
And the app 'routes' define all the possible screen routes available.

Figure 4-20 Main Class Code

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        // The StreamProvider will keep the user authentication status
        // up to date across all widgets that consume it.
        StreamProvider<User>(
          create: (context) => FireBaseAuth.getUserStatus(),
          initialData: null,
        ), // StreamProvider
        ChangeNotifierProvider<HistoricSitesProvider>.value(
          value: HistoricSitesProvider(),
        ), // ChangeNotifierProvider.value
        ChangeNotifierProvider<NMSProvider>.value(
          value: NMSProvider(),
        ), // ChangeNotifierProvider.value
        ChangeNotifierProvider<UserProvider>(
          create: (_) => UserProvider(),
        ), // ChangeNotifierProvider
      ],
      child: MaterialApp(
        title: 'Ringforts of Ireland',
        theme: ThemeData(
          primarySwatch: Colors.green,
          primaryColor: Colors.grey,
        ), // ThemeData
        home: RingfortHomeScreen(),
        // Routing table for the app screens
        routes: {
          //Route - Ringfort Home screen
          RingfortHomeScreen.routeName: (context) => RingfortHomeScreen(),
          //Route - add a ringfort screen
          AddRingfortScreen.routeName: (context) => AddRingfortScreen(),
          //Route - Ringfort List screen
          RingfortsListScreen.routeName: (context) => RingfortsListScreen(),
          //Route - Ringfort Detail screen
          RingfortDetailScreen.routeName: (context) => RingfortDetailScreen(),
          //Route - AuthScreen
          AuthenticationScreen.routeName: (context) => AuthenticationScreen(),
          //Route - Map Overview screen
          MapOverviewScreen.routeName: (context) => MapOverviewScreen(),
          //Route - Update Approval screen
          ChangeApprovalScreen.routeName: (context) => ChangeApprovalScreen(),
          //Route - Approval Detail screen
          ApprovalDetailScreen.routeName: (context) => ApprovalDetailScreen(),
          //Route - User Approval History screen
          ApprovalHistoryScreen.routeName: (context) => ApprovalHistoryScreen(),
          //Route - Display Image screen
          DisplayImageScreen.routeName: (context) => DisplayImageScreen(),
          //Route - NMS Map Overview screen
          NmsOverviewScreen.routeName: (context) => NmsOverviewScreen(),
        },
      ), // MaterialApp
    ); // MultiProvider
```

5 Development Phases

5.1 Introduction

The next sections describe what was included in each sprint of the project. For the full original sprint plan see appendix D. I have used the Trello tool to manage my sprint progress and Appendix H shows the sprint progress for the first 6 sprints. After each sprint the progress to date and tasks for the next sprint were reviewed. Based on my available time for the next sprint and progress in the previous one, I carried out re-planning of the tasks to be included based upon priority. See appendix E and appendix F for details of sprint replanning at 01.03.2022 and the 14.03.2022 respectively.

5.1.1 Sprint 1

This start-up sprint was centred around determining the best Front end technology framework to use for building the app. Once Flutter/Dart were picked, the next task was to become proficient using this technology. This involved reading the Flutter documentation, taking online tutorials and courses, and building a few small demo Flutter Apps to prove the concept.

Table 5-1 Sprint 1 Details

Sprint No:	1
Start Date	04.01.2022
End Date	16.01.2022
Tasks	1) Compare and Select Front End Technologies (4 hours) 2) Study selected Front End Technology to become competent (25 hours) TOTAL: 29 hrs
Status	Complete

5.1.2 Sprint 2

This sprint saw the start of the requirements gathering process. At this point I spoke about the project with the local historian and based on that drew sample screens for discussion and update. His biggest priority while discussing the screen design, was making the user experience as intuitive and simple as possible. We also discussed what data should be captured by the App and how it would be displayed to the user. During this sprint, I did more study on the Flutter front technology and practiced on some small apps to give me confidence that I had the skills necessary to successfully build the Ringforts App.

Table 5-2 Sprint 2 Details

Sprint No:	2
Start Date	17.01.2022
End Date	30.01.2022
Tasks	1) Contact User (Local Historian) and discuss requirements (1 hour) 2) Mock up sample screens by manually drawing and confirm with user (4 hours) 3) Study selected Front End Technology to become competent (12 hours) 4) Start project report layout and structure (3 hours) 5) Document design of Data model (3 hours) TOTAL: 23
Status	Complete

5.1.3 Sprint 3

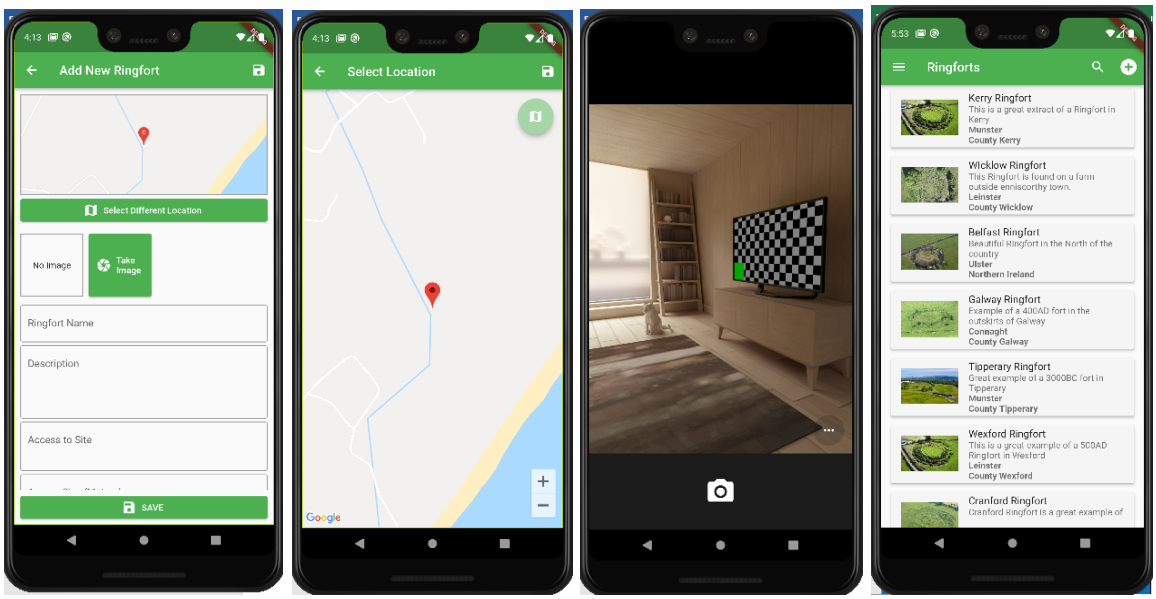
This 3rd sprint involved getting the user requirements written up into user stories and taking the manual screen drawings and mocking them up using an online tool. The first of the 'development' user stories were taken into this sprint which resulted in a working "Add Ringfort" screen with map location and images being captured. At this point the data captured by the App was being stored locally on the device. The interim report was also due at the end of this sprint.

Table 5-3 Sprint 3 Details

Sprint No:	3
Start Date	31.01.2022
End Date	13.02.2022
Tasks	1) Write user Stories for highest priority tasks (3 hour) 2) Turn screen drawings into Wireframes (4 hours) 3) Write project report to interim report level (10 hours) 4) User Story 1 Dev (7 hours) – (Add a New Ringfort) 5) User Story 2 Dev (5 hours) – (Add New Ringfort Location) 6) User Story 3 Dev (4 hours) – (Add New Ringfort Images) 7) User Story 4 Dev (5 hours) – (List all Ringforts) TOTAL: 38
Status	Complete

Here are the results of the first development sprint which was the first iteration of the 'Add Ringfort' screen, which triggers a Map location selection screen and the camera app.

Figure 5-1 Screens Developed in Sprint 3



5.1.4 Sprint 4

Based on the progress in sprint 3, this sprint was re-planned. It was predominantly a development sprint, but includes the design of the login and Signup screens and the continuation of the Project Report. This sprint also focused on hooking up the App to the Firestore backend collections and adding authentication using Firebase Auth services.

Table 5-4 Sprint 4 Details

Sprint No:	4
Start Date	14.02.2022
End Date	27.02.2022
Tasks	1) Continue Project Report (2 hour) 2) User Story 5 Dev (5 hours) – (Edit Ringfort Details) 3) User Story 6 Dev (4 hours) - (Delete Ringfort Details) 4) User Story 7 Dev (5 hours) - (Make Data Available to All Users) 5) User Story 8 Dev (5 hours) - (Allow User to Signup) 6) User Story 9 Dev (4 hours) - (Allow User to Login) 7) User Story 10 Dev (2 hours) - (Allow User to Logout) 8) User Story 11 Dev (5 hours) -(Limit Access until User Logged-in) TOTAL: 32
Status	Complete

Here are the results from this development sprint. This sprint focused on authenticating users with the Firebase Auth SDK for Flutter. It included a landing screen for the app and also locked down some functionality based on whether a user was logged into the app.

Figure 5-2 Screens Developed in Sprint 4

5.1.5 Sprint 5

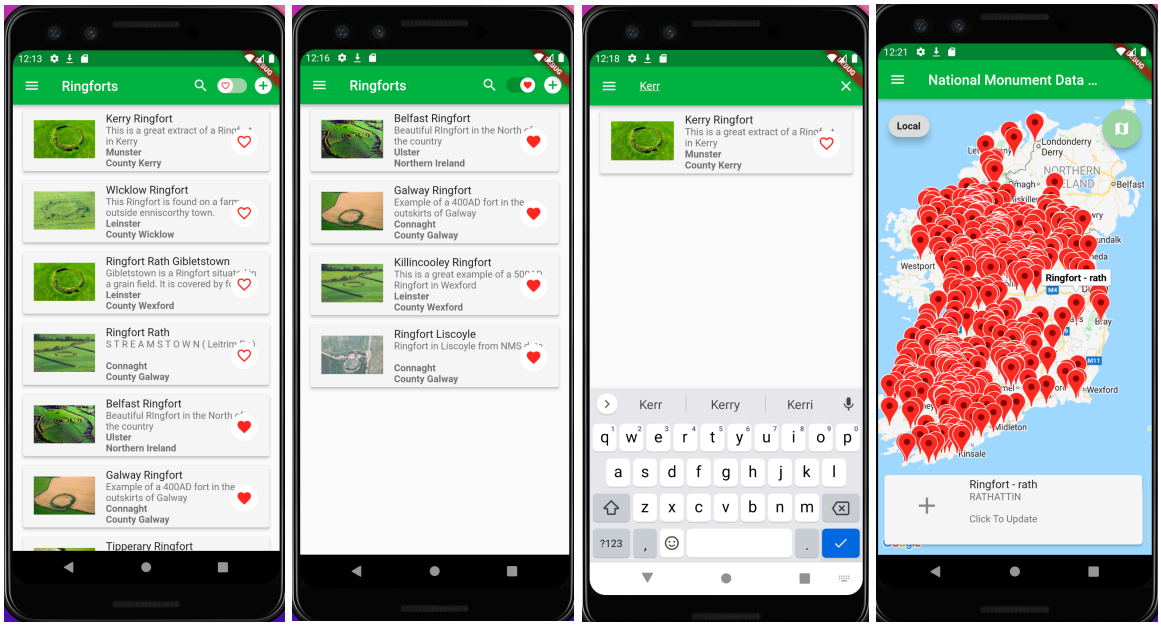
Based on the progress in sprint 3 and 4, this sprint was also replanned. I was able to move most of the outstanding development tasks into this sprint as progress was ahead of schedule and I was able to allocate more of my time to the sprint. This sprint dealt with search and map functionality but also included the loading of the national monument service data to a new collection on Firestore. Another major part of the sprint was the approval system, whereby normal users' updates need to be approved by admin users. The app was installed on my personal IOS iPhone in this sprint.

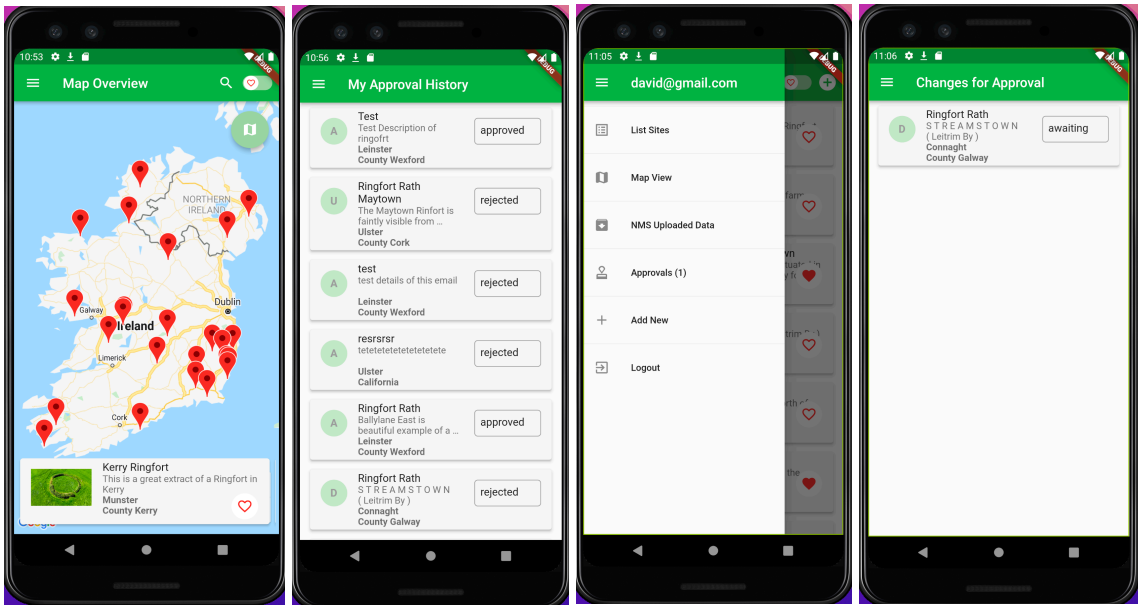
Table 5-5 Sprint 5 Details

Sprint No:	5
Start Date	28.02.2022
End Date	13.03.2022
Tasks	1. Continue Project Report (5 hour) 2. User Story 16 Dev (5 hours) (Make Photos Available to all) 3. User Story 13 Dev (6 hours) (Searching the Ringfort List) 4. User Story 14 Dev (4 hours) (Searching the Ringfort Map) 5. User Story 15 Dev (7 hours) (Setting Ringfort as User Favourite) 6. User Story 18 Dev (12 hours) (Adding NMS data and displaying on Map and allow to update) 7. User Story 17 Dev (10 hours) (Approval System for Normal users) 8. Install XCode on apple IMAC and build the project for IOS. (4hrs) TOTAL:52
Status	Complete

Here are the results from this sprint (which was predominantly development). This was also the longest sprint from a time spent point of view. The focus was on mapping and searching of the Ringforts List and the Map screens. It also included the approval process implementation. It added the favourite user functionality and the chip to select these favourites.

Figure 5-3 Screens Developed in Sprint 5





5.1.6 Sprint 6/7/8/9

Sprint 6 was the last one with development tasks, while sprints 7, 8 and 9 focused on the other deliverables of the project, for example: this report; the project showcase; the demo video; and the presentation.

Table 5-6 Sprint 6 Details

Sprint No:	6
Start Date	14.03.2022
End Date	27.03.2022
Tasks	1. Get Handbook Entries ready (10 hours) 2. Complete self-reflection in report (6 hours) 3. User Story 12 Dev (8 hours) (Display image on new screen - zoomable) 4. Bug fixes and commenting (4 hours) TOTAL: 28
Status	Complete

Sprint 7 was re-planned because a deadline extension given. It was extended to 2 weeks which brought me to the report deadline date.

Table 5-7 Sprint 7 Details

Sprint No:	7
Start Date	28.03.2022
End Date	15.04.2022
Tasks	NOTE: 2+ week sprint due to extension of deadline dates 1. Complete Project Report 2. Fix any code bugs and see if any small additions possible. 3. Complete README
Status	Complete

Sprint 8 was also re-planned and will start on the 16th April after this report has been completed and submitted.

Table 5-8 Sprint 8 Details

Sprint No:	8
Start Date	16.04.2022
End Date	24.04.2022
Tasks	NOTE: 1 Week Sprint 1. Prepare Demo Video
Status	Complete

Sprint 9 was also re-planned and will start on the 25th April after the Demo video has been completed and submitted.

Table 5-9 Sprint 9 Details

Sprint No:	9
Start Date	25.04.2022
End Date	02.05.2022
Tasks	NOTE: 1 Week Sprint 1. Prepare Presentation
Status	Complete

6 Critical Self Analysis

6.1 Introduction

This section of the report is my personal reflection on the project, and will provide examples of: my learning during the project; what I achieved during the project; problems I encountered and how I resolved them. I will also outline future plans for possible additional features to the app which were not possible within the timelines of the project.

6.2 What I learned

6.2.1 Flutter SDK And Dart Programming Language

One of the challenges which I set myself for this project was to learn a new skill. This was to demonstrate that this course has given me the ability and confidence to independently learn a new programming language and development environment. There was a 4 or 5 week lag at the start of the project where I was studying the Flutter SDK and the Dart programming language. Consequently, at the time I began worrying that I was making no actual progress on the project itself. However, I found that by sticking to my plan, that once the development sprints started, I was able to make progress quickly because of the time I had spent on learning.

I took a risk by building this project in a technology which was completely new to me, but I think this has paid off. The reason I took this risk was because I had an interest in exploring cross platform app development. Early in studying Flutter and Dart, I had some doubts as to whether I would be able to successfully deliver the plans I had for the app. This was because I was in unchartered territory without formal guidance or support as they were off-course technologies. In hindsight I should have regularly assessed the feasibility of the approach and considered if it was better to proceed or revert to a single platform approach with a course taught technology.

6.2.2 Agile Development, Project Planning and Re-planning

The decision to choose Agile as the project methodology, in my opinion, proved to be correct. I learned that this methodology is good for handling uncertainty in a project because the iterative approach means that requirements are continuously reviewed, and development is in discrete sections which allows regular validation against expectations. Changes can easily be catered for with this development approach. And while I planned out all the sprints at the start of the project, these plans were in no way set in stone, and at the end of each 2 week sprint there was an opportunity to review and re-plan if required. The knowledge expert played an integral part of this review and re-plan process based on their feedback of the working prototype produced at the end of each sprint.

6.2.3 Report Writing

I am most comfortable when deep in the middle of code, trying to solve a problem or resolve a bug. During this project, I learned that writing a report which clearly documents my project is much more difficult than I ever imagined. I think I have overcome some of my weaknesses in report writing and I learned how to structure and reference a good quality report.

6.3 What I Achieved

6.3.1 Building Cross Platform Mobile Application

The main achievement of the project was the production of a high-quality cross platform mobile application. Being able to build and install a native standard application on both an Android device and an IOS device ticked a big box on my objectives. Flutter has not yet developed a mature online community to search for answers to problems. This is something which is available for the technologies which have been around longer. This posed a challenge for me but I overcame it by working closely with the official Flutter documentation.

I built this app from first principles, meaning it was started from the basic 'hello world' starter project. Previous assignment projects had the benefit of being started from a version of a working app built during the labs. This would then be extended and adapted to produce the final assignment project. Starting the project with essentially a blank canvas felt slightly chaotic, until the project structure became clearer. There was a challenge early in the project to balance getting a feature built in the sprint versus producing well-structured maintainable code. I should have spent more time on the project architecture at the start before delving too deeply into the coding.

6.3.2 Making NMS Data Available to My Application

Another of the objectives of the project was to use the existing data archives from the National Monument Service in my App. I was initially not clear about how to use this data in the best way. I wanted the 'live' ringfort data to be as consistent and accurate as possible, and the existing NMS data extract did not provide this consistency.

The approach I took was to show the NMS data on a map as a tool to find Ringforts around the country. These could then be converted to 'live' data when the user selected one and performed the updates and saved. Because of the volume of NMS ringfort data, approx. 24,000 records, I had to limit the number of ringforts which appear on the NMS map screen to 500 at any one time. This was to protect the cost structure of the project which relies on 50,000 reads from Firebase per day in the free price tier. See Appendix G for the Firebase pricing structure.

Future development could look at different options around how to retrieve the full collection of NMS ringforts in a cost-efficient way. One suggested approach is to use Firestore data bundles. The data bundles are a *serialized group of documents, retrieved using a specific query*. (Firebase, 2021). These data bundles can then be stored on a Content Distribution Network (CDN) which can improve the speed and cost of how data is retrieved with the help of caching, (Cloudflare, 2022).

6.4 Problems I encountered

6.4.1 Proving the Application on an IOS Device

During development I used an Android emulator to test the new features being added, but because a major project objective was to create a cross platform mobile app, I really needed to test the app on an IOS device.

The only way this was possible was to build and install the app using XCode which is only available on MAC. However, I was using a Windows laptop! After exploring all options, I took the plunge and bought a 2015 iMac and installed XCode. In my sprint planning I had the task estimated at 4 hours to build and install the app for IOS, and 20+ hours later I had managed to get it built and installed to my iPhone.

Several issues were encountered, including initially not being part of the Apple developer program. This meant the release version did install but was only valid for 7 days before needing to be re-validated with a new build and install. This was resolved by discovering that WIT had access to this program for students. Another issue seemed to be caused by my apple watch being synced with my iPhone which seems to be an issue which caused development updates for the watch to try to install endlessly. In the end, I needed to make sure my apple watch was turned off when building and installing the app on IOS.

6.4.2 State Management in the Application

An area I struggled with initially, was implementing successful state management within the app. I settled on using a Flutter package called *Provider* to assist with this. It allowed me to define multiple provider classes in my top-level main class. This allowed multiple screens to have access to the state data and the methods on that data.

For example, this allowed me to implement an app wide **setFilteredSites()** method which updates a Ringfort state variable List called **filteredSites** when called. This method takes a few parameters including: a search string; a favourites Boolean and a local Boolean. This method could be called from the search bar with the search string and it would set the filteredSites to those which matched the search string. It could be triggered from the favourites chip, or triggered from the 'local' chip and set the filteredSites variable to either a List of the users favourite or a List of local Ringforts or both. This method and variable could be used in both the List Ringforts screen and the Map overview screen to give consistent results across the app.

In hindsight I should have spent more time researching state management solutions. There are other options available including one called *Riverpod* which is from the creator of the provider package which has some advantages.

6.5 Possible Future Development

The main constraint for most projects is time, and this project was no different. If there was more time available I would have considered the following:

- An API backend should be considered again if time was not a constraint. There is a benefit to having the data gathered available to others via API calls.
- I would explore allowing multiple images to be stored for each Ringfort and allow for the deletion of images.
- I would investigate better offline functionality for the app, allowing Ringforts to be added and cached locally until internet coverage was available again.
- I think there is an opportunity to adapt this basic functionality of this app for subjects other than Ringforts and re-brand the app accordingly.

7 Project Conclusion

I found this project both challenging and at the same time very rewarding. I took a risk at the start of the project when I picked Flutter SDK and Dart as my front end technology choice having had no previous experience with either. But this choice took me on a learning journey which though daunting at first, gave me a lot of confidence in my ability to learn new skills. I believe that Flutter turned out to be the correct choice which enabled me to deliver a high quality and very consistent cross platform UI experience within the App.

I believe the project overall can be considered a success in that it delivered on all the objectives which were set out at the start in section 1.3. The project Agile/Prototyping methodologies chosen played a big part in making the project successful. This approach meant that the knowledge expert could influence the design of the app from the very start when the screens were prototyped with hand drawings through to the App demonstrations at the end of each sprint. These were valuable opportunities to get feedback which was taken forward in subsequent sprints.

This methodology also has re-planning built in, with the iterative approach that delivers small pieces of working functionality. It meant that problems were found and addressed much earlier, and if changes were needed, they could easily be re-planned in the next sprints.

The local historian played a huge part in the success of the project. I had initially only expected him to give me some time at the start to discuss my ideas. But the fact that he was available for feedback after each sprint meant that development always stayed focussed on the important user requirements rather than getting side-tracked with functionality which was not core to the application.

Bibliography

Blake, S. (2021). *Your Guide To Agile Software Development Life Cycles*. [online] easyagile.com. Available at: https://www.easyagile.com/blog/agile-software-development-life-cycle/.

Clark, J. (2021). *Top 10 Advantages of Firebase*. [online] Back4App Blog. Available at: https://blog.back4app.com/advantages-of-firebase/.

Cloudflare (2022). What is a CDN? | How do CDNs work? | Cloudflare UK. *Cloudflare*. [online] Available at: https://www.cloudflare.com/en-gb/learning/cdn/what-is-a-cdn/.

Department of Housing Local Government and Heritage (2019). *Archaeological Survey of Ireland | National Monuments Service*. [online] Archaeology.ie. Available at: https://www.archaeology.ie/archaeological-survey-ireland [Accessed 4 Dec. 2019].

Dziuba, A. (2021). *Choosing a Map API for Your Next App: Mapbox, Google Maps, or OpenStreetMap?* [online] Relevant Software. Available at: https://relevant.software/blog/choosing-a-map-amapbox-google-maps-openstreetmap/ [Accessed 7 Feb. 2022].

Firebase (2019). *Firebase Pricing | Firebase*. [online] Firebase. Available at: https://firebase.google.com/pricing.

Firebase (2021). *Load Data Faster and Lower Your Costs with Firestore Data Bundles!* [online] The Firebase Blog. Available at: https://firebase.blog/posts/2021/04/firestore-supports-data-bundles [Accessed 1 Apr. 2022].

FireFoo (2022). *Powerful GUI for Firestore - Firefoo*. [online] firefoo.app. Available at: https://firefoo.app/ [Accessed 9 Apr. 2022].

Flutter (2021). *Flutter architectural overview*. [online] docs.flutter.dev. Available at: https://docs.flutter.dev/resources/architectural-overview [Accessed 7 Feb. 2022].

Google (2020). *Firestore Service Level Agreement (SLA)*. [online] Google Cloud. Available at: https://cloud.google.com/firestore/sla [Accessed 29 Mar. 2022].

Hendicott, J. (2017). *The Story Behind Ireland's Fairy Forts and Where to See Them*. [online] Culture Trip. Available at: https://theculturetrip.com/europe/ireland/articles/the-story-behind-irelands-fairy-forts-and-where-to-see-them/.

Kamani, S. (2019). *Building a REST API server in Node.js with Express and MongoDB*. [online] codetree.dev. Available at: https://codetree.dev/node-rest-api-tutorial/.

kella (2021). *Firebase V/S REST APIs — A Developer's Guide*. [online] Medium. Available at: https://medium.com/@sandeepkella23/firebase-v-s-rest-apis-a-developers-guide-54e9b17a4d98 [Accessed 7 Feb. 2022].

Manning, C. (2004). *Irish Field Monuments*. [online] GOVERNMENT PUBLICATIONS OFFICE. Available at: https://www.archaeology.ie/sites/default/files/media/publications/irish-field-monuments.pdf [Accessed 14 Mar. 2022].

Mapbox (2021). *Mobile applications | Help*. [online] Mapbox. Available at: https://docs.mapbox.com/help/getting-started/mobile-apps/ [Accessed 25 Jan. 2022].

Moqod (2021). *Downsides of Firebase: limitations to be aware of*. [online] Moqod — Software company. Available at: https://medium.com/moqod-software-company/downsides-of-firebase-limitations-to-be-aware-of-886ade5ae5a2 [Accessed 7 Feb. 2022].

National Monument Services (2022). *National Monuments Service - Archaeological Survey of Ireland - data.gov.ie*. [online] data.gov.ie. Available at: https://data.gov.ie/dataset/national-monuments-service-archaeological-survey-of-ireland [Accessed 29 Mar. 2022].

Stempniak, A. (2021). *React Native vs. Flutter: A Comparison of Pros and Cons*. [online] www.stxnext.com. Available at: https://www.stxnext.com/blog/react-native-

vs-flutter-comparison/?utm_term=&utm_campaign=&utm_source=adwords&utm_medium=ppc&hsa_acc=4807594886&hsa_cam=15491073565&hsa_grp=&hsa_ad=&hsa_src=x&hsa_tgt=&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gclid=Cj0KCQiAu [Accessed 25 Jan. 2022].

Strapi (2021). *Comparing Mobile Development Platforms in 2021*. [online] Strapi. Available at: https://medium.com/strapi/comparing-mobile-development-platforms-in-2021-a3eb16c4fe1b [Accessed 22 Mar. 2022].

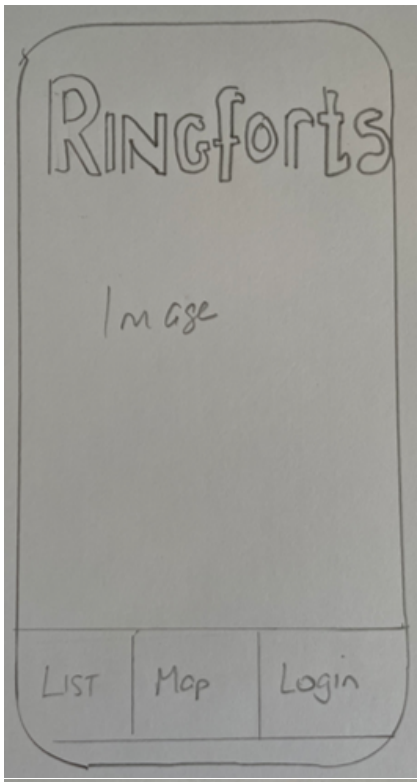
Suranga, S. (2021). *You Don't Have to Compare Flutter and React Native Anymore*. [online] Medium. Available at: https://betterprogramming.pub/you-dont-have-to-compare-flutter-and-react-native-anymore-15ddc4c1342a#:~:text=Flutter%20apps%20come%20with%20different [Accessed 7 Feb. 2022].

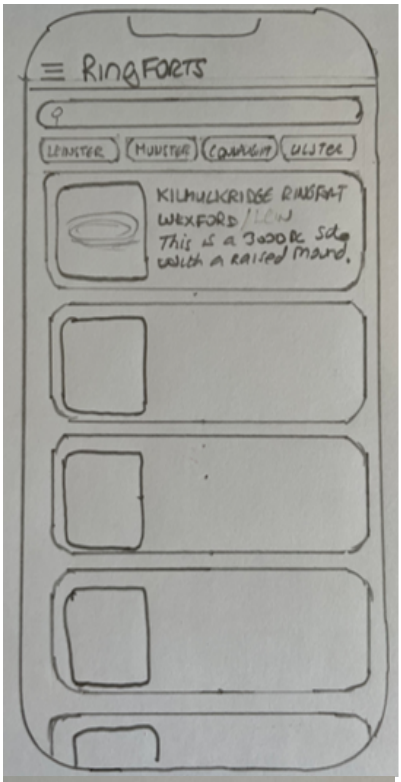
Vailshery, L.S. (2022). *Cross-platform mobile frameworks used by global developers 2020*. [online] Statista. Available at: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/ [Accessed 7 Feb. 2022].

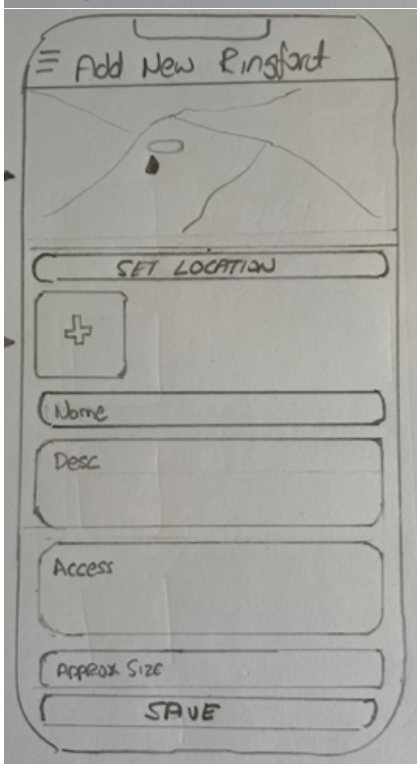
Wikipedia (2021). *Ringfort*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Ringfort#:~:text=Ringforts%2C%20ring%20forts%20or%20ring [Accessed 3 Feb. 2022].

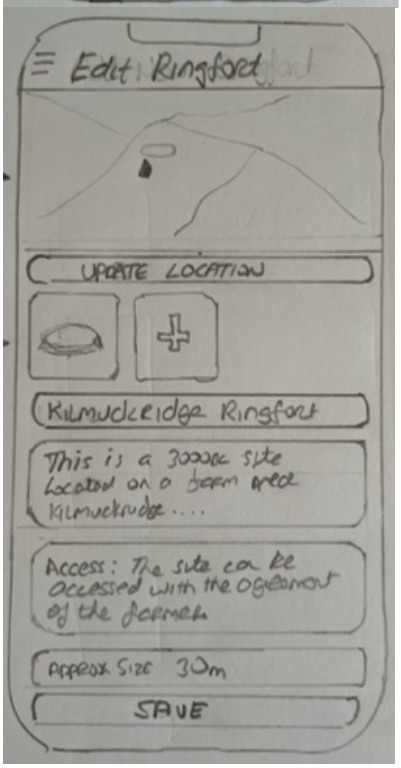
Appendices

Appendix A – Hand Drawn Screens

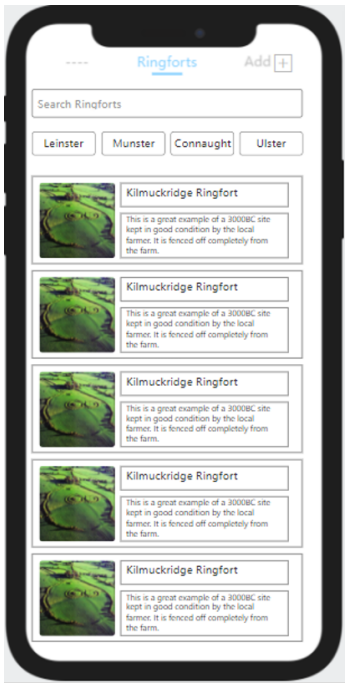


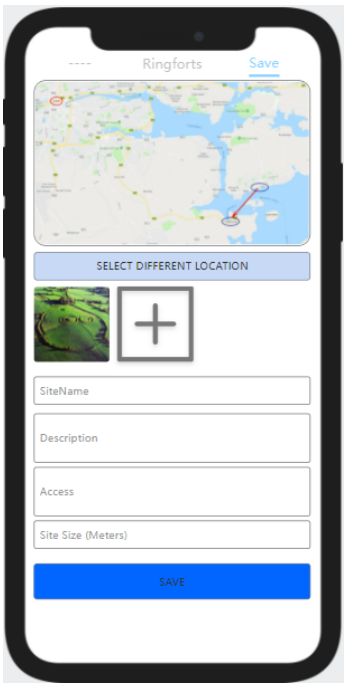


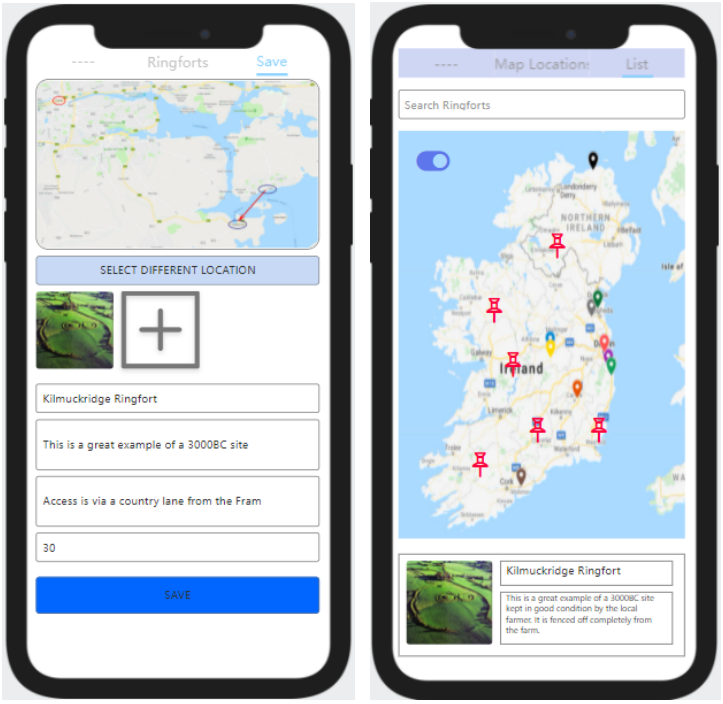




Appendix B – Wireframe Design using Online Tool






Ringforts
Save
SELECT DIFFERENT LOCATION
Kilmuckridge Ringfort
This is a great example of a 3000BC site
Access is via a country lane from the Fram
30
SAVE
Map Location
List
Search Ringforts
Ireland
Kilmuckridge Ringfort
This is a great example of a 3000BC site kept in good condition by the local farmer. It is fenced off completely from the farm.

Appendix C – User Stories

Title	(1) Add a New Ringfort	Title	(2) Add New Ringfort Location
As a	Ordinary User	As a	Ordinary User
I Want	To have the ability to add a new Ringfort to the system in an easy way with basic information.	I Want	To have the ability to easily add the location of a ringfort when adding a new one.
So That	The following important basic information is stored about the Ringfort 1) Ringfort Name 2) Short Description 3) Is the site accessible 4) The approximate size of the site	So That	The exact location of the Ringfort is known with information such as 1) GPS Coordinates 2) Address Details 3) Country 4) Province
Priority	1	Priority	2
Estimate	7 hours	Estimate	5 hours

Title	(3) Add New Ringfort Images	Title	(4) List all Ringforts
As a	Ordinary User	As a	Ordinary User
I Want	To have the ability to easily take a camera image of a ringfort when adding a new one.	I Want	To have the ability to see a list of all Ringforts which have been added to the system.
So That	1) The image is available to be viewed 2) The image is stored locally 3) The location of the image is available and linked to the ringfort	So That	1) A list of all ringforts is available and scrollable 2) Each Ringfort entry displays the Image taken 3) Each Ringfort entry displays the Ringfort Name 4) Each Ringfort entry displays the Ringfort Description
Priority	3	Priority	4
Estimate	4 hours	Estimate	5 hours

Title	(5) Edit Ringfort Details	Title	(6) Delete Ringfort Details
As a	Ordinary User	As a	Ordinary User
I Want	To have the ability to edit an existing Ringfort on the system in an easy way.	I Want	To have the ability to delete an existing Ringfort on the system in an easy way..
So That	The following important basic information is updateable: 1) Ringfort Name 2) Short Description or known information 3) Is the site accessible 4) The approximate size of the site 5) An new image can be added 6) The Ringfort location can be updated	So That	1) When the delete is complete the data is completely removed from storage. 2) I will be given the chance to confirm that the delete should happen before it is executed
Priority	5	Priority	6
Estimate	5 hours	Estimate	3 hours

Title	(7) Make Data Available to All Users	Title	(8) Allow User to Signup
As a	Ordinary User	As a	Ordinary User (New)
I Want	To have the ability to see Ringforts added by other users and that they can see those add by me.	I Want	To be able to sign up to the App using an email and password.
So That	1) When a new Ringfort is added other users can immediately see it in their list when they refresh 2) When a Ringfort is deleted it is removed from other users when they refresh 3) When a Ringfort gets updates the changes are available to other users when they refresh **Note:** Data will be stored centrally on Firebase Firestone	So That	1) When I click Sign-up, I am presented with a screen to allow be to enter an email and password (and confirm password) 2) If the email exist already then I will be notified and allowed to select another email 3) If successful then I will be authenticated to the App. **Note:** Authentication will be managed using Firebase Auth services.
Priority	7	Priority	8
Estimate	8 hours	Estimate	7 hours

Title	(9) Allow User to Login	Title	(10) Allow User to Logout
As a	Ordinary User (Existing)	As a	Ordinary User (Existing)
I Want	To be able to login to the App using an email and password which I Signed-up with previously.	I Want	To be able to logout of the App if I am currently logged in.

So That	1) When I click Login, I will be presented with a screen which will allow me to enter an email and password, 2) If the email and password combination do not match an existing user, then I will be given a message to say that login credentials are not valid 3) If successful then I will be authenticated to the App. **Note:** Authentication will be managed using Firebase Auth services.	So That	1) When I click Logout, I will be logged out of the app and taken to the main screen with options to List, Map and Login. **Note:** Authentication will be managed using Firebase Auth services.
Priority	9	Priority	10
Estimate	4 hours	Estimate	2 hours

Title	(11) Limit Access until User Logged-in
As a	Ordinary User (Existing)
I Want	To disable certain functionality until a user has been authenticated.
So That	The following functionality will not be allowed for **non**-logged-in users, and they will be taken to the login screen 1) Deleting a Ringfort 2) Updating a Ringfort 3) Creating a Ringfort 4) View Ringfort Details **Note:** Authentication will be managed using Firebase Auth services.
Priority	11
Estimate	5 hours

Title	(12) Display image on New Screen
As a	Ordinary User (Existing)
I Want	To be able to display a saved image on a new screen and map zoomable
So That	1) When editing a Ringfort, allow the user to click an existing saved image which would open on a new screen. 2) Allow the image to be zoomable on the new screen. 3) Allow for an attractive transition from one screen to the other.
Priority	12
Estimate	5 hours

Title	(13) Searching the Ringfort List
As a	Ordinary User (Existing)
I Want	To be able to search the list of Ringforts and filter by Province
So That	1) On the Ringfort List screen there will be a search bar, that when typed into will immediately start filtering Ringforts. The search will be in all these fields - Name and Description - Access - Address and Province 2) There are 4 province buttons showing on the list screen. And pressing one will limit the Ringforts displayed to those located in that province. 3) The province selection can be used in conjunction with the search bar. 4) There is an option to revert to all Ringforts displayed.
Priority	13
Estimate	8 hours

Title	(14) Searching the Ringfort Map
As a	Ordinary User (Existing)
I Want	To be able to search the Map of Ringforts and display only marker for matching Ringforts
So That	1) On the Ringfort Map screen there will be a search bar, that when typed into will immediately start filtering Ringforts. This will limit the markers displayed to those which match the search criteria The search will be in all these fields - Name and Description - Access - Address and Province 2) There is an option to revert to all Ringforts displayed on the Map.
Priority	14
Estimate	5 hours

Title	(15) Add a Ringfort as a Favourite
As a	Ordinary User (Logged-in)
I Want	To be able set a Ringfort as a favourite

Title	(16) Make Photos Available to all
As a	Ordinary User (Logged-in)
I Want	To be able to view images taken by other users

So That	1) On the Ringfort List screen there is an option to set one as a favourite 2) On the update Ringfort screen there is an option to set it as favourite 3) There is an option to filter ringforts by users' favourites 4) There is an option to revert to all Ringforts displayed.	So That	1) Make the images taken by a user available to other users. i.e., on the Edit screen and on the list screen NOTE: Image storage will be handled using Firebase Storage
Priority	15	Priority	16
Estimate	7 hours	Estimate	5 hours

Title	**(17) Set up Approval System**	**Title**	**(18) Load and Display NMS Data**
As a	Ordinary User (Logged-in) (Admin-user)	As a	Ordinary User (Logged-in) (Admin-user)
I Want	To make any Add/Update/Deletes subject to approval by an Admin User	I Want	To make the NMS data available to the app and display on a new map. With the ability to click this data to update and add to live database.
So That	1) When an Ordinary user adds a new Ringfort it is sent to an Admin user for approval before it will appear to other users. 2) When an Ordinary user deletes an existing Ringfort it is sent to an Admin user for approval before it will disappear for other users. 3) When an Ordinary user updates an existing Ringfort it is sent to an Admin user for approval before it will appear updated to other users. 4) An Admin user will see an new screen of changes awaiting approval which they can approve or reject	So That	1) Download National Monument Service data from their website and load to a new Firebase collection. 2) When a user chooses NMS Uploaded Data from the Nav Drawer they would be sent to a map screen of all the NMS sites 3) When a user clicks on one of these Ringfort sites they would be brought to an update screen where additional info and images can be added. 4) When the data is saved, it is added to the live collection of Ringforts and removed from the NMS collection on Firebase.

Appendix D – Original Sprint Plan Progress as at 07.02.2022

This is the initial 3 sprint plan with progress showing as at the 07.02.2022. This was completed successfully on the 13.02.2022

Sprint No:	1	2	3
Start Date	04.01.2022	17.01.2022	31.01.2022
End Date	16.01.2022	30.01.2022	13.02.2022
Tasks	1) Compare and Select Front End Technologies (4 hours) 2) Study selected Front End Technology to become competent (25 hours) TOTAL: 29 hrs	1) Contact User (Local Historian) and discuss requirements (1 hour) 2) Mock up sample screens by manually drawing and confirm with user (4 hours) 3) Study selected Front End Technology to become competent (12 hours) 4) Start project report layout and structure (3 hours) 5) Document design of Data model (3 hours) TOTAL: 23	1) Write user Stories for highest priority tasks (3 hour) 2) Turn screen drawings into Wireframes (4 hours) 3) Write project report to interim report level (10 hours) 4) User Story 1 Dev (7 hours) (Add a New Ringfort) 5) User Story 2 Dev (5 hours) (Add New Ringfort Location) 6) User Story 3 Dev (4 hours) (Add New Ringfort Images) TOTAL: 33
Status	Complete	Complete	In progress

Sprint No:	4	5	6
Start Date	14.02.2022	28.02.2022	14.03.2022
End Date	27.02.2022	13.03.2022	27.03.2022
Tasks	1) Continue Project Report (5 hour) 2) User Story 4 Dev (5 hours) (List all Ringforts) 3) Wireframe Login/Sign-up Screens (2 hours) 4) User Story 5 Dev (5 hours) (Edit Ringfort Details) 5) User Story 6 Dev (4 hours) (Delete Ringfort Details) 6) User Story 7 Dev (8 hours) (Make Data Available to All Users) TOTAL: 29	1) Continue Project Report (5 hour) 2) User Story 8 Dev (7 hours) (Allow User to Signup) 3) User Story 9 Dev (4 hours) (Allow User to Login) 4) User Story 10 Dev (2 hours) (Allow User to Logout) 5) User Story 11 Dev (5 hours) (Limit Access until User Logged-in) 6) User Story 12 Dev (5 hours) (Display saved Image on new screen) TOTAL: 28	1) Get Handbook Entries ready (3 hours) 2) Complete self-reflection in report (6 hours) 3) User Story 16 Dev (5 hours) (Make Photos Available to all) 4) User Story 13 Dev (8 hours) (Searching the Ringfort List) 5) User Story 14 Dev (5 hours) (Searching the Ringfort Map) 6) User Story 15 Dev (7 hours) (Setting a Ringfort as a favourite) TOTAL: 34
Status	Awaiting	Awaiting	Awaiting

Sprint No:	7	8	9
Start Date	28.03.2022	04.04.2022	11.04.2022
End Date	03.04.2022	10.04.2022	17.04.2022
Tasks	NOTE: 1 Week Sprint 1) Complete Project Report 2) Fix any code bugs and see if any small additions possible. 3) Complete README	NOTE: 1 Week Sprint 1) Prepare Demo Video	NOTE: 1 Week Sprint 1) Prepare Presentation
Status	Awaiting	Awaiting	Awaiting

Appendix E – Updated Sprint Plan Progress as at 01.03.2022

It became apparent after the 1st three sprints that progress was being made quicker than originally planned, largely due to more time being spent than original envisaged. Based on this I decided to do some sprint re-planning to get a more accurate picture. These show the replanned sprint details and the progress as at the 01.03.2022.

Sprint No:	1	2	3
Start Date	04.01.2022	17.01.2022	31.01.2022
End Date	16.01.2022	30.01.2022	13.02.2022
Tasks	1) Compare and Select Front End Technologies (4 hours) 2) Study selected Front End Technology to become competent (25 hours) TOTAL: 29 hrs	1) Contact User (Local Historian) and discuss requirements (1 hour) 2) Mock up sample screens by manually drawing and confirm with user (4 hours) 3) Study selected Front End Technology to become competent (12 hours) 4) Start project report layout and structure (3 hours) 5) Document design of Data model (3 hours) TOTAL: 23	1) Write user Stories for highest priority tasks (3 hour) 2) Turn screen drawings into Wireframes (4 hours) 3) Write project report to interim report level (10 hours) 4) User Story 1 Dev (7 hours) (Add a New Ringfort) 5) User Story 2 Dev (5 hours) (Add New Ringfort Location) 6) User Story 3 Dev (4 hours) (Add New Ringfort Images) 7) User Story 4 Dev (5 hours) (List all Ringforts) TOTAL: 38
Status	Complete	Complete	Complete

The next 3 sprints have been totally re-planned. Based on spending more time per sprint and tasks taking less time than originally planned.

Sprint No:	4	5	6
Start Date	14.02.2022	28.02.2022	14.03.2022
End Date	27.02.2022	13.03.2022	27.03.2022
Tasks	1) Continue Project Report (2 hour) 2) User Story 5 Dev (5 hours) (Edit Ringfort Details) 3) User Story 6 Dev (4 hours) (Delete Ringfort Details) 4) User Story 7 Dev (5 hours) (Make Data Available to All Users) 5) User Story 8 Dev (5 hours) (Allow User to Signup) 6) User Story 9 Dev (4 hours) (Allow User to Login) 7) User Story 10 Dev (2 hours) (Allow User to Logout) 8) User Story 11 Dev (5 hours) (Limit Access until User Logged-in) TOTAL: 32	1) Continue Project Report (5 hour) 2) User Story 16 Dev (5 hours) (Make Photos Available to all) 3) User Story 13 Dev (6 hours) (Searching the Ringfort List) 4) User Story 14 Dev (4 hours) (Searching the Ringfort Map) 5) User Story 15 Dev (7 hours) (Setting Ringfort as User Favourite) 6) User Story 18 Dev (12 hours) (Adding NMS data and displaying on Map and allow to update) 7) Install XCode on apple IMAC and build the project for IOS. (4hrs) TOTAL: 43	1) Get Handbook Entries ready (10 hours) 2) Complete self-reflection in report (6 hours) 3) User Story 12 Dev (8 hours) (Allow images to be displayed on new screen and zoomable) 4) Bug fixes and commenting (4 hours) TOTAL: 28
Status	Complete	In-progress	Awaiting

Appendix F – Updated Sprint Plan Progress as at 14.03.2022

The replan on the 01.03.2022 proved to be accurate. An additional task was added into sprint 5, this was User story 17 – adding approval system for normal users. Here is the status as at the 14.03.2022 which is the start of sprint 6.

Sprint No:	1	2	3
Start Date	04.01.2022	17.01.2022	31.01.2022
End Date	16.01.2022	30.01.2022	13.02.2022
Tasks	1) Compare and Select Front End Technologies (4 hours) 2) Study selected Front End Technology to become competent (20 hours) TOTAL: 24 hrs	1) Contact User (Local Historian) and discuss requirements (1 hour) 2) Mock up sample screens by manually drawing and confirm with user (4 hours) 3) Study selected Front End Technology to become competent (12 hours) 4) Start project report layout and structure (3 hours) 5) Document design of Data model (3 hours) TOTAL: 23	1) Write user Stories for highest priority tasks (3 hour) 2) Turn screen drawings into Wireframes (4 hours) 3) Write project report to interim report level (10 hours) 4) User Story 1 Dev (7 hours) (Add a New Ringfort) 5) User Story 2 Dev (5 hours) (Add New Ringfort Location) 6) User Story 3 Dev (4 hours) (Add New Ringfort Images) 7) User Story 4 Dev (5 hours) (List all Ringforts) TOTAL: 38
Status	Complete	Complete	Complete

Sprint No:	4	5	6
Start Date	14.02.2022	28.02.2022	14.03.2022
End Date	27.02.2022	13.03.2022	27.03.2022
Tasks	1) Continue Project Report (2 hour) 2) User Story 5 Dev (5 hours) (Edit Ringfort Details) 3) User Story 6 Dev (4 hours) (Delete Ringfort Details) 4) User Story 7 Dev (5 hours) (Make Data Available to All Users) 5) User Story 8 Dev (5 hours) (Allow User to Signup) 6) User Story 9 Dev (4 hours) (Allow User to Login) 7) User Story 10 Dev (2 hours) (Allow User to Logout) 8) User Story 11 Dev (5 hours) (Limit Access until User Logged-in) TOTAL: 32	1) Continue Project Report (5 hour) 2) User Story 16 Dev (5 hours) (Make Photos Available to all) 3) User Story 13 Dev (6 hours) (Searching the Ringfort List) 4) User Story 14 Dev (4 hours) (Searching the Ringfort Map) 5) User Story 15 Dev (7 hours) (Setting Ringfort as User Favourite) 6) User Story 18 Dev (12 hours) (Adding NMS data and displaying on Map and allow to update) 7) User Story 17 Dev (10 hours) (Approval System for Normal users) 8) Install XCode on apple IMAC and build the project for IOS. (4hrs) TOTAL:52	1) Get Handbook Entries ready (10 hours) 2) Complete self-reflection in report (6 hours) 3) Complete report (7 hours) 3) User Story 12 Dev (8 hours) (Allow images to be displayed on new screen and zoomable) 4) Bug fixes and commenting (4 hours) TOTAL: 35
Status	Complete	Complete	In-progress

Appendix G – Firebase Pricing

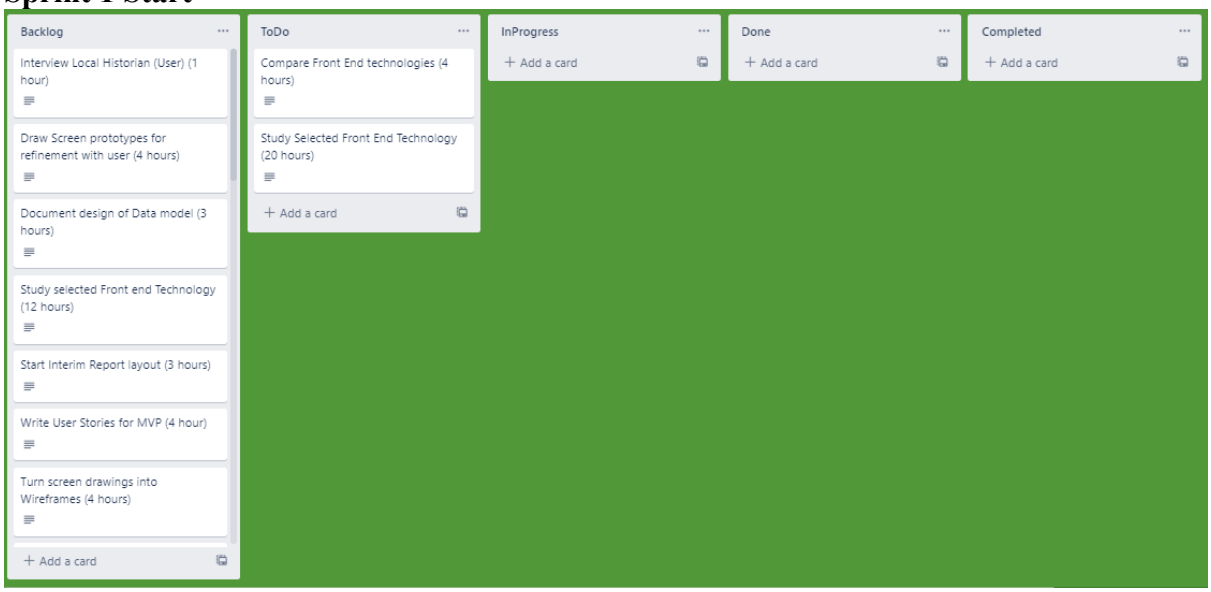
These are the latest Firebase prices from their website.

Products	No-cost Spark Plan	Pay as you go Blaze Plan
Cloud Firestore		
Stored data	1 GiB total	No-cost up to 1 GiB total Then $0.108 per additional GiB
Network egress	10 GiB/month	No-cost up to 10 GiB/month Then Google Cloud pricing
Document writes	20K writes/day	No-cost up to 20K writes/day Then Google Cloud pricing
Document reads	50K reads/day	No-cost up to 50K reads/day Then Google Cloud pricing
Document deletes	20K deletes/day	No-cost up to 20K deletes/day Then Google Cloud pricing
Cloud Storage		
GB stored	5 GB	$0.026/GB
GB downloaded	1 GB/day	$0.12/GB
Upload operations	20K/day	$0.05/10k
Download operations	50K/day	$0.004/10k
Multiple buckets per project	✗	✓

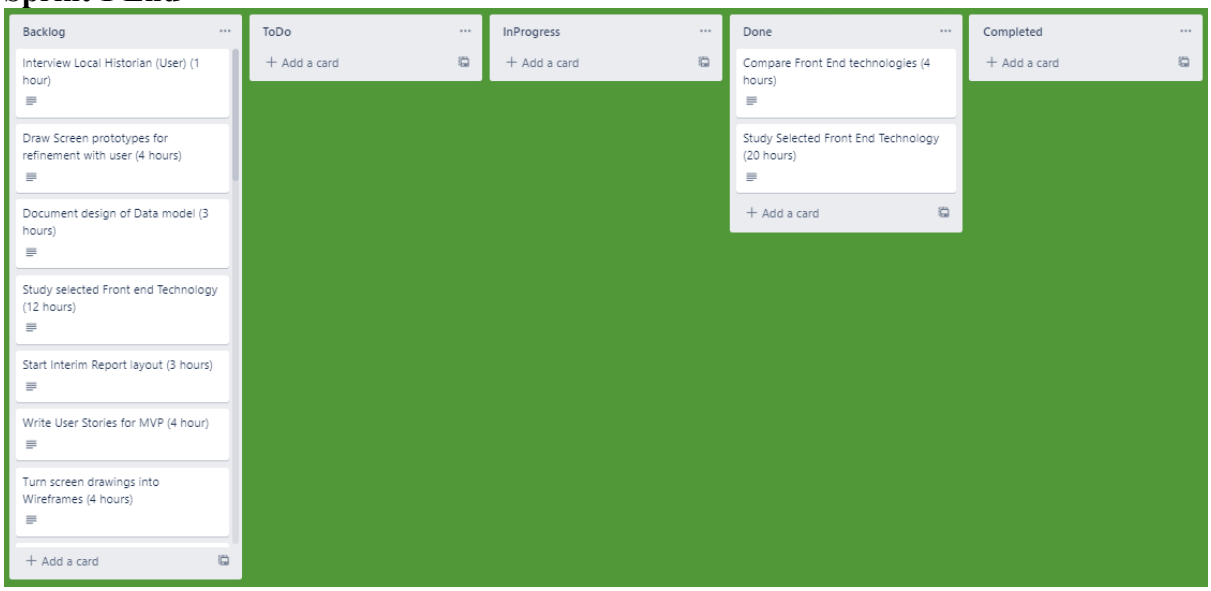
Appendix H – Trello Sprint Tracking

This section shows the sprint progress, with a snapshot of the Trello board before and after each sprint up to and including the sprint 6 the last development sprint.

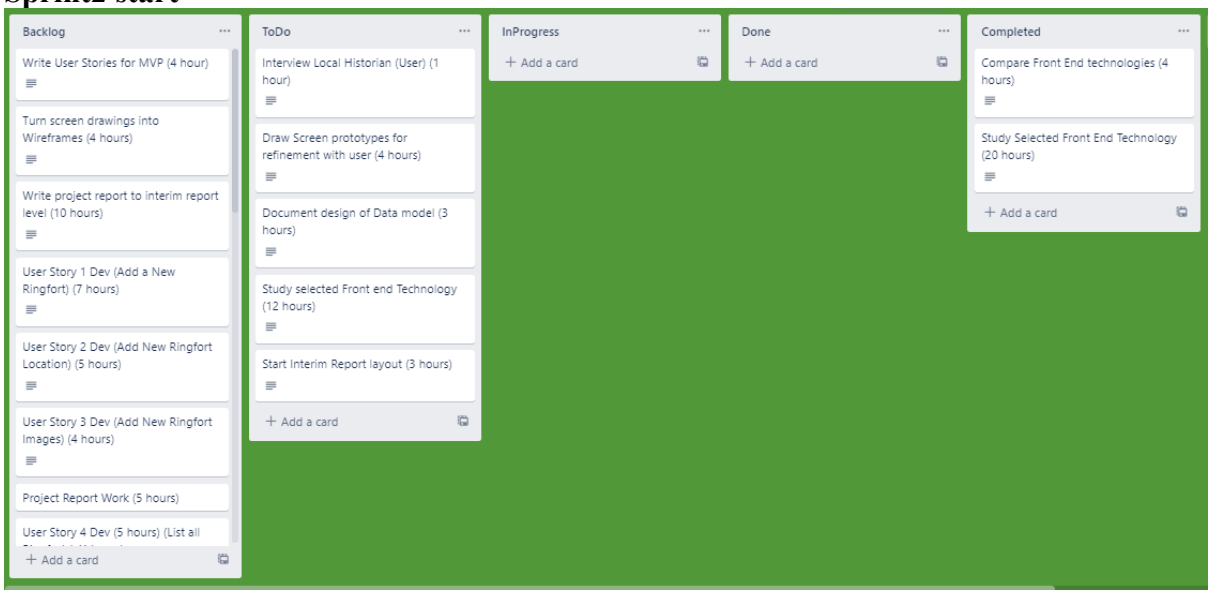
Sprint 1 Start



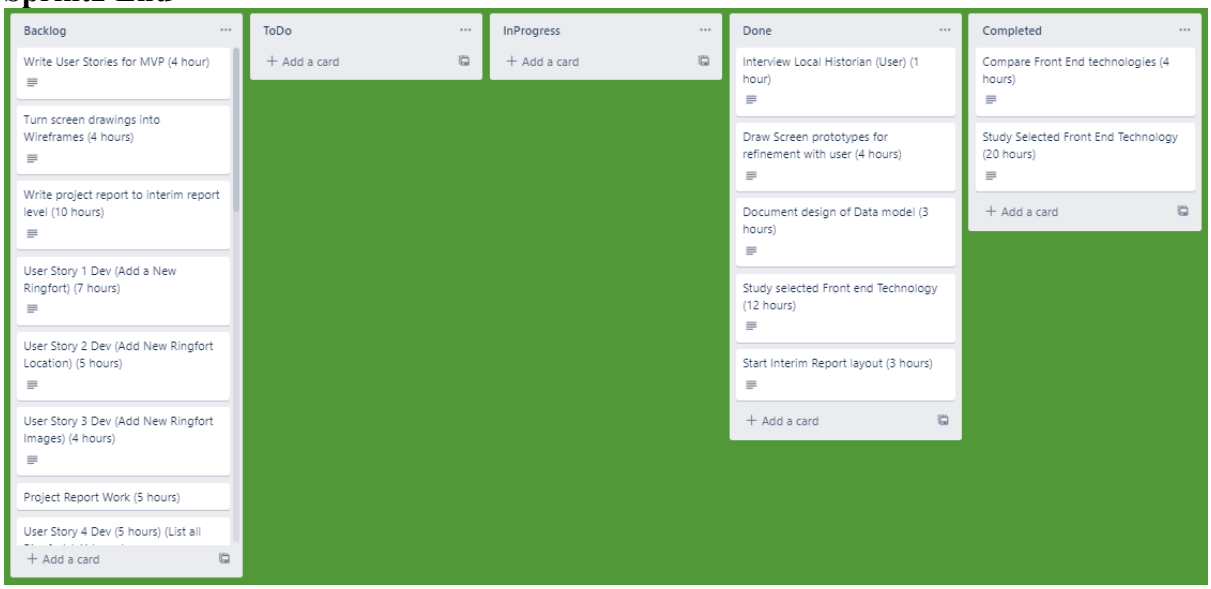
Sprint 1 End



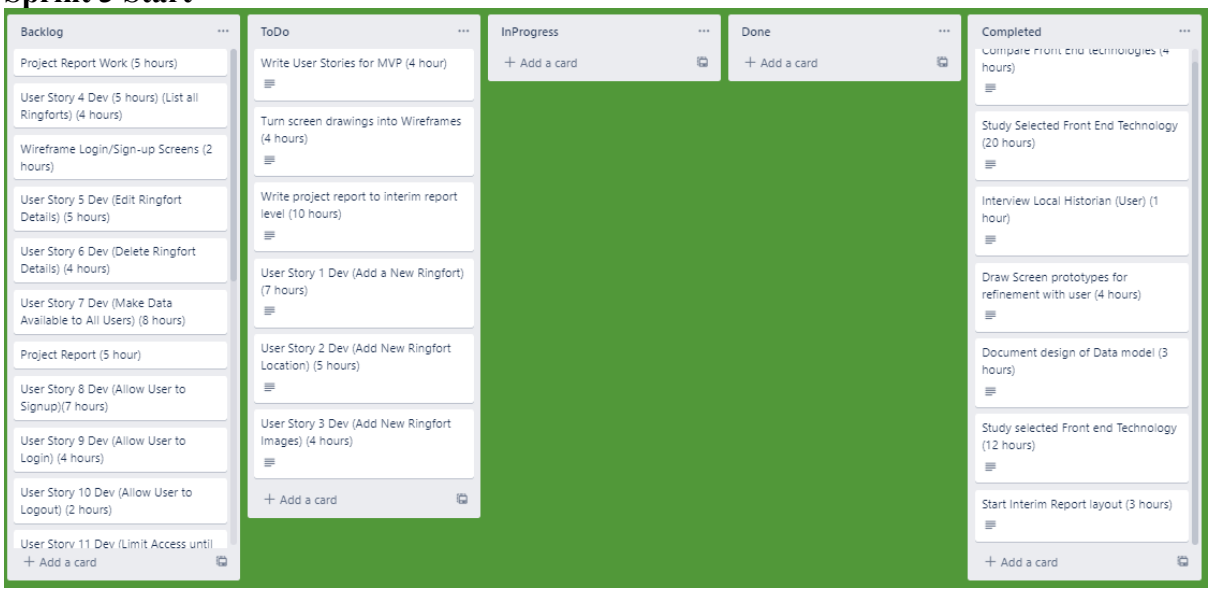
Sprint2 start



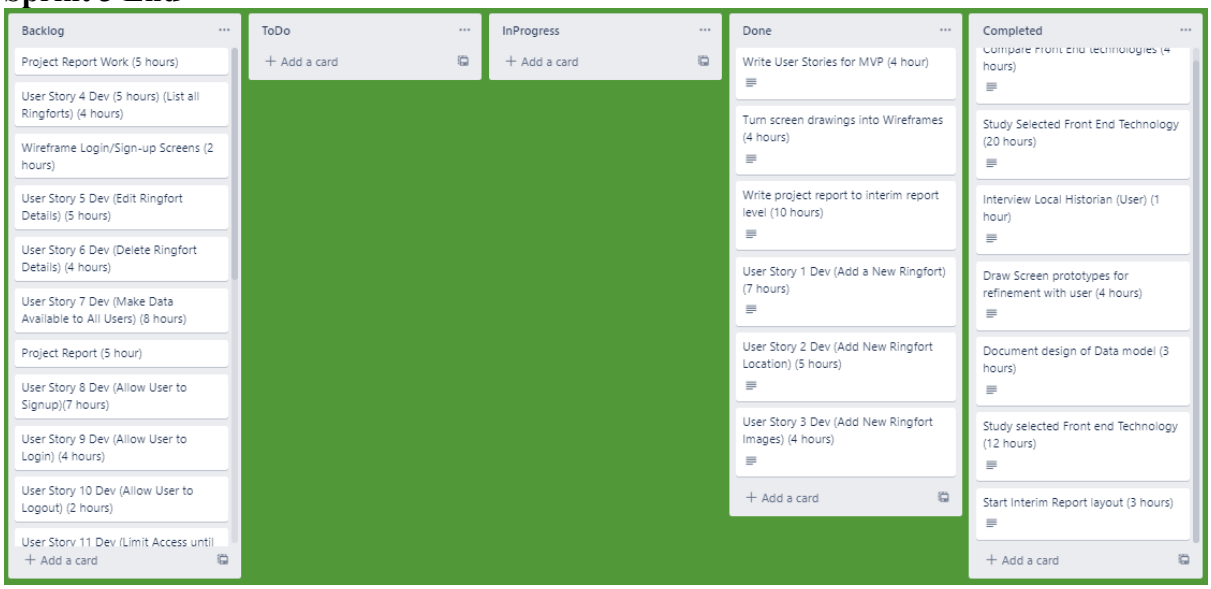
Sprint2 End



Sprint 3 Start



Sprint 3 End



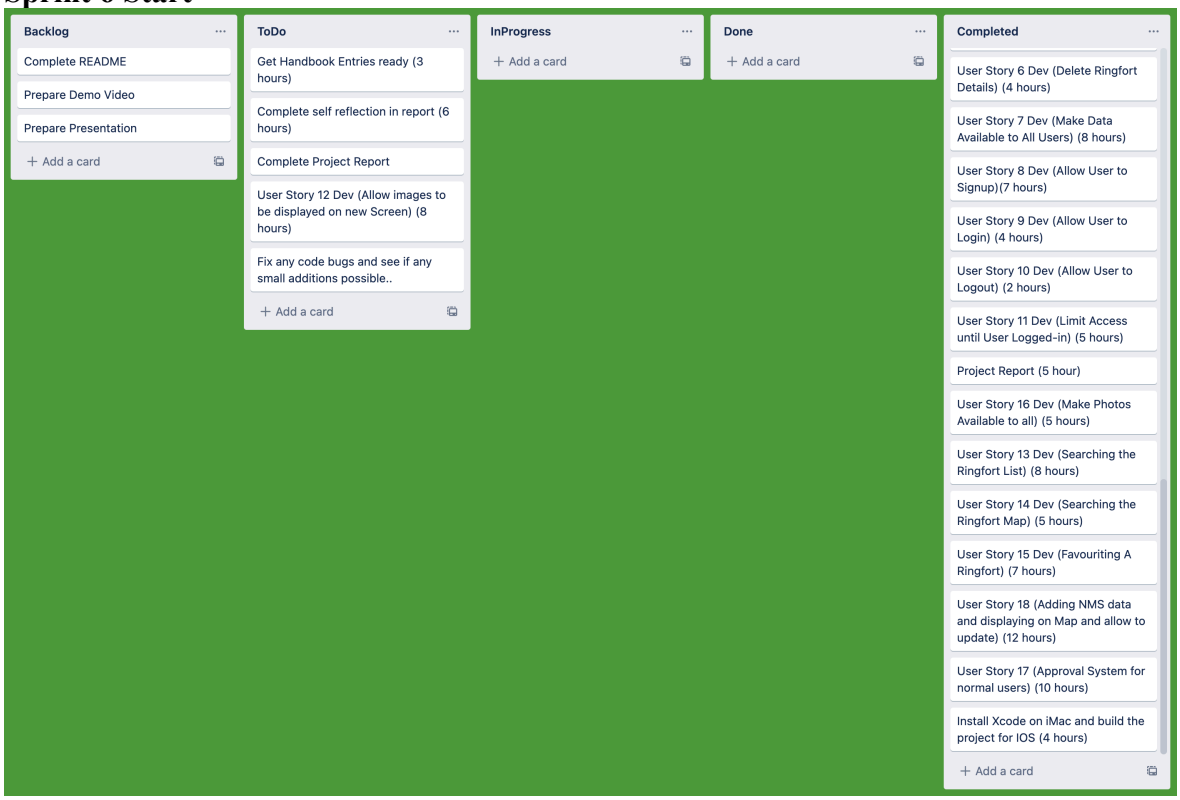
Sprint 4 Start

Sprint 4 End

Sprint 5 Start

Sprint 5 End

Sprint 6 Start



Sprint 6 End