
The use of the mean stack in an ionic application

Jason McTigue

Alan Murphy

B.Sc.(Hons) in Software Development

APRIL 17, 2017

Final Year Project

Advised by: Dr. Matrin Kenirons

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)



Contents

1	Introduction	6
1.1	What it's about	6
1.2	Scope	7
1.2.1	Backend/Database tier	7
1.2.2	Server side/Node	7
1.2.3	Frontend/ Ionic application	7
1.3	Objectives	8
1.4	Documentation	8
2	Methodology	10
2.1	Initial planning and research	10
2.2	Project planning and meetings	11
2.3	Github as a development tool	11
3	Technology Review	13
3.1	MEAN-Stack	13
3.1.1	MongoDB	14
3.1.2	How mongo works in our application	15
3.1.3	Express	16
3.1.4	Node.js	16
3.1.5	AngularJs	16
3.2	RESTful API	17
3.2.1	How RESTful API's work	18
3.3	Ionic 2	18
3.3.1	What Ionic is	18
3.3.2	Typescript	20
3.4	Firebase	20
3.5	Heroku	21
3.6	GitHub	21

4	System Design	23
4.1	Architecture	23
4.2	Front end	24
4.3	Middle Tier	24
4.3.1	Node server	24
4.3.2	Express	25
4.3.3	Heroku and mLabs (Cloud Hosting)	25
4.3.4	AngularFire2	26
4.4	Back end	26
4.5	Ionic Application	27
4.5.1	Login Page	28
4.5.2	Sign Up Page	29
4.5.3	Reset Password Page	30
4.5.4	Map Page	31
4.5.5	My Places	31
4.5.6	Landmarks Page	33
4.5.7	Reviews Page	34
5	System Evaluation	35
5.1	Testing	35
5.2	Limitations	37
6	Conclusion	38
7	Appendices	41

About this project

Abstract This project was developed as our final year project in our fourth year to achieve a level 8 in B.Sc. (Hons.) in Software Development in GMIT. The purpose of this project which we set out for ourselves was to develop a cross-platform mobile application Using Ionic 2 and the MEAN stack of technologies (MongoDB, Express.js, Angular.js and Node.js). The application is a review app which allows users to see their current location on a map and save that location on their device. Then the users can go to the review page to leave reviews about a given location and see what others are saying about that location or other locations. The app also points out specific landmarks which the users can go and check out if they are new to the area. The app consists of a 3-tier architecture, a front-end, back-end and middle tier. The front end consists of our ionic application, the middle tier consists of the node server which contains our API and builds our express app. The backend is our MongoDB database for storing reviews. Ionic handles all of the User Interface and allows the app to be cross platform. The node server acts as a bridge between the database and the app, it's hosted online using Heroku so the mobile application can access the API as Mobile devices cannot run the node server we have created. Our database is connected to a mlab deployment which is also connected to our Heroku deployment. This all brings the app together with our frontend connected to our hosted node middle tier so it can access the data from the database by consuming the API. Our API is a RESTful API which sends JSON data back and forth between the database and the server and also back and forth between the Mobile application and the server. This is how our reviews are kept up to date with multiple concurrent users on the app at the same time. Google maps API is then used to display the user's location on the map and for saving landmarks or custom locations to the app.

Authors This project was developed by two students from GMIT Alan Murphy and Jason McTigue in order to achieve a level 8 B.Sc. (Hons.) in Software Development. The front end and the design of the UI were devel-

oped by Alan and the backend and RESTful API was developed by Jason. We then both worked on the Node server to connect all the technologies together to create our application. Both of us are involved in writing this document.

Chapter 1

Introduction

This is our dissertation for our final year project which we are developing to achieve a B.Sc. (Hons.) in Software Development. This project was done in a group of two people (Alan Murphy and Jason McTigue) and together we came up with the idea of creating a location-based Event/Review cross-platform app which would allow users to create events in their location and save them to the app. It also allows users to create reviews about any location or event in their area. We named the app GoLocate.

1.1 What it's about

The premise of our project was to build a hybrid application that used a collection of technologies that worked together, were new and complex. Our project originally consisted of Ionic 2 and Firebase 3 but after much discussion with our supervisor and after some of our own research we realised that the two technologies together were too simple for a level 8 project and took too much control away from our server side and database logic. With this in mind, we went to research more technologies which we would deem more of an appropriate standard for a level 8 project. After all our research we discovered the MEAN stack[1]. The MEAN stack is a collection of technologies that are all Javascript based which is great for combined development between the technologies. The MEAN stack consists of an Angular.js frontend (In our case Angular 2), A MongoDB backend for our database while Express and Node.js are the remaining technologies which control our server side logic. Along with the MEAN stack setup, we also include Ionic 2. Ionic 2 is the framework which we use to handle the frontend of our app. Ionic also allows us to cross develop the app for Android and IOS synchronously.

1.2 Scope

This project was developed by two people. With this in mind, we both agreed that the scope of the project was correct. The project has a three-tier architecture with 3 unique parts all communicating with each other to form our app. The 3 tiers are as follows:

1.2.1 Backend/Database tier

This is our mongo database which contains all our information on reviews that users can create and delete from the app. This database is also connected to an mLabs connection which in turn is then connected to our Heroku deployment which hosts our Node server with our RESTful API. Together these technologies all form the pieces which allow users to add to and delete from review items. In summary, the backend of the app sends information in JSON format to the Node server and anything sent from the frontend to the Node server is also sent back to the database in JSON format.

1.2.2 Server side/Node

This is our middle tier which connects our Ionic app to our Mongo database. This tier consists of our Node server which contains our RESTful API and tells our app to connect to our database containing our reviews. The REST API allows the app to get JSON data from the server and output it in our UI in the app. This tier is also hosted on Heroku. This allows users on their phone to use this app as Node servers can not be run by a mobile device. Instead, the Ionic app connects to the hosted Node server on our Heroku deployment.

1.2.3 Frontend/ Ionic application

The front-end of our application is the cross-platform app developed using Ionic 2. Ionic 2 uses a combination of HTML, SCSS and Typescript to design the UI of our app. The Ionic app gets data from the database by connecting to our Node server which contains our REST API. It also allows users to add data to our server using a UI to input information which is pushed in JSON format to our server and into our database.

1.3 Objectives

Our primary object in developing this application was to learn a set of new skills using the MEAN stack to build a cross-platform mobile application. To go about this we set these milestones from the beginning:

- To create our app using Ionic 2 and the MEAN stack by learning the new framework and getting to grips with Typescript. Learning how to create our MongoDB database and connect it to m Labs and then connecting our Ionic app to our database using node. By the end of this project, we wanted to be very familiar with these technologies and felt we've learnt a lot about how they work.
- We wanted users to be able to access the app securely with authentication and be able to login with a username and password or with their social media profiles.
- We wanted to add a map feature where users can see their current location, save their location and name it and see important landmarks in their area.
- We wanted our users to be able to post reviews or just general comments about the events around them and save them to the app and also be able to delete them again.
- We wanted Users to be able to see immediately any newly added reviews to the app by other users. If users refresh the Review page in the app and someone has added a new review since the last refresh, the new review should appear on their device.

1.4 Documentation

Up until now, this document has given a brief description of what this project is, how it was developed, where the idea came from, our goals for the project and the scope of the project. This project will be discussed in even more detail under the following headings in this document.

Methodology - This section contains how we went about our project in terms of both development and research and will also give our opinion on how we felt everything was approached at a proper level for a level 8 project. Our project planning will be discussed in terms of which project methodology was used, how often we had meetings and what the meetings consisted of,

how we divided out the workload between the group and the feedback we got from our supervisor

Technology Review - This part of the documentation documents the outcome of our research in the technologies we were using and how we used them to achieve our objectives described in the introduction. Each technology we used for the app will be described in detail in both what they are and exactly how we used them.

System Design - This section will describe the architecture of the project. It will describe how each technology works together to form this application. It will have diagrams to help visualise the architecture of the project and will contain screenshots of the application.

System Evaluation - This is where the project will be evaluated by comparing the final outcome of the app to our objects initially set out. Difficulties and limitations met in the development process of the app will also be discussed in this section.

Conclusion - This will be a summary of our entire finished project. It will discuss the aforementioned goals of the project and how well we managed to achieve them throughout the development process. It will discuss what we learned and our experiences in developing this application. It will mention what we done right during development and what we would go back and change if we were given the chance. Overall it will nicely wrap up this documentation and add some final piece of insight on the overall project experience.

Chapter 2

Methodology

2.1 Initial planning and research

When we set out to create this application we first ensured that we did a lot of research about the technologies we would be using. We has a basic knowledge of Ionic from previous projects that we worked on and we both felt it was a good decision to jump from Ionic 1 to Ionic 2 for this application. Ionic 2 was a completely new framework to Ionic 1 and we had to take some time to research the new features and technologies like Typescript. After completing research of the framework we felt Ionic 2 was the right choice and was at the right level of complexity to be worthy of a level 8 project.

With our frontend technology chosen we then continued research for our backend. We knew our backend would have to have a database for our reviews and events and we knew in the future this database would have to be hosted online. This resulted in us discovering Firebase. Firebase would have allowed us to store information in an online database, Authenticate users with their firebase authentication service and host our app online for use on a web browser.

In the end, our original approach planning proved to be ineffective and not well thought out. We were advised by our supervisor that we should be building our application with a 3 tier architecture. This set us back on the development of the project as we were already well into the development stage of the application with Firebase. We had to set out and quickly research some new technologies and see how we could work them into our app idea. Our methods of project planning are discussed below.

2.2 Project planning and meetings

All planning of this project were done in weekly meetings. In our first meeting for the project, we sat down and discussed what we wanted the app to be and how we were going to go about developing it. When we came to a conclusion we then set out some objectives to be met by the next meeting and began researching the technologies.

After each group meeting about the development of the application, we then had another weekly meeting with our supervisor Dr. Martin Kenirons to whom we reported our research and findings and discussed how to go about development in a structured and organised fashion. Martin was very helpful when it came to project planning and he advised us greatly on how to approach development as a group. He also helped us realise that our original project idea might not have been to the standards of a level 8 assignment and he advised us to research the MEAN stack which is the group of technologies we decided to use for development.

Each of these technologies were new to us so heavy research had to be completed in a shorter amount of time in order for us to properly develop this application. With all the technologies researched and our application clear in our head, we went about developing the app. We thought about our app in 2 parts with both the ionic 2 frontend and the MongoDB backend. We decided to split development of these to ends of our project between us with Jason taking up development of the database and Alan taking up development of the frontend UI. We then worked together to get the node server side of things to connect these two parts together to complete our application architecture. Weekly meetings between the group and meetings with Martin continued and each week we would set an objective and try to have it completed by the next meeting. If our objectives were not met we would discuss any problems between us and try to determine if we were going about our goals in the correct way. We also created a messaging group on facebook so if we had any urgent problems to be sorted outside of college we could instantly message the whole group for discussion and advice.

2.3 Github as a development tool

Because this is a group project we decided using Github would be the best way to share or work between us. A GitHub repository was created and all group members and our supervisor were added as a collaborator to the repository. The app was first created in Ionic and pushed up to GitHub for all members of the group to pull down and start development. We decided

the best approach to using Github was to always leave our master branch with a clean working version of the application and any new developments to be made to the app would be done in branches which we made many of throughout the entire development process. This saved us a lot of trouble when it came to any errors or bugs in our code. If the problem proved impossible to fix we could always go back to the master branch pull that back down and try again or try something new. We had all used Github before but we had never utilised branches in such a way as this before. We were pleased with the results of our approach to Github to manage our code and we believe it was the right way about it and saved us a lot of trouble in the later stages of development.

We also used GitHub to discuss any issues we had during the development process. Github offers an issue feature in which the user can state an issue they have met and ask for help from other collaborators of the project. We made sure to use this as often as we needed to quickly resolve them and continue development of the application. Once they were resolved we made sure to close the issue to avoid any confusion.

Chapter 3

Technology Review

In this section, we are going to talk about the different technologies we incorporated into this project. We have used a variety of new technologies which we felt would work well in our application. In this section, we will talk about the technologies we choose and our reasons for choosing these technologies. For the backend development, we used the MEAN stack, for front-end development we chose Ionic Framework for cross platform app development so we could build the app on Android, IOS and Windows Phone. We then used Node.js for the server side aspect of the application, the node js server was then hosted on Heroku. First, we are going to talk about the MEAN stack, then about Ionic and Heroku and then finishing off with GitHub.

3.1 MEAN-Stack

What is the MEAN stack? A straight to the point explanation would be that the MEAN-Stack is a free and open-source JavaScript software stack for building dynamic web sites and web applications. Mean Stack is a combination of four widely popular and highly efficient Javascript libraries, namely MongoDB, Express.js, Angular JS and Node.js. (Shortened down to M for Mongo, E for Express, A for Angular and N for Node).

Mongo DB can be used to store documents in the JSON format, these JSON queries are then handled by Express JS and Node.js on the server side. Angular JS on the frontend is then fed these JSON documents. With the same language on both the client side and the server side. These two technologies work extremely well together and integration between these two environments is seamless and very subtle.[1]

Because all components of MEAN stack support programs written in JavaScript, MEAN applications can be written in one language for both

server-side and client-side execution environments. In order to completely understand the reasons why mean stack is so widely used, we will talk about each of its components in detail below.

3.1.1 MongoDB

The MEAN stack comes with a NoSql database technology called MongoDB. It is important that we understand what a NoSQL database is before going into further detail on MongoDB. NoSQL stands for (Not only Sequential Query Language). It is a database that provides a mechanism for storage and retrieval of data which is modelled by means other than the tabular relations used in relational databases.

NoSQL databases have been proven to be the solution to what is known as Big Data as they follow a schema-less data model. A NoSQL database provides increased scalability and flexibility compared to relational databases. Studies show that in recent years developers and organisation have experienced a sharp rise in the volume of user data and products that have to be stored in databases.

NoSQL databases started gaining popularity in the 2000's when companies began investing and researching more into distributed databases [2]. NoSQL databases are widely used to store and retrieve very large amounts of data using a key-value format. These types of databases have emerged as the best choices that suite modern mobile and web development. So now that its clear what NoSql is we are going to talk about MongoDB in more detail.

MongoDB is a schema-free document database written in C++ and developed in an opensource project by the company 10gen Inc [3]. The name mongo is extracted from the word humongous. According to its developers, the main goal of MongoDB is to close the gap between the fast and highly scalable key-value-stores and feature-rich traditional RDBMSs. It provides high availability , high performance, and automatic scaling and allows data insertion without a predefined schema.

MongoDB is one of several database types to arise in the mid-2000s under the NoSQL banner. Instead of using tables and rows as in relational databases, MongoDB is built on an architecture of collections and documents. Documents comprise sets of key-value pairs and are the basic unit of data in MongoDB. Collections contain sets of documents and function as the equivalent of relational database tables. A record in MongoDB is composed of field and value pairs and are similar to JSON objects. The value of field may consists of arrays, and array of documents or other documents.

MongoDB maintains data consistency in the sense that one write operation to the data in the database allow subsequent read operations. They use a locking mechanism that contributes to increased execution time as the number of update operation increases. [8] [10].

MongoDB supports dynamic schema design, allowing the document a collection to have different fields and structures. The database uses a document storage and data interchange format called BSON, which provides a binary representation of JSON-like documents. MongoDB also uses Automatic sharding which enables data in a collection to be distributed across multiple systems for horizontal scalability as data volumes increase.[4].

3.1.2 How mongo works in our application

In our application, we used mongo to store information from our reviews section. We wanted people to be able to add a review and post so other users could then also see that review. An example of how we set up our endpoints in the node server is shown here:

```
app.post('/api/reviews', function(req, res)
```

Next, we created a Review provider which was a Typescript file that contained three methods `getReviews`, `createReview`, and `deleteReviews`. The `getReview` function sends a get request to our Heroku server that will then return our review data. The `createReview` function accepts a review object as a parameter and then posts that to the same endpoint. While our delete review function will make a request to the API to delete it. A small extract of our `getReviews` function is shown below:

```
getReviews(){  
  
  if (this.data) {  
    return Promise.resolve(this.data);  
  }  
  
  return new Promise(resolve => {  
  
    this.http.get('https://fathomless-chamber-74986.herokuapp.com/api/reviews')  
      .map(res => res.json())  
      .subscribe(data => {  
        this.data = data;  
        resolve(this.data);  
      });  
  });  
}
```

```
});  
});  
}
```

We used mongoose to connect our application to mlabs. Mongoose is a MongoDB ODM for Node. It provides the user with a simple validation and query API to help you interact with the MongoDB database.

```
mongoose.connect('mongodb://<username>:<password>@<mlabconnection>');
```

3.1.3 Express

Express is another component of MEAN Stack. Express is a nodejs asynchronous based web framework. Express.js builds on the underlying capability of Node, by providing a web application server framework. Express.js is a Node.js web application server framework, designed for building single-page, multi page and cross platform hybrid web applications and it gives Node.js a more realistic website structure that is not present when using Node by itself. For our application express will allow us to create routes for the REST API we will be creating.[5]

3.1.4 Node.js

Node.js is a Javascript runtime built on Chrome's V8 JavaScript Engine. Node.js uses an event driven, non-blocking I/O model that makes it lightweight and efficient web server environment, idea for constructing a web-service API's. Node.js package ecosystem, npm, is the largest ecosystem of open source libraries in the world. For our application node will be our server which will sit between the frontend of the application and the MongoDB database.[6]

3.1.5 AngularJs

The final component making up the mean stack is Angularjs. Defined in Angulars official documentation[7] - AngularJS is a structural framework for dynamic web apps. Angular allows the user to use HTML as their front end language and lets you extend HTML's syntax to express an application's component clearly. Angular also has a data binding and dependency injection which eliminates much of the code you currently have to write. This all happens within the browser, making it a perfect partner for any server technology.

AngularJS simplifies application development by presenting a higher level of abstraction to the developer. Like most types of abstraction, it comes at a cost of flexibility. In other words, not every app is a good fit for AngularJS. AngularJS was built with the CRUD application in mind and this was another reason why we thought this technology would be a good fit for our application as we wanted to have a section where we could add and delete data to a database. [8]

There are a number of reasons why angular js is so popular and some of which include how Angularjs structures the source code by following the Model View Controller. The second reason is Angularjs ability to do two-way data binding. It decreases the amount of code that is written to keep the model and view in agreement. Angularjs models are old java object (POJO), therefore it is quite simple to change or append properties without any major complications. Finally, and probably the most important feature that Angular js has to offer is about dependency injection. Dependency injection is a software design pattern that deals with how components get hold of their dependencies. The angular injector subsystem is in the charge of creating components resolving their dependencies and providing them to other components as requested [9].

After speaking in some detail about each component of the mean stack you should have a clearer understanding of this technology and how it works and benefits applications.

3.2 RESTful API

A RESTful API is an application program interface (API) that uses the HTTP requests GET, POST, PUT, and DELETE data. A RESTful API, which is also referred to as a RESTful web service is based on representational state transfer (REST) technology[10].

A RESTful web service is based on representational state transfer (REST) technology. It is an API that communicates with HTTP requests to GET, PUT, POST and DELETE data and then links to the four fundamental database operations - CREATE,READ,UPDATE,DELETE. An API for an application is basically code that allows two software programs to communicate with each another. In our case the api is created in our node server file and this then links to our database where we are able to apply the HTTP requests.

3.2.1 How RESTful API's work

The API basically takes different parts of a transaction to make a number of small modules. These modules then target a specific underlying part of the transaction. As a result, developers are provided with a lot of flexibility.

The RESTful API uses GET for read and idempotent requests to retrieve a resource, POST for write requests which create a resource, PUT to change the state of or update a resource and DELETE to remove it. All calls are presumed to be stateless which means nothing can be retained by the RESTful service between executions. As a result, REST is suited to cloud applications because stateless components can be freely redeployed if something fails. [11]

The reason for this is that requests can be directed to an instance of a component and therefore there is nothing kept that needs to be remembered by the next transaction.

For these reasons REST is largely preferred for web/mobile use. The RESTful model can also be extremely helpful in the cloud as using APIs to bind services is as simple as controlling how the URL is decoded.

3.3 Ionic 2

The end user mobile app has been developed in Ionic 2 Framework. When in the planning stage of this application we wanted to target both Android and IOS platforms. The Ionic framework suited us because it is completely free to use and Ionic offers a library of mobile-optimised HTML, CSS, and JS CSS components, gestures and tools for building high performing applications. Below we are going to talk about ionic and its uses in more detail.

3.3.1 What Ionic is

Ionic is an HTML5 framework targeted at building hybrid mobile applications. First created by Max Lynch, Ben Sperry, and Adam Bradley in 2013, It is a complete open-source SDK for hybrid application development. Hybrid applications are essentially small websites running in a browser shell. This shell then has access to the native platform layer. Hybrid applications have many advantages in comparison to native applications. Such advantages include the speed of development, platform support, and access to 3rd party code. Ionic even provides different services and tools for development of cross platform applications by using technologies like CSS, HTML5, and Sass.[12]

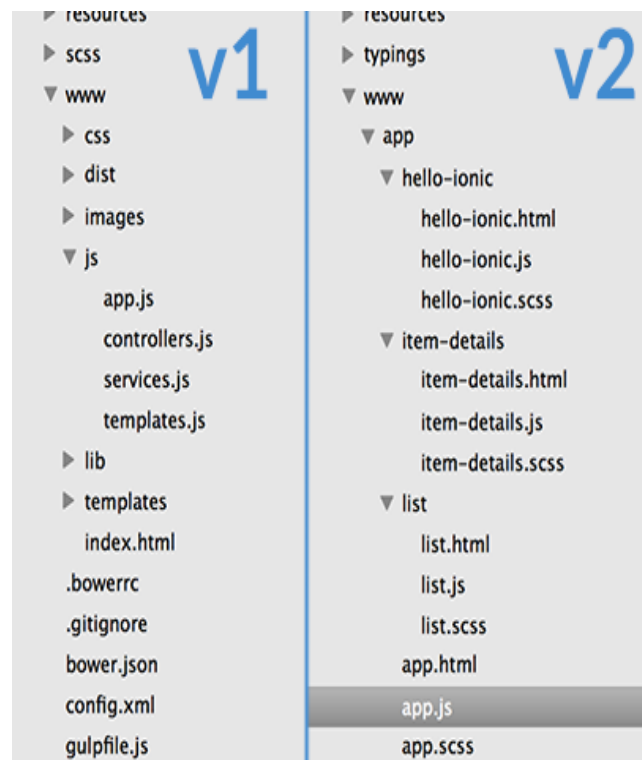
Ionic is described as the front-end framework that controls the design,

feel and UI that every application needs in order to be compelling. Ionic also adds support for a wide range of common native mobile components, animations, and compelling designs.

Ionic has native-styled mobile UI elements and layouts that a user would expect to get with a native SDK on iOS or Android.

As Ionic is an HTML5 framework, it requires a native wrapper like Cordova or PhoneGap in order to run as a native application. Both wrappers contain a wide variety of plugins which can be easily in-cooperated into any application. As mentioned above ionic originally released in 2013 and was quite popular on release. In late 2016 ionic then release version 2 which was a vast improvement on the original. The Ionic 2 structure was much improved in comparison to ionic 1 with each folder being an individual feature and that folder containing all the code necessary for that feature. This makes the code much easier to manage as it is more modular which allows the developer create components that can be dropped into a project at ease, components are a core concept of Ionic 2 and Angular 2[13]. While trying to understand Ionic 2 better we followed along with some tutorials we found on this blog post [14].

An example of the difference between the file structure of an ionic 1 and 2 application is shown below:



3.3.2 Typescript

As previously mentioned one of the main difference between Ionic 1 and 2 is the fact that Ionic 2 uses Typescript instead of javascript. Microsoft created TypeScript with its first public release in October 2012 but Typescript has only become more popular in web development since angular and ionic added it to their 2.0 frameworks. So to get a better grasp of Typescript we are going to talk about it here in more detail. Basically, TypeScript is a superset of JavaScript that compiles into Java, which means it behaves identical to JavaScript but with some extra features added in. So you don't run TS on your web server, ultimately it's all JavaScript. TypeScript also allows developers access to powerful tools for writing modern JavaScript.[15] [16]

3.4 Firebase

For our login page we needed to handle user Authentication. To do this we used Firebase Authentication.

Firebase Authentication provides backend services, SDKs, and ready to use UI libraries to authenticate users of your application. It supports authentication using passwords, popular federated identity providers like Google, Facebook and Twitter.[17]

Firebase Authentication integrates with other Firebase services, and it competes with industry standards like OAuth 2.0 and OpenID Connect, so it can be easily integrated with your custom backend.

A user is able to sign in to a Firebase app by either using FirebaseUI as a complete drop-in auth solution or by using the Firebase Authentication SDK to manually integrate one more of the sign-in methods into the application. For our application we used the firebase SDK. By doing this we were able to choose which sign in methods we wanted to add to our application. In early development of we decided to use just email and password authentication in later development we then added social login options such as facebook. Firebase Authentication also handles sending password reset emails. The firebase SDK also comes with a reset password feature which sends the user an email with a link to reset there password. This is a good security measure as the user will need to have access to their personal email account before they can reset their password.[18]

3.5 Heroku

In our application, we needed somewhere to deploy our application. After a lot of research and experimentation, we decided to use Heroku[19]. Heroku hosts our Node server with our RESTful API. Our Heroku deployment is then connected to mLabs which is a fully managed cloud database service that hosts MongoDB databases and connects them to services like Heroku [20]. This then allowed us to run the application with out any intervention.

This was the most difficult part of the project to get working as we had a lot of trouble getting our database to properly connect with Heroku but after extensive research and a lot of testing we figured out that we needed to point the node.js server to the build of the application and then add the URL of our Heroku instance in replace of the localhost URL.

3.6 GitHub

GitHub is a web-based collaboration platform for software developers, delivered through a software-as-a-service (SaaS) business model which allows you to host and review code along with managing projects. Github first launched in 2008 and was founded on Git which is an open source code management system created by Linus Torvalds to make software builds faster.[21]

Git works by storing source code from projects and tracking all changes made to that code to a repository. Repositories can be made either public or private so developers can share their code. It is a great tool for developers when collaborating projects as it provides users tools for managing changes from different developers. We found this to be greatly beneficial in our development as we took advantage of the branches fracture. This feature basically allowed us to work on a separate part of the project and then upload that part to a branch on Github. This allowed the other person to then see those changes and even download the source code on their machine so they could run it and see the changes. This meant that we didn't have to be constantly pushing code to our master branch as this would frustrating if the code didn't work or broke another part of the project that was working.

GitHub works by using git commands to push projects up to its website [22]. The process behind this is straight forward. First the user needs to use git bash to navigate to the folder they want to upload. This folder then needs to be initialized by using the command:

```
$ git init
```

This command creates an empty Git repository which has a `.git` directory with subdirectories for objects, refs/heads, refs/tags, and template files. A HEAD file is also created and this file references the HEAD of the master branch.

The next command you have to use is:

```
$ git add .
```

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit.

Once the files are added you need to run:

```
$ git commit -m""
```

This stores the current contents of the index in a new commit along with a log message from the user describing the changes. And then finally run:

```
$ git push origin master
```

to push all the files to the master branch of your repository.

Other features that GitHub has to offer are its ability to fork, pull and merge from someone's repository. A fork is essentially a copy of a repository that allows developers to make modifications without affecting the original code. This is similar to the branch feature that we mentioned above. The only difference is a branch is apart of your current repository whereas a forked project is a complete copy of someone else's repository. If the developer would like to share the modifications, they can send a pull request to the owner of the repository. The owner can then decide after reviewing the modifications if they would like to pull the modifications into the repository. They then have the option to accept the modifications and merge them with the original repository.[23]

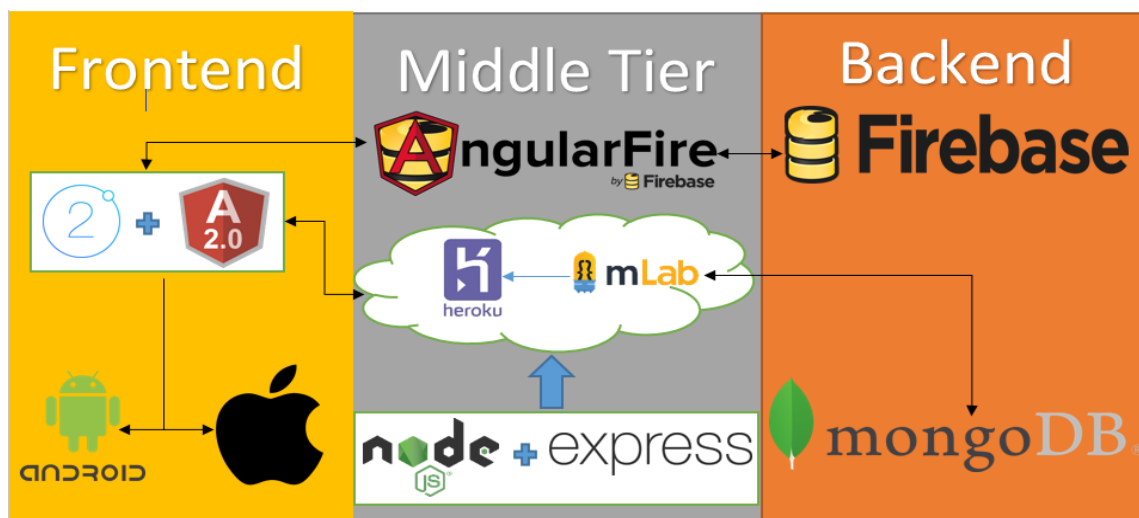
Another feature of Github that we found to be very helpful is the commit history. Everytime you make a change to your code and push it up to Github you are creating a commit. A commit is Githubs way of tracking the changes you have made to the project since your last commit. We found this useful because we could easily see what the other person has pushed up and exactly what they added or removed. You are also able to download the project from a commit at any point, so for example you added features to the project that you no longer want you can revert back to any point in your commit history. We used this feature on a few occasions and It saved us a lot of time as a result.

Chapter 4

System Design

4.1 Architecture

The architecture of this project is divided into 3 different tiers of technologies. The frontend tier containing the Ionic application, the middle tier containing our node server which contains our endpoints for our REST API. The middle tier also consists of a Heroku deployment which serves our node server online and AngularFire which is used for connecting the app to our Firebase databases for user authentication. Our backend is our data tier which contains the MongoDB database which stores all of our reviews created by the users. This database is in turn connected to an mLabs database which links to our Heroku deployment. The backend also contains our firebase database which stores all our User authentication details for our Login system. A diagram of our architecture is shown below:



4.2 Front end

In the image above to the left, you can see the technologies that consist of our front end. We used Ionic 2 to with Angular 2 to create a cross-platform application that would run on both Android and IOS. As you can see in the diagram our Ionic 2 application is connected to our Heroku deployment which hosts our REST API. The app sends a GET request to get the JSON data from the API to display our reviews inside the app. The app can also send a POST request to the API when a new review has been created on the app and also a DELETE request when a review has been deleted from the app. When the application development was completed we were able to use the Ionic CLI to build or application .apk for both platforms using the following command:

```
$ >cordova build.
```

Since the Ionic app is our frontend then it of course also contains our User Interface for our app which will be discussed in further detail in the following sections.

4.3 Middle Tier

In the diagram above in the middle tier, you can see most of our technologies reside in this tier. This is the main logical side of our application and it is a very important part in bringing all our technologies together to complete our app.

4.3.1 Node server

The most imported part of this entire tier is or Node server. With node, we created a file called server.js and defined our REST API endpoints. We also defined any modules necessary for the application. Our node server connects to our mLabs deployment and translates the data into JSON format so it can be sent back and forth between the node server and the database and also so it can be sent back and forth from the node server to the Ionic application.

Module definition

```
var express = require('express');
var app = express();
var morgan = require('morgan');
var bodyParser = require('body-parser');
```



```
var methodOverride = require('method-override');
var cors = require('cors');
var mongoose = require('mongoose'),
```

Api Endpoints

```
app.get('/api/reviews', function(req, res) {...}
app.post('/api/reviews', function(req, res) {...}
app.delete('/api/reviews/:review_id', function(req, res){...}
```

Connection to our mlab

```
mongoose.connect('mongodb://jason:Finalyear17@ds145780.mlab.com:45780/
heroku_tmdt9wp4');
```

4.3.2 Express

Express is used in our application in conjunction with our node server in order to build the server as a node application. This allows us to run the server and handle all API requests and gather the JSON data and apply it to our app. Express also allows our Heroku deployment to run our application online for use in a browser by finding the latest build of the project in our www directory and pushing it to our Heroku deployment. By doing this we have also pushed our API for our reviews to Heroku for our Ionic app to connect to.

4.3.3 Heroku and mLabs (Cloud Hosting)

In order for our application to access our node server, we needed to be able to host the server online. Mobile devices are not able to run Node applications, therefore, our only way to connect to our server is to get it hosted online, otherwise, the reviews on the app would not appear. Heroku and mLabs combined worked well together when it came to solving this problem. We initialized a git repository inside our app and pushed the Node/Express app to the server. We then connected our local database to an mLabs deployment which allowed for our hosted API to now interact with our online database. Our review data can be viewed at the following URL to verify our API has been hosted.

<https://fathomless-chamber-74986.herokuapp.com/api/reviews>

Using the above link we were able to connect our Ionic app to our now online Node server. This means the API can now be used by the application on mobile devices. An example of how we connected them is shown below:

```

this.http.get('https://fathomless-chamber-74986.herokuapp.com/api/reviews')

this.http.post('https://fathomless-chamber-74986.herokuapp.com/api/reviews'
, JSON.stringify(review), {headers: headers})

this.http.delete('https://fathomless-chamber-74986.herokuapp.com/api/reviews/'
+ id)

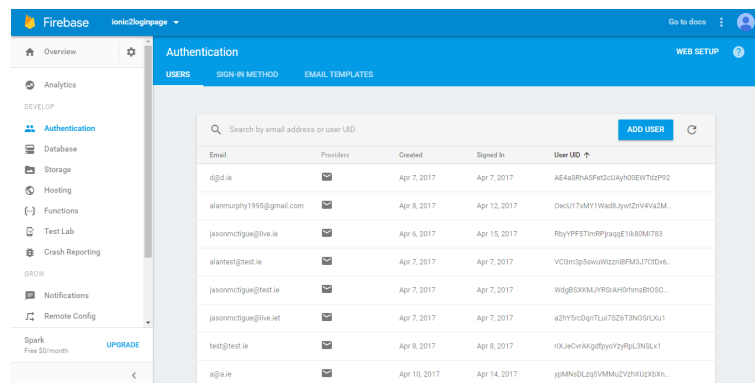
```

4.3.4 AngularFire2

Angularfire 2 is a library created for connecting Firebase 3 to Angular 2. We use this for our authentication services. AngularFire2 contains functions and classes which help connect our app to our Firebase database containing all our User information. AngularFire allowed us to create functions with firebase so users can sign up to the app, login, or if they have forgotten their password an email will be sent to them so they can reset it.

4.4 Back end

We have two database services in the backend of our application. The firebase database which contains all information of our users who can log in with their email and password or with Facebook. An example is shown below:



The screenshot shows the Firebase Authentication console. The left sidebar contains navigation links for Overview, Analytics, Database, Storage, Hosting, Functions, Test Lab, Crash Reporting, Notifications, Remote Config, and Spark. The main area is titled 'Authentication' and has tabs for USERS, SIGN-IN METHOD, and EMAIL TEMPLATES. The USERS tab is active, displaying a table of users with columns for Email, Providers, Created, Signed In, and User UID. A search bar at the top allows searching by email address or user UID. An 'ADD USER' button is in the top right corner of the table.

Email	Providers	Created	Signed In	User UID
d@d.ie		Apr 7, 2017	Apr 7, 2017	AE4a3RPA5P4t2CU4yH00EW10pP92
alanmurphy1995@gmail.com		Apr 8, 2017	Apr 12, 2017	0acU17AMy1Wad8JyWZv4V9a2M...
jasconnectgus@live.ie		Apr 6, 2017	Apr 15, 2017	RbyYFSTInRPyaqE1h80MI783
alantest@test.ie		Apr 7, 2017	Apr 7, 2017	VC0m3p5ewuW0z2nBPM3J7CDv6...
jasconnectgus@test.ie		Apr 7, 2017	Apr 7, 2017	W6g8X0XMYR5AH0hm8B0SC...
jasconnectgus@live.ie		Apr 7, 2017	Apr 7, 2017	a2NY5nDqTlU762zT3N89LXu1
test@test.ie		Apr 8, 2017	Apr 8, 2017	rXJwCvAKgd9y07y2yRPL3NLSLx1
al@ie.ie		Apr 10, 2017	Apr 14, 2017	y6MhGDLz3VMMu2V2HfU2BKH...

Our mongodb contained data on our reviews which users can create. This mongodb instance was connected to a mLab deployment in order for our hosted node application to connect to it. Mongoose provides a straightforward, schema-based solution to model your application data. We used a review model which consisted of

- Title

- Description
- Rating

Our model was designed on what we felt users would want to input into their reviews. The title is simple the place the user is at then description is used for writing comments about that place and finally the user can give the place a rating out of 100. The model represented in our code is as follows:

```
var Review = mongoose.model('Review', {  
  title: String,  
  description: String,  
  rating: Number  
});
```

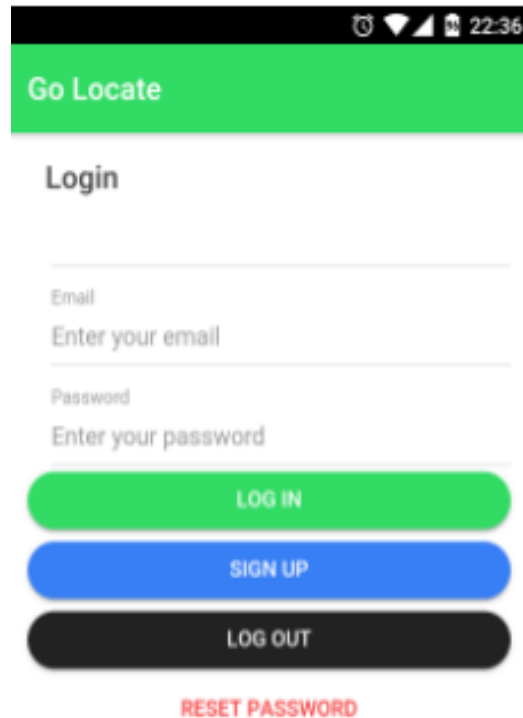
We were then able to connect this database to mlabs which uses collections to store its data. Mlabs automatically creates an id number from the mongodb database that it is connected to and it assigns it to the review that is pushed up. The design of the mlabs database is shown below:

```
{  
  "_id": {  
    "$oid": "58c190ae6a34ad2228dece35"  
  },  
  "title": "Eyre Sqaure",  
  "description": "A cool place to hang out in the center of Galway City",  
  "rating": 91,  
  "__v": 0  
}
```

4.5 Ionic Application

With the architecture discussed above we can see how they were used to form our app. In this section the individual pages will be discussed.

4.5.1 Login Page



Go Locate

Login

Email
Enter your email

Password
Enter your password

LOG IN

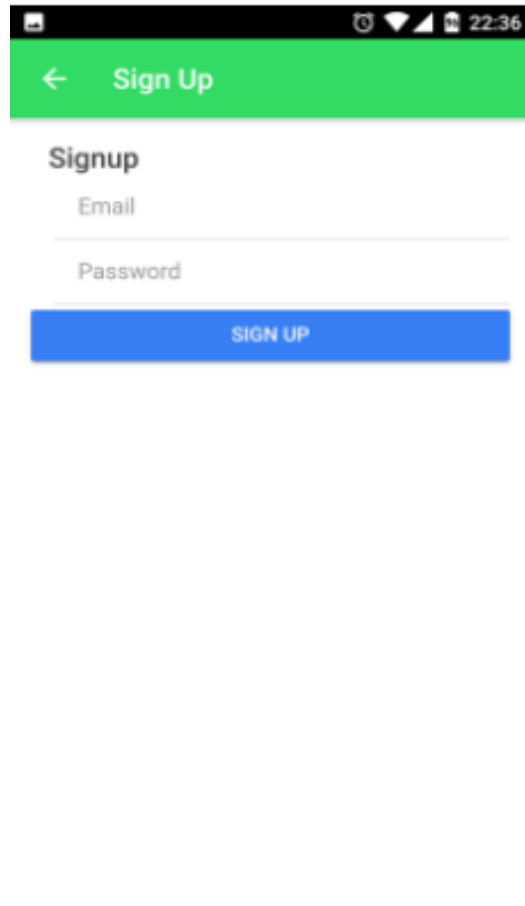
SIGN UP

LOG OUT

RESET PASSWORD

This page allows the user to log into the app using our firebase authentication. The user enters their email and password and presses login. When the login button is pressed the user details are compared to details in the firebase database to see if it's a valid user. If the information entered is correct the app navigates to the main map page. If the information is invalid and an error message is shown and the user cannot navigate to the main part of the app. There is also a signup button which navigates to the signup page, a forgot password button which navigates to the forgot password page. and a logout button which changes the state of the current user to being logged out.

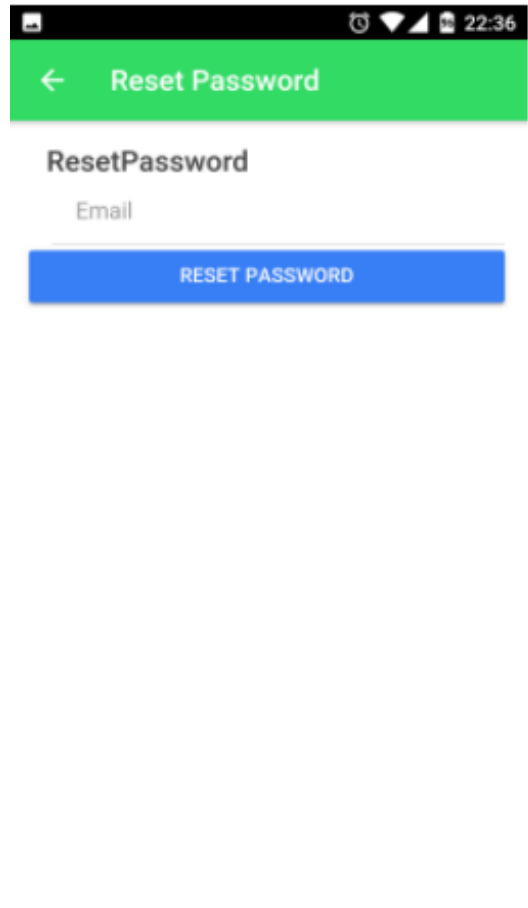
4.5.2 Sign Up Page



The screenshot shows a mobile application interface for a sign-up page. At the top, there is a green header bar containing a white back arrow on the left and the text "Sign Up" in white. Below the header, the word "Signup" is displayed in a bold, black font. Underneath "Signup", there are two input fields: "Email" and "Password", both in a light gray font. A blue button with the text "SIGN UP" in white capital letters is positioned at the bottom of the form area. The background of the page is white. The status bar at the very top shows the time as 22:36 and various system icons.

The signup page allows users to enter their credentials to register for use of the app. When the email and password are entered they are validated to make sure they are correct. If they are the details are stored in the database and the app navigates to the home page. If the details aren't valid the user is told what details need to be changed in order to sign up for the app.

4.5.3 Reset Password Page



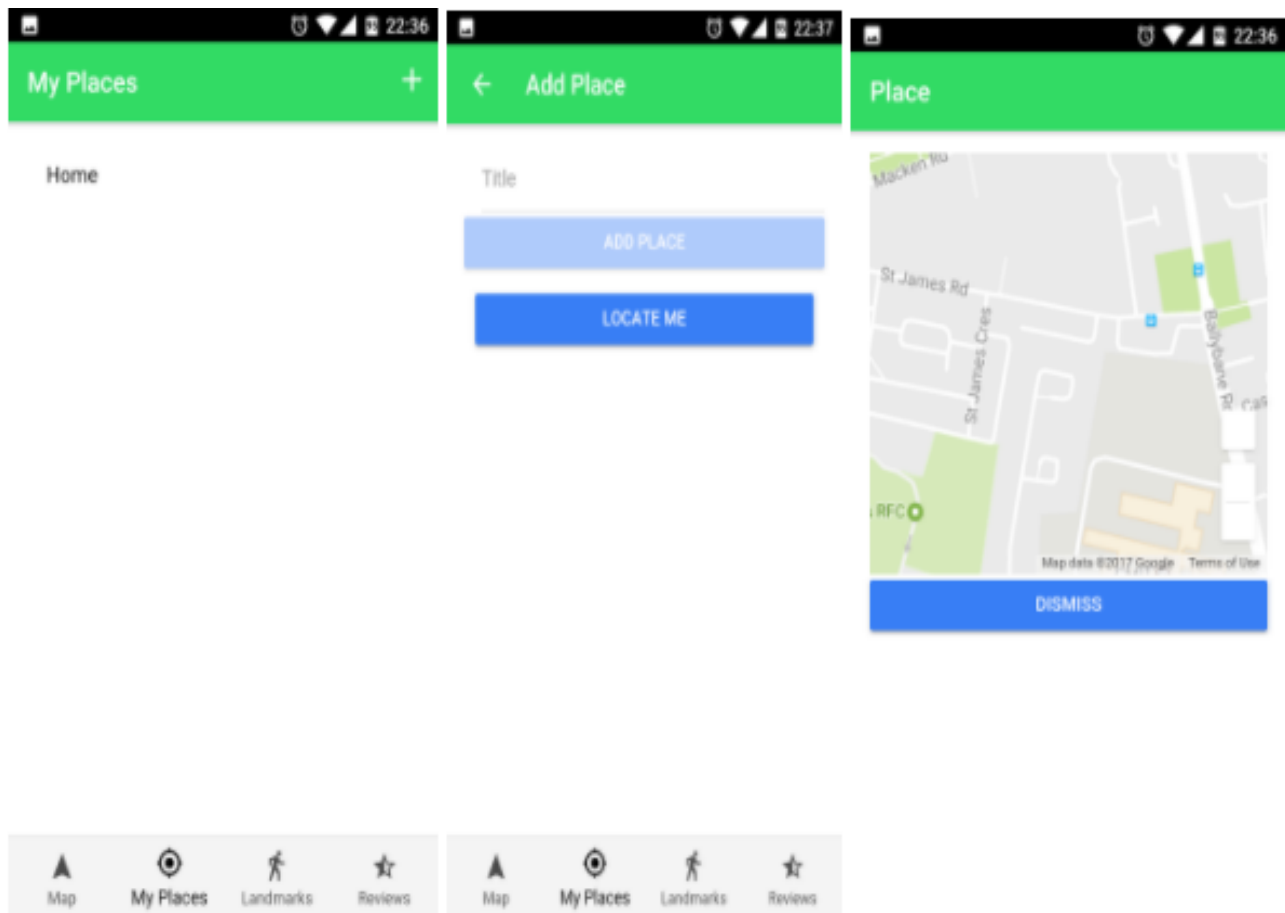
On the reset password page the user can enter their email that they used to register to the app and firebase will send them an email they can use to reset their password.

4.5.4 Map Page



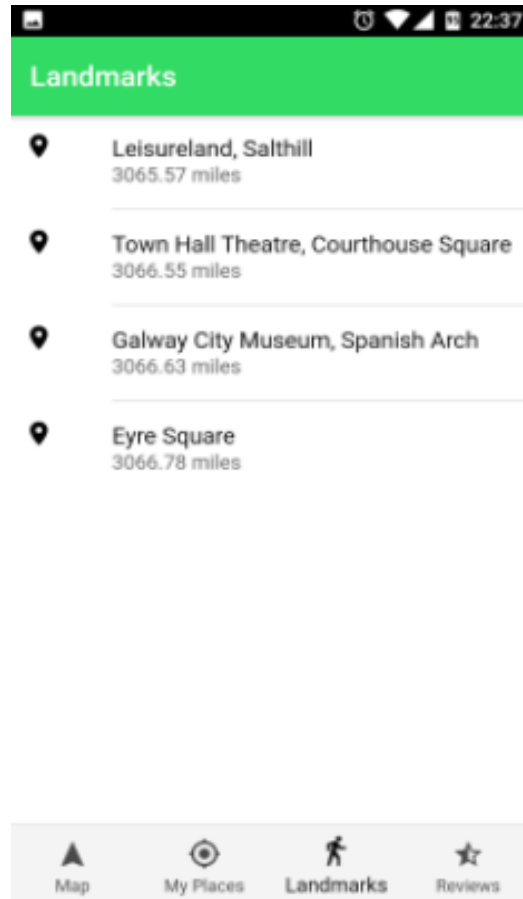
When the user first navigates to the map page they will be asked to share their location. Once they do the map will zoom in on the user's current location. This page uses the google maps API. When the user taps on the map a marker is placed which display how far away they are from that location. Street view is also available with this map. The logout button at the top of the page will logout the current user and return them to the login page. The four tabs at the bottom are used to go to the other pages in this app which are my places page, the Landmarks page and the Reviews page.

4.5.5 My Places



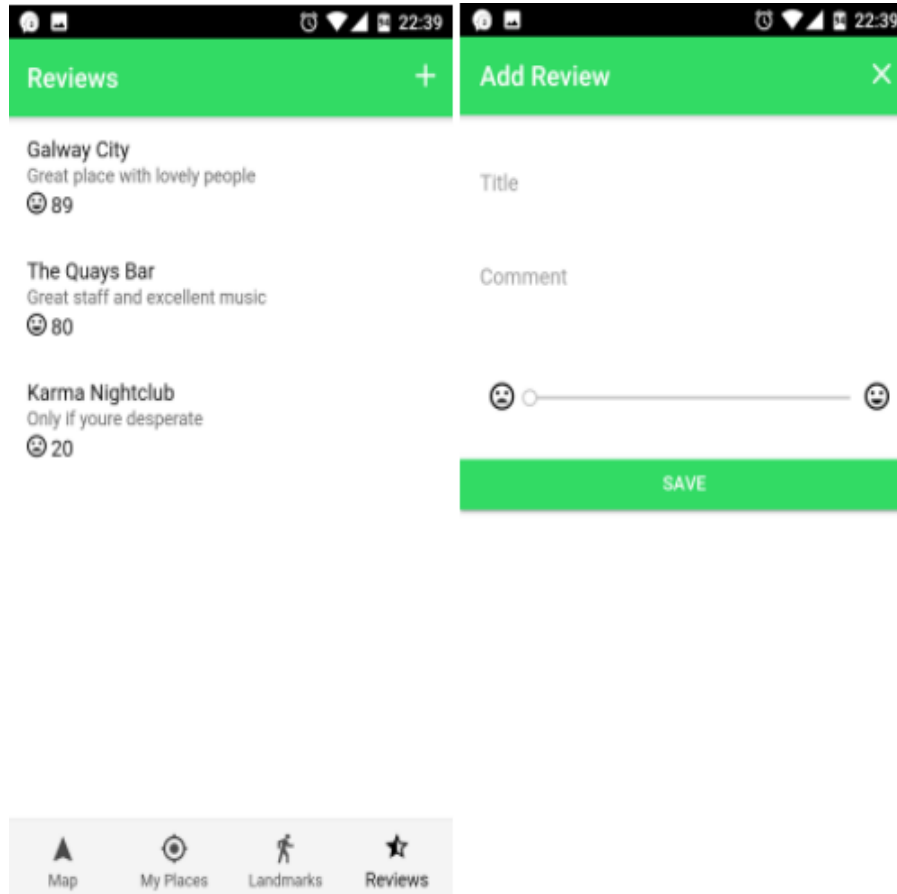
In my places page of the app, the user can save their current location and name it. The location will then be saved in local storage on the user's device. To add a location the user must click on the plus icon at the top of the screen. This will navigate them to the Add places page. Here the user must name their place to enable the add place button but before they press this they must save their location by pressing to locate me button. This button gets the latitude and longitude of the user and saves it. The app then translates this into a map for the user to view. They can then dismiss the location and return to their list of places.

4.5.6 Landmarks Page



In the landmarks page, the user can see information about the area around them and some popular locations. These locations are hardcoded into the app and are displayed by markers on the map page.

4.5.7 Reviews Page



In this page, in the app, the user can view, create and delete reviews about locations in their area. The list of reviews is pulled from our hosted Node server which contains the REST API which sends JSON data to the app which is translated into the list you see above. If the plus symbol at the top of the app is selected the user is navigated to the add review page. Here the user must enter a title for their review, a description and give a rating and then press save. Once this is done the app sends JSON data to our node server hosted on Heroku. This information is then used to store the review in the database. Now, this review will be added for every user using the app. If a review is swiped to the left a review can be deleted. This deletion also takes place for all users of the app.

Chapter 5

System Evaluation

As mentioned in the introduction of this paper our main objective when we initially set out developing this app was to learn a whole new set of skills. Primarily using the MEAN stack to develop a cross platform mobile application. Our main objective in terms of functionality of the application was for users to be able to post reviews or just general comments about the events around them and save them to the application and also be able to delete them again. We felt like we achieved this goal as we were able to create a page on our application that allowed users to do exactly that. Our next goal was to have a feature where the review page would be hosted online. Again we achieved our goal here as we were able to connect our mongodb database to Heroku. This meant that all reviews would be saved here and any changes will reflect on everyone's application. Our login authentication was handled by Firebase Auth the end user app was built in ionic framework which is easily consuming the data from an API, and the synchronization is very smooth and is working effectively.

5.1 Testing

To be sure all parts of the above mentioned works correctly you need to run unit tests on your application. Testing is an important task of any software project, without testing, programmers could easily introduce more bugs to a project. It's extremely important to test each section of a software project before releasing a new version of the application. Creating automated tests for your application is a great way to improve the quality of your code and protect against code breaking. For testing our application we used the Jasmine framework which is commonly used for writing tests in ionic and karma is then used to run these tests.[24][25] Karma tests primarily through

the following three functions: `describe`, `it`, and `expect`.

- `describe()` defines a suite of tests (or “specs”)
- `it()` defines a test or “spec”, and it lives inside of a suite (`describe()`). This is what defines the expected behaviour of the code you are testing.
- `expect()` defines the expected result of a test and lives inside of `it()`.

An example of a test we done in our application was a test that made sure the app loaded correctly on the login page. An example of the code used for this is shown below:

```
describe('Component: Root Component', () => {
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [MyApp],
      providers: [
      ],
      imports: [
        IonicModule.forRoot(MyApp)
      ]
    }).compileComponents();
  }));
  beforeEach(() => {
    fixture = TestBed.createComponent(MyApp);
    comp    = fixture.componentInstance;
  });
  afterEach(() => {
    fixture.destroy();
    comp = null;
  });

  it('is created', () => {

    expect(fixture).toBeTruthy();
    expect(comp).toBeTruthy();

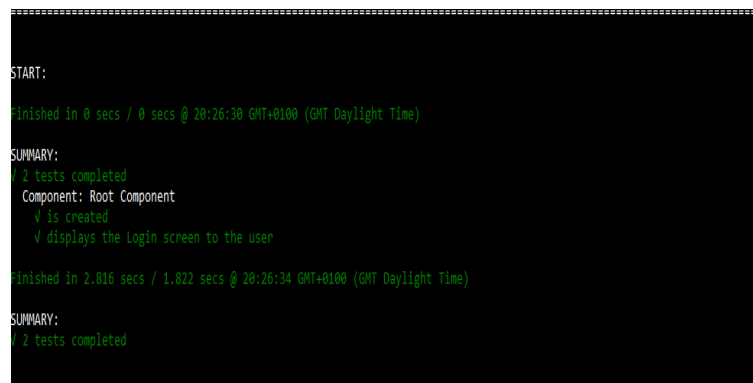
  });

  it('displays the Login screen to the user', () => {
    expect(comp['rootPage']).toBe(LoginScreen);
  });
});
```

This is a test we ran by using the command

```
$ npm test
```

This gives feedback in the terminal on whether the test passed or failed and if it succeeded in doing its given task. An example of a test we did is shown below:

A terminal window with a black background and green text. It shows the output of an npm test command. The output includes a 'START:' line, a line indicating the test finished in 0 seconds, a 'SUMMARY:' line, and two test results: '✓ 2 tests completed'. The first test is 'Component: Root Component' and the second is '✓ is created'. The output also shows the test finished in 2.816 seconds. The terminal text is as follows:

```
START:
Finished in 0 secs / 0 secs @ 20:26:30 GMT+0100 (GMT Daylight Time)

SUMMARY:
✓ 2 tests completed
Component: Root Component
✓ is created
✓ displays the login screen to the user

Finished in 2.816 secs / 1.822 secs @ 20:26:34 GMT+0100 (GMT Daylight Time)

SUMMARY:
✓ 2 tests completed
```

We also tested the application on an android device and found all the features to work relatively smoothly. One bug we did come across was that the map wasn't receiving an internet connection. After some research, we found out that we needed to add a plugin that allowed google maps to the internet when being used on a device. This plugin then fixed our issue.[26]

5.2 Limitations

Most software projects come with limitations and our project is no different. The main limitation for us is the fact that google maps and the reviews part of the application need a constant internet connection in order to work. We looked at possibly implementing a CouchDB technology however we felt like it wouldn't really solve our problem as the main point of the application is that it tracks your current location [27]. We did however include a page that stores places you visited to local storage so that aspect of the application is still functional with out an internet connection. Other limitations of the project include an issue we had with the list that locates you not adding a marker to the screen. Google maps api only allows you to use its map feature once in an application so when the second time it gets call it causes the main google maps page to stop showing.

Chapter 6

Conclusion

In Conclusion, it has been extremely interesting and highly rewarding working on this application. We faced a few problems during the development of this project but we feel like we handled them quite well. Our first problem was that the fact that we changed technologies mid-development. This required some adjustment but was also a great learning experience as we had to adjust quickly to the change. Our second major problem was that one of our team members wasn't contributing as much to the project as they should have been. We had to have a long discussion with this team member and they agreed to contribute more. However, the workflow from this member decreased again thus was slowing down our workflow. Our team member eventually decided that they were going to leave our group and defer the rest of the year. This was difficult for us as we had to try and scale back the project while also implementing the work they should have been doing. Despite these setbacks, we still gave it our best and were able to implement all the objectives we set out to. In our introduction, we set our self five main objectives that we wanted to complete for this project. They were as follows:

- Create an application using Ionic 2 and the MEAN stack.
- Have a secure login page for users.
- Implement a map page where users could see their current location, save their location and see nearby landmarks and points of interest.
- A section where users are able to post reviews or general comments
- A feature where these reviews would then be seen immediately by other users.

Our first objective was successfully achieved when we implemented the MEAN stack into our ionic 2 application. The use of MongoDB, Express, Angular and node gave our application a more complex feel when developing, as these technologies are widely used together in industry. These technologies were also brand new for us so it was challenging at first to get to grips on how they all worked together, but to have fully implemented them in this project was very satisfying for us. Another challenge we faced with these technologies was the fact that we originally had planned to use just firebase but after a discussion with our supervisor he advised we changed as using just firebase wouldn't have been enough for a project at this level. We are glad we listened because we feel that we wouldn't have gotten a lot out of using just that technology.

We completed our second objective of having a secure login page by using firebase authentication. As we had previously done some work with firebase we decided to carry this aspect over as we found it to be very secure. This allowed us to store usernames and passwords in an online database and also have a function which allowed users to reset their passwords. The reset function was very secure as the user would have to enter their email address and if that email address matches an email address on our firebase database they would then receive an email with a reset password link.

Our third objective was successfully completed when we used a google maps API which allowed the user to see their current location when they logged in and also display an array of markers that signified an event or point of interest. This was a feature we really wanted to implement because we feel that it gives a professional look and feel to the application. As an added feature to this section, we also added a list page where users could see the events or points of interest and see how far that place is from their current location.

Our fourth objective was to add a page where users could post reviews and comments about an event or location. This part was achieved using the mean stack and was one of the harder parts of the project to implement. As mentioned above the technologies were new for us so it was challenging, however, we wanted this project to challenge us as we wouldn't have learned anything from doing a project that was easy and already familiar to us.

Our final objective and defiantly the hardest part of the application to implement was having somewhere to store the reviews online. For this, we used Heroku which was difficult to get connected to our application. To go about this we had to host our application on a website called mlab which is then connected to our local MongoDB database. We then connected our mlab instance to Heroku and after a lot of research and trial and error it eventually worked and the applications reviews now update for all users.

Getting this part of the project working was probably the most rewarding part as neither of us had connected a database to Heroku before and we spent a lot of time researching how to do this.

An extra feature we added into the application was the my places section. This section essentially allows the user to save their current location from the map to a list on their device. This was a feature we thought people would like as they can then look back at places they have visited and see it displayed on the map. There is also a lot of room for improvement in this application but one of our main goals for this project was to learn a variety of new technologies and we feel like we have achieved that.

As a whole, we were extremely happy with how this project turned out. We learned a lot more than we thought we would in terms of the technologies we used and how we implemented them. We will also take valuable lessons from this experience such as better planning when deciding the technologies that are going to be used. The biggest lesson we will take from this experience is how to work effectively in a team. Despite the fact that one of our members wasn't contributing we dealt with the situation and didn't let it affect us too much. Working then in a group of two we felt we worked really well together we were able to set ourselves targets and the majority of the time we met our targets. We also managed to meet our project deadline. Overall these lessons will stand to us in the working world and this experience has defiantly better prepared us for exactly that.

Chapter 7

Appendices

GitHub: https://github.com/Smurfffy/FinalYearProject_MEANIonic2

Bibliography

- [1] M. Stack, “<https://www.sitepoint.com/introduction-mean-stack/>.”
- [2] NoSql, “<https://en.wikipedia.org/wiki/nosql>.”
- [3] . Gen, “<https://www.mongodb.com/press/10gen-announces-company-name-change-mongodb-inc>.”
- [4] MongoDB, “<http://searchdatamanagement.techtarget.com/definition/mongodb>.”
- [5] Express, “<https://expressjs.com/>.”
- [6] NodeJS, “<https://nodejs.org/en/>.”
- [7] Angularjs, “<https://docs.angularjs.org/guide/di>.”
- [8] W. is AngularJS, “<https://docs.angularjs.org/guide/introduction>.”
- [9] A. features, “<https://code.tutsplus.com/tutorials/5-awesome-angularjs-features-net-25651>.”
- [10] R. Api, “https://en.wikipedia.org/wiki/representational_state_transfer.”
- [11] RESTful-API, “<http://searchcloudstorage.techtarget.com/definition/restful-api>.”
- [12] I. Framework, “<https://ionicframework.com/>.”
- [13] I. . vs 2, “<https://www.joshmorony.com/7-reasons-why-ionic-2-is-better-than-ionic-1/>.”
- [14] J. Morony, “<https://www.joshmorony.com/>.”
- [15] Typescript, “<http://whatpixel.com/is-typescript-worth-learning/>.”
- [16] Microsoft and google collaborate on typescript, “<https://techcrunch.com/2015/03/05/microsoft-and-google-collaborate-on-typescript-hell-has-not-frozen-over-yet/>.”

- [17] F. Authentication, “<https://firebase.google.com/docs/auth/>.”
- [18] Firebase, “<https://firebase.google.com/>.”
- [19] Heroku, “<https://www.heroku.com/about>.”
- [20] mlab, “<https://mlab.com/>.”
- [21] GitHub, “<https://github.com/>.”
- [22] G. commands, “<https://education.github.com/git-cheat-sheet-education.pdf>.”
- [23] G. Operations, “<http://searchitoperations.techtarget.com/definition/github>.”
- [24] Karma, “<https://karma-runner.github.io/1.0/index.html>.”
- [25] Jasmine, “<https://jasmine.github.io/>.”
- [26] C. Network, “<https://github.com/apache/cordova-plugin-network-information>.”
- [27] CouchDB, “<http://couchdb.apache.org/>.”