Student Accommodation App – Built using the Ionic 3 Platform

Patrick Moran

Gerard Naughton

Andrei Petruk

B.Sc.(Hons) in Software Development

April 17, 2018

Final Year Project

Advised by: Daniel Cregg

Department of Computer Science and Applied Physics

Galway-Mayo Institute of Technology (GMIT)

Contents

List of Figures

Abstract

For our final year project, we wanted to develop an app that would challenge and hone our technical skills whilst creatively addressing a current issue. Students are particularly vulnerable in the current accommodation crisis across Ireland, as they are restricted by budget, time-pressures, proximity to campus and are commonly excluded from most properties. In addition, many students requiring accommodation are often arriving alone to a new area. The search for accommodation is typically time-consuming and stressful and comes with repeated rejection. We conceived of an accommodation app called Digs tailored specifically for students that would simultaneously provide a forum to empower students to grasp more renting opportunities by clubbing together. Working as a team and following an agile methodology, we developed an app that allows users to view student accommodation ads and post accommodation listings. We created a three tier application, using Mongo Db and Firebase as our Data Tier, NodeJS for our Logic Tier and Ionic 3 for our Presentation Tier. Adding specific features such as, adding college campus as a search criterion and creating a message board for students to group together and find accommodation helped us complete our objectives by gearing our app specifically for students.

Authors:

- Patrick Moran
- Gerard Naughton
- Andrei Petruk

Acknowledgements

,

Figure 1: Digs Logo

We would like to acknowledge our project supervisor Daniel Cregg for his help and supervision during the creation of Digs.

We would also like to thank John Healy, Head Lecturer of the Applied Project and Minor Dissertation module.

We would also like to thank Sinead Howard, 2nd Year Digital Media Student at GMIT, for designing the logo for our app as seen in Figure 1.

Chapter 1

Introduction

When choosing our project we wanted to pick something that was relevant to current everyday life. We wanted a project that also highlighted our existing skills and allowed us to learn new skills and develop as part of a team. With this criteria in mind we started brain storming ideas for our project.

We eventually decided on a App called Digs. Digs would be a Student Accommodation app. We recognized the current accommodation crisis throughout Ireland and thought it would benefit students and landlords to have a specific letting app for the student population. Also for students it was quiet disheartening to apply for accommodation and be rejected for being of student status. Currently students in Ireland can find themselves in hostel accommodation for up to 2 or 3 months before finding proper accommodation.This is quiet common when renting for students and with our app they would only be applying for lettings specific to them.

Having a student accommodation app as our starting point we started to research other apps and websites in the accommodation letting/renting category. The main one we focused on was the Daft App[1]. Daft would be the most popular accommodation app in Ireland and it provided us with a great deal of insight on what our app would need to have and also on what Daft lacks for the student market.

We wanted a app that had essentially the same functionality as Daft but which was geared more towards student. We did this by focusing on the college campuses and not specific cities. We also created a forum for students to get in touch with one another, to either get a house together or to get someone for a room in the house they are already renting. These would compliment the normal functionality of a accommodation app such as, publish ads (either rooms or properties), search lists (rooms or properties), view rooms and properties and edit and delete your personal ads. This would also require log in and registration functionality.

After deliberating over the objective of our application, we then turned to what technology we would use. We wanted to use a array of technologies to show our capabilities but also gain more knowledge and become better programmers over all. We settled on a 3 tier structure containing a Ionic 3 presentation tier (front end), NodeJs logic tier (middleware) and Mongo/-firebase database data tier (back end).

From what we hoped to achieve from creating this app was the knowledge of new technology, personal growth through team building and of course to produce something of substance and something we could be proud of.

Chapter 2

Context

The general context of our application revolves around students having a platform which is geared towards finding accommodation. Students will be able to search through available listings, advertise their own rooms for rent and in addition, letters can use the app to advertise full properties to students. Listings can be created by advertising a room or property, with a series of criteria. Some examples of criteria would be proximity to college campus, room/property type, location, price, contact information and a description. Along with this information, images can be taken with the user's device camera or uploaded directly from their device which is then linked with that listing. Students looking for accommodation can also search through available listings based on the following criteria:

- Price
- Which college campuses are close
- Room/Property Type
- Parking Availability

2.1 Objectives

The main objective of our application is to help students who may be struggling with the current accommodation crisis in Ireland. We also wanted to make it easier for students and landlords, who are looking to rent or let. The following is a list of the main pages in our application along with the objectives for each page.

1. **Login/Register Page** Our application has a login and a register page. The register page allows the user to securely register with our application while the login page allows the user to securely login to our application. Once logged in, the user has access to extra features not accessible to unregistered users.

2. **Forgot-Password Page** The objective of the forgot-password page is to allow a user who has forgotten their password to enter their email address. They will then receive an email containing a link which will allow them to create a new password.

3. **Home Page** The objective of the homepage is that it is a base of navigation for the application. It displays the total number of both properties and rooms that are currently listed. The homepage will also display if a user is logged in or not.

4. **List of Rooms/Properties Page** The objective of both these pages is to display all the currently listed rooms and properties to the user. Showing the main criteria on each room/property so the user can make an informed decision.

5. **Create a Room/Property Page** The objective of both these pages is to allow the user to create a room or property listing and then publish it. This will allow the user to showcase their room/property using images, filling in form details and their own personal description.

6. **Message Board** The objective of the message board is to have a place where students can communicate with each other, helping themselves find other students that are looking for accommodation

7. **Search** The objective of the search page is to allow students to refine the number of listings based upon criteria they specify.

8. **My Ads Page** The objective of the My Ad's page is to allow a user to view a list of their published ads. In here they can delete these ads or they have the option to edit their published ads.

2.2 Project Links

Link to Repository

- https://github.com/gerardnaughton7/4th-Year-Final-Year-Project

2.3 Chapters Review

This paper is broken down into different chapters, ranging from planning, design and development. The following sub-sections give a brief overview of each chapter in this paper.

2.3.1 Methodology

In this Chapter we discuss some of the methodologies we used during the various stages of creating Digs. In this section we discuss Agile, version control, testing and sprints.

2.3.2 Technology Review

In the Technology Review chapter we review the various technologies we used in our project. We review back and front end technologies, development tools and version control.

2.3.3 System Design

In this chapter we give a detailed explanation of the overall system architecture and design of Digs. We explain why we chose these specific technologies and how we implemented them into our system design.

2.3.4 System Evaluation

In this chapter we evaluate our system in the areas of Robustness, Testing, Scalability, Results against Objectives and Limitations.

2.3.5 Conclusion

In the conclusion we summarise our project against our goals and objectives. We review our our application from the various stages and speak about possible future development.

Chapter 3

Methodology

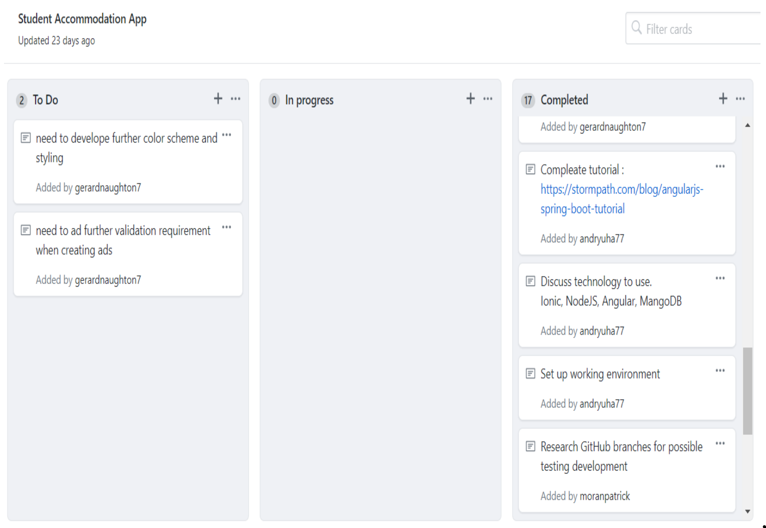
In this chapter we discuss the methodologies used in our project. A methodology is just a way to plan and control the development process of a piece of software. There are various methodologies to choose from including Extreme Programming, Rapid Application Development or Waterfall but for this project we use Agile as our main methodology.

3.1 Agile Development

During the life cycle of our project we attempted to use an Agile like approach to the research, design and implementation stages of the project. In the initial stages of the project we discussed various methodologies we could use, for example Waterfall but we decided on Agile because of waterfalls lack of flexibility. Agile offers continual improvement, flexibility and incremental delivery of the software.

During the research and development process of our project we used a Scrum like approach. Scrum is an Agile method in which a development cycle is carried out in what are known as sprints. Section 3.4 and 3.5 contains more information on each sprint of the research and development phase of the project.

Throughout the life cycle of the project we held weekly meetings. In these meetings we would plan and discuss the features of our application. Before each sprint of our project, we used these meetings to plan our sprint cycle. Results from these meetings were then added as to-dos and added to a GitHub projects page. GitHub projects board help us prioritise and organise our work load. We found this to be an especially useful tool. Once an issue was created it was added into the To-do phase. Then, when we started working on a problem it got added to in-progress, then finally into the completed section once the task was done. A snapshot of our GitHub Projects page can be found in Figure 3.1


Student Accommodation App
Updated 23 days ago

Filter cards

2 To Do

need to develope further color scheme and styling
Added by gerardnaughton7

need to ad further validation requirement when creating ads
Added by gerardnaughton7

0 In progress

17 Completed

Added by gerardnaughton7

Compleate tutorial : https://stormpath.com/blog/angularjs-spring-boot-tutorial
Added by andryuha77

Discuss technology to use. Ionic, NodeJS, Angular, MangoDB
Added by andryuha77

Set up working environment
Added by andryuha77

Research GitHub branches for possible testing development
Added by moranpatrick

,

Figure 3.1: GitHub Projects

We also had weekly contact with our project supervisor. Our supervisor was instrumental in advising us on our progress and helping us with issues with the project.

3.2 Version Control

Throughout the life cycle of our project we used GitHub. GitHub is a hosting service for version control. We created a GitHub repository, adding all the members of the team to work as collaborators. We found GitHub to be an extremely useful tool in the research and development of Digs. Even though we primarily used GitHub to manage source code, we also took advantage of some collaboration features that GitHub offers. We used wikis to track what we learned in the research stages of our project. A wiki is a website where multiple users can modify content directly from their browser. We used GitHub's task management tool when a task was set and then used this tool to track that tasks progress. We found this tool particularly useful as it helped break down the workload into small manageable tasks.

As a team, working with source code on GitHub proved to be its most valuable feature. GitHub allows all members of the team to see all the commits made to the repository. We can see every update, previewing every change that was made and even rolling back a commit if necessary.

3.3 Testing

For our application, while we did not choose to use a framework (i.e. Junit) to test our application, we did however used both white and black box testing.

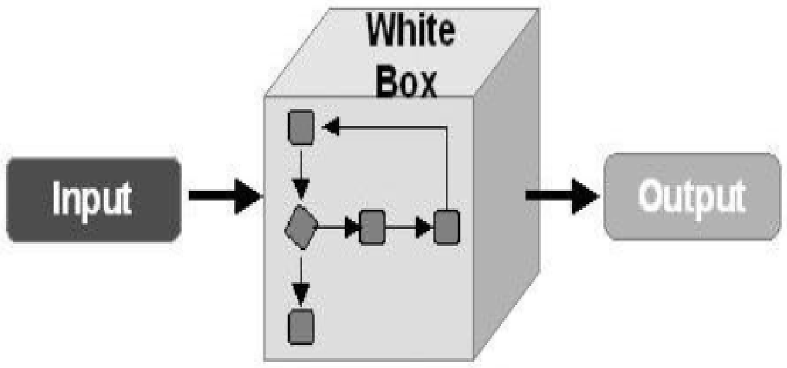
3.3.1 White Box Testing

White box testing is a way of testing software where the tester can see the internal workings of the software. White box testers have full knowledge of the internal makeup of the software and are usually software developers themselves. In our application, to white box test, one team member would write a feature (e.g. login) and then another team member would test that functionality. That team member would present input, follow the path through the code and then examine the output. A diagram of white box testing can be found in Fig 3.2.

3.3.2 Black Box Testing

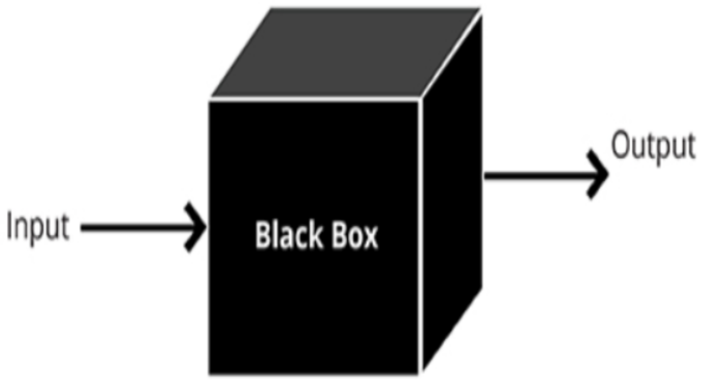
Black box testing is a way of testing a piece of software without knowing the internal functionality of the software being tested. A black box tester has no knowledge of the internal design, structure or implementation of the software and are often not programmers themselves. In our application, to black box test, we would ask friends to use our application e.g. asking them to login with Google and then reporting their experience. With this type of testing, the tester is trying to break your application looking for faults. We found this type of testing especially useful as it gave us an insight into the way a user would use our application, something we previously would not have thought of when developing Digs. A diagram of black box testing can be found in Fig 3.3.

The following two sections contain information about the various sprints we undertook in the research and development cycles of our project.



,

Figure 3.2: White Box Testing



,

Figure 3.3: Black Box Testing

3.4 Sprints

Each of the sprints we completed during the research of our project gave us the opportunity to learn as a team and adapt to the various needs of the project. Each sprint represents a vital aspect of the application, i.e., creating three tier architecture, accessing native phone capabilities through Cordova and learning how to integrate all these features together. We prioritized the research sprints in the order as follows.

3.4.1 Sprint 1 Review King

This was our first sprint and it was worked on by all three members in our team. Review King is an application we found on a blog by Josh Morony [2] in which he explains the process of building a simple review app using three tier technology. On the front end a user can write a small review and then add a score. That information is then sent to a node server which stores the information in a MongoDB database. We felt it was very important that all three members understood the process involved in connecting these three layers together so we all worked on this sprint together.

3.4.2 Sprint 2 Image Gallery

This was a relatively small sprint and was completed by Patrick. The plan for this sprint was to create a small app which displayed images in a gallery style format. Using a plugin for ionic called ionic-img-viewer [3], an image is made larger when it is clicked. Images can also be moved left and right in a gallery style look while incorporating zoom functionality.

3.4.3 Sprint 3 Image Back-end

The next phase of the project was figuring out a way of using a mobile devices camera to take a picture, upload that picture to a node server and then save a reference to that image in a MongoDB database. To do this we split this process up into two sprints. Both Sprint 3 and 4 were worked on by all three members in our team as it was important for all members to understand the process involved in implementing these features. The first sprint involved setting up the back end. We achieved this functionality by following a blog by Devdactic [4] where we set up our NodeJS back-end. Once completed, our backend had the capability to upload and store images. This functionality was tested using software called Postman [5]. Postman is a HTTP client service which helps developers test their API with various HTTP requests.

3.4.4 Sprint 4 Image Front-End

This was the second of a two-part sprint in which we developed the front end of a mobile app using the Ionic framework. This front end would then be able to upload and save a photo to a NodeJS server running locally on our machines. We achieved this functionality by following part 2 of a blog by Devdactic [6].

3.4.5 Sprint 5 Login/Register

In this sprint, implemented by Patrick, a small mobile application was developed where login and registration with firebase was implemented. We are using Firebase in our Digs app to securely authenticate our users with email and password registration and login. The user also has the option of logging into our app with Google Plus.

3.5 Development Sprints

3.5.1 Sprint 1 Create Pages

From the result of team meetings, we had decided on the number of pages our application would require. In the first development sprint, we created all the pages that we would need for our application. The sprint was completed by Gerard and Andrei and this formed the base skeleton of the Digs app.

3.5.2 Sprint 2 Menu Navigation

The next step was to implement a menu into our app. The result of this sprint was so that each page could now be navigated to, by either the main menu or by a button on a page. This sprint was completed by Gerard.

3.5.3 Sprint 3 Login/Registration

It was now time to implement the registration and login functionality to Digs. Patrick implemented this sprint as he had previous knowledge of implementing this functionality with Firebase from a previously completed research sprint.

3.5.4 Sprint 4 Implementing the Back End

In this sprint we implement our NodeJS server running on one of our AWS instances. This sprint was completed by Patrick and Andrei and in it we set up our API routes along with MongoDB to store text data in our database. The text data being stored would be the data from a created listing i.e. Room Type, Price, Location etc.

3.5.5 Sprint 5 Camera Functionality

This sprint was worked on by all three members of our team because taking and storing images is one of the prominent features of Digs. We first implemented the functionality on the front end. This allows a user to take a photo using the camera on their device or by selecting a photo from their devices internal storage. We then implemented our backend API to store those images mapping each image/s with its unique ad id. We then set up the functionality to send the full listing including images to our back-end which we set up in sprint 4.

3.5.6 Sprint 6 Displaying Listings

Once a listing was stored in our database, it was now time to display all the stored listings in our app. Gerard completed this sprint by successfully displaying a full list of properties and rooms including their images.

3.5.7 Sprint 7 Accessing Google Maps

To make it easier for the user to find the location of a listed property, we display to the user that properties location on the maps application installed on their device. This sprint, implemented by Patrick, uses a plugin from Cordova called launch navigator [7]. This plugin is used to navigate to a destination, in our case using an Eircode, by launching the native application apps on android, iOS and windows devices.

3.5.8 Sprint 8 Message Board

The next task, implemented by Andrei, was the Message board. The result of this sprint was the creation of a forum in which users can communicate with each other by creating and posting messages. Andrei first set up the backend to store these messages in MongoDB database and then displays the stored messages to the user. Message output includes the users name, message and date created.

3.5.9 Sprint 9 Validation

In this sprint Patrick and Gerard implemented form validation on the register, login, message board and create ad pages. For the matching passwords validation, we created a custom function which checks that the passwords are the same. For all other form validation, we use Angular directives to check that correct input is entered and that all fields are complete before the from can be processed.

3.5.10 Sprint 10 Search Functionality

One of the last features of our application is the search functionality. Implemented by Gerard and Patrick, users can narrow down the listings they see by applying some search criteria. A user can put a limit on the max price, choose which college campuses are close by, what type of room they are looking for, the number of rooms the property has or if the property has parking as an option.

Chapter 4

Technology Review

4.1 Data Tier

4.1.1 MongoDB

MongoDB is a cross-platform, document-oriented database that provides high performance and is easily scalabe. MongoDB is structured using collections and documents.

MongoDB is ideal for projects that involve working with very large amounts of data or where the requirements need to be scalable. MongoDB is also ideal where information is too complex, there is a need for real-time analysis or where a model of data does not need to be predefined as opposed to a relational database.

When comparing MongoDB to a relational database, the contents of a relational database are tables and MongoDB databases consist of collections. Each collection has its own unique name - an arbitrary identifier consisting of no more than 128 different alphanumeric characters and an underscore. Unlike relational databases, MongoDB does not use a table with data types. MongoDB is a document-oriented system in which the central concept is a document.

A document can be represented as an object storing some information. In a sense, it is like strings in relational database, where lines store information about the element. [8]

An example of a typical MongoDB document:

```
{
    "name": "Andrei",
    "surname": "Petruk",
    "age": "40",
    "college": {
        "name" : "GMIT",
        "year" : "2018",
        }
}
```

4.1.1.1 Document ID

For each document, a unique identifier called id is defined in MongoDB. When you add a document to the collection, this ID is automatically generated. However, the developer can explicitly set the identifier, rather than relying on the automatically generated ones, specifying the corresponding key and its value in the document.

This field must have a unique value within the collection. And if we try to add two documents with the same identifier to the collection, we will get an error.

If the identifier is not specified explicitly, then MongoDB creates a special binary value of 12 bytes.[9]

4.1.2 Firebase Authentication

This authentication service closely integrates with other Firebase services, uses industry standards such as OAuth 2.0 and OpenID Connect, so that it can be easily integrated with your backend.

The Firebase Authentication SDK provides methods for creating and managing users who use email and password to log on to the system.

According to the documentation, the listener receives notifications in the following situations:

- When the FirebaseAuth object completes the initialization and the user is already authorized in the previous session or is redirected from the input stream of the other authentication provider.
- When the user logs in (the current user is installed).
- When the user exits (the current user becomes null).
- When the access token of the current user is updated. This can happen if the token expired.
- The user re-authenticates.
- The user changes their password.

In this case, Firebase issues new access tokens and marks the old ones as invalid. For security reasons, when a user changes their password, that user is automatically logged out of each device they are logged into. [10]

4.1.3 NoSQL database

The rise of Big Data made an interest for a horizontally adaptable Data Management System. This prompted advancement of various types of Database Management System which all things considered go under NoSQL. NoSQL Databases are comprehensively separated into following sorts: Document, Key-value, Graph, Native object, Table type, Native XML and Hybrid Databases. All Relational Database Management Systems depend on a very strict model to adhere to, while each of the NoSQL databases takes after a distinctive model. NoSQL moves far from the robust institutionalized type of SQL database and empowers less complex information capacity arrangements. Therefore a NoSQL database is enhanced for the particular application.[11]

4.1.3.1 Advantages of NoSql

NoSQL databases are versatile, dependable and scalable. One of the main advantages of NoSQL is that it can handle unstructured data. Unstructured data can be word reports, messages, sound, video, or even interpersonal social network information. NoSQL databases can handle object-oriented programming that is flexible and easy to use.

To ensure quick execution, NoSQL databases don't rely in ACID (Atomicity, Consistency, Isolation, Durability) transactions in comparison to relational databases. [12]

4.1.4 Examples of NoSQL

- Key-Value Storage:

  MUMPS, CouchDB, FoundationDB, Redis, Aerospike, Dynamo, MemcacheDB, Riak, OrientDB, Fair Com c-treeACE, Redis.

- Apache CouchDB, MarkLogic, OrientDB, Clusterpoint, Couchbase, MongoDB.

- Columnar Storage:

  Vertica, Cassandra, Hbase, Accumulo, Druid.

- Graph Storage:
  Neo4J, OrientDB, Stardog, Allegro, InfiniteGraph, Virtuoso.

4.2 Logic Tier

4.2.1 NodeJS

NodeJs or Node was developed by Ryan Dahl in 2009. Node.js is a server-side platform based on the google V8 engine which translates JavaScript into machine code. Developed as a asynchronous event driven javascript runtime. This design makes it perfect for building light-weight, scalable network applications. Due to it non-blocking I/O calls not only allows to handle thousands of concurrent connections but also alleviates us from having to worry about locking/deadlock as we dont use threads.[13]

4.2.1.1 Asynchronous calls

Asynchronous calls are especially important for web servers. It is quite typical for modern web applications to have access to databases. While waiting for the result to return from the database, Node can process more queries. This allows you to cope with thousands of parallel requests with little overhead compared to creating a separate thread for each connection. [13]

4.2.1.2 npm

Node.js Package Manager is the package manager that is part of the Node.js. Now, instead of installing each library separately, we can run one command and install everything in one go.

```
npm install
```

When you run the command, npm will look for the package.json file in the current folder. If found, it will install each library from the list. [13]

4.3 Presentation Tier

4.3.1 Ionic

Ionic is a framework with a stunningly designed graphical interface, styled for various platforms, for maximum similarity to native applications. AngularJS and sass are taken as a basis for the framework, which brought them great popularity among developers, as these technologies reduce the threshold of entry to a minimum. The core of the framework is Cordova, so all the plugins of Cordova will work fine on ionic.

In addition to the platform and ready-made UI components, Ionic has a thoughtful command line interface that allows you to generate icons, splash screens, run the file in the browser with the app for debugging, and build applications with short commands in the console.

Also worth noting is the detailed documentation with examples.[14]

4.3.1.1 Ionic UI

Ionic has dozens of ready-made components, which are constantly being refined and replenished with new ones. You can fully read them on the documentation page.[15]

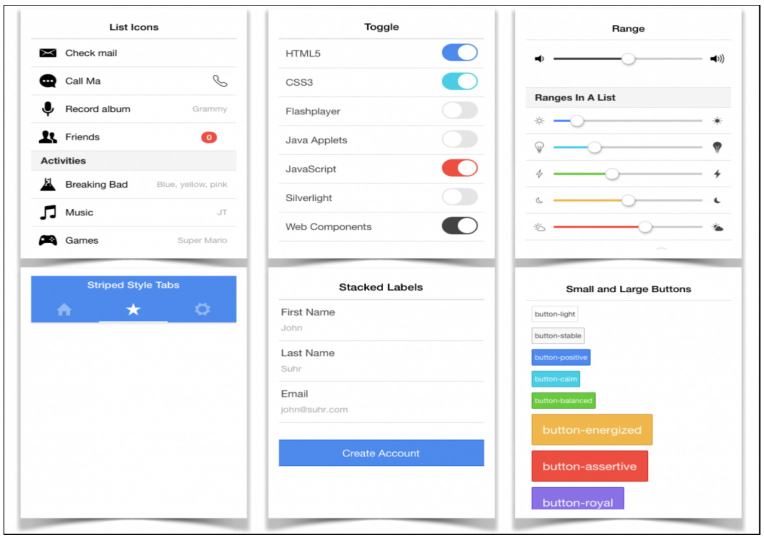


Figure 4.1: Ionic UI.

4.3.1.2 OS oriented styles

Ionic is not just a good interface, it is adapted to the specifics of various operating systems and is thought through to the smallest detail. Animations, icons, fonts, all this with the same code will look like the operating system under which the application is built (Android or iOS, and starting with the second version and Windows Phone).[16]

4.3.1.3 Cordova

Apache Cordova is a platform for developing open source mobile applications. It allows you to use standard web technologies such as HTML5, CSS3 and JavaScript for cross-platform development, avoiding the native development language for each of the mobile platforms. Applications are performed inside a wrapper targeted at each platform and rely on standard APIs to access device sensors, data, and network status. [17]

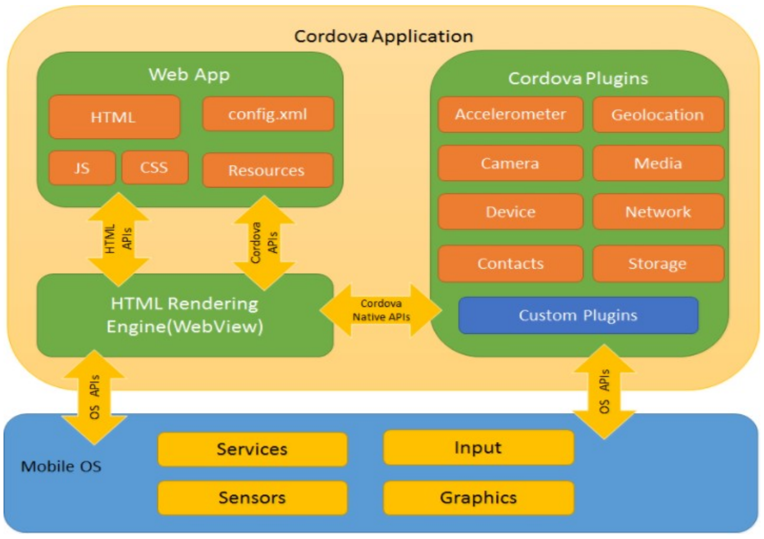


Figure 4.2: Cordova Application

Cordova is a platform that allows you to develop mobile applications on different platforms, by embedding the browser in a mobile application. In fact, your application is a mini-browser that shows one single site - your application. All resources can be placed in the distribution package of the application to accelerate the download, and you can download from the server if necessary.

By default, Cordova provides only basic browser capabilities that are available on this mobile device, but it allows you to extend the set of functions available in the browser by using plugins. Each plugin provides a unified interface that can be used from a browser on different platforms. And if the supported CSS / JS functions differently in each browser from the version of the operating system or platform, the Cordova functional tries to provide unified functionality for all supported versions of mobile OS.

Even though this approach has its limitations in the form of the need to consider differences in mobile browsers, it still allows you to create high-quality cross-platform applications. [17]

4.3.2 Angular

Angular represents a framework from Google to create client applications. First, it is aimed at developing SPA-solutions single-page applications. In this respect, Angular is the successor of another AngularJS framework. At the same time, Angular is not a new version of AngularJS, but a fundamentally new framework.

Angular 5 provides functionality such as two-way binding, which allows you to dynamically change data in one space, including templates, routing, and so on.

One of the key features of Angular is that it uses TypeScript as its programming language. Therefore, before starting work, it is recommended that you familiarize yourself with the basics of this language.

But we are not limited to TypeScript. If you want, you can write the Corner applications with languages such as Dart or JavaScript. However, TypeScript is still the main language for Angular.

One of the key elements of the application are the components. The component controls the display of the presentation on the screen. [18]

```
import { Component } from '@angular/core';

@Component({
    selector: 'my-app',
    template: `<label>Please enter Name:</label>
                 <input [(ngModel)]="name" placeholder="name">
                 <h1>Hello {{name}}!</h1>`
})
export class AppComponent {
    name: "";
}
```

Component class:

```
export class AppComponent {
    name: "";
}
```

For a class to be used in other modules, it is defined with the export keyword. In the same class, only one variable is defined, which stores a string as a value.

To create a component, you need to import the @Component decorator function from the @angular/core library. The decorator @Component allows us to identify the class as a component.

If we did not apply the decorator @Component to the AppComponent class, the AppComponent class would not be considered a component.

The decorator @Component takes as an argument the object with a configuration that specifies the framework, how to work with the component and its representation. [18]

4.4 languages

4.4.1 Typescript

TypeScript is backwards compatible and compiled in JavaScript. In fact, after compilation, the program written using TypeScript can be executed in any modern browser or used in conjunction with the server platform Node.js. The code of the compiler that translates TypeScript in JavaScript is distributed under the Apache license. Its development is carried out in the public repository through the GitHub service.

TypeScript differs from JavaScript in the ability to explicitly enforce static types, support the use of classes, as in traditional object-oriented languages, and support for connecting modules, which is designed to improve the development speed, facilitate readability, refactoring and reuse of code, search for errors at the stage of development and compilation. TypeScript also accelerates the execution of programs. [19]

4.4.1.1 Services

When Angular switched to typescript from javascript it moved away from using factories to using services.

To implement the service, we pre-announce the interface for the service that we are exporting. Then we create a class for the service that implements the service interface. And then we call the Angular method to add the service. [19]

```
module MyApp {

    "use strict";

    export interface IMyService {
        myMethod(param: string);
    }

    class MyService implements IMyService {

        static $inject = ["Dependency"];

        constructor(private dependency: IDependency) {
        }

        myMethod(param: string) {
            return this.dependency.field + param;
        }
    }
    angular.module("services").service("MyService", MyService);
}
```

The constructor parameters declared with private or public modifiers become class fields and get values corresponding to the values of the constructor parameters.

4.4.1.2 Components

Fragments of the same type of page layout can be conveniently combined into components. This allows you to simplify page layout, reuse the same code, facilitates refinement and improvement. It is enough to correct only the code or template of one component, and all the pages on which it is used will update accordingly.[19]

4.4.2 JavaScript

Javascript is a programming language with which web pages are given interactivity. It creates applications that are included in the HTML-code (for example, questionnaires or registration forms that are filled in by the user).

The uniqueness of this programming language is that it is supported by almost all browsers and fully integrated with them and all that can be done with it is done very simply. No other technology accommodates all these advantages together. To date, this technology is currently working on language Javascript2. [20]

```
// add two numbers
<script>
    var num1, num2, sum
    num1 = prompt("Enter first number")
    num2 = prompt("Enter second number")
    sum = parseInt(num1) + parseInt(num2) // "+" means "add"
    alert("Sum = " + sum)  // "+" means combine into a string
</script>
```

[21]

4.4.3 HTML

HTML is a hypertext mark-up language, which is used to create documents on the Internet (web pages). The web page is created using HTML that contains all the necessary elements. On the HTML page, you can place a simple text, highlight it in bold or italic, insert a link, a table, a numbered or unnumbered list, pictures, divide the text into paragraphs and sections and specify headings for sections. Also, on the HTML page, you can insert a form with text fields, buttons, select options from the list, checkboxes and radio elements. In HTML5, you can embed video and audio files in a page, draw with canvas, make simple animations with new tags. [22]

```
<html>
   <head>
      <title>My HTML page</title>
   </head>
   <body>
      <h1>Page title</h1>
     <p>Page description</p>
   </body>
</html>
```

4.4.4 SCSS

SCSS(Sassy Cascading Style Sheets) is a super set of css. So let me fisrst explain css as it covers alot of what Sccss does. CSS - cascading style sheets. In fact, they serve to separate the page structure and its content from its appearance.

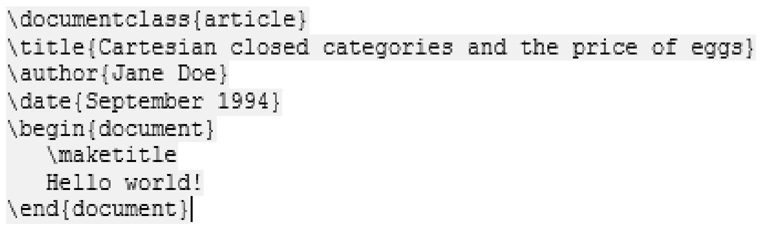
If the page is completely written in HTML, then each element of the code determines not only the content element of the page but also its way of displaying. For example, not only that there is a text "Hello" in such and such a place, but also that this text is highlighted in bold and red.

With the use of CSS code, everything happens a little differently. With the help of HTML, only the order of the content elements of the page and their classes are described. The corresponding classes are written in the CSS file. Each of them is assigned a set of properties. Now, when we assign a class to an HTML element, then all the properties of this class are applied to it. This greatly reduces the repetition of code. Now, when sites have many pages, you can not do without CSS. [23] Having covered CSS we now have the basics of SCSS. The only commanding difference between the two is that SCSS allows the use of declaring specific variables which you can use throughout the SCSS file. This saves the user a great deal of time and also allows for more uniform results on the html page. Here is examples of CSS and SCSS:

```
/*CSS*/
body {
  font-family: Verdana,
  Arial, sans-serif;
  }

/*SCSS*/
$magenta: #4ccbfce;
$margin: 22px;

.content-navigation {
  border-color: $magenta;
  color: darken($blue, 9%);
}

.border {
  padding: $margin / 2; margin: $margin / 2; border-color: $magenta;
```

4.4.5 LaTeX

LaTeX is a typing system based on a special scripting programming language. LaTeX is not so simple and intuitive as, for example, Microsoft Word, but once you have learned it you will not go back to Microsoft Word in the future. The fact is that LaTeX has long been a de facto standard for the collection of scientific articles, course and diploma papers, technical specifications, textbooks, etc. The main advantage of LaTeX is the completely identical appearance of the finished pages in all operating systems and this is unsurpassed up to now. In addition, the scripting language LaTeX is a universal language for the exchange of formulas. Mathematicians from different countries easily understand each other if they write formulas in this language. LaTeX goal is that it is better to leave document design for the document designers and to let authors get on with writing documents. In LaTeX, your ,input will look like this: [24]



```
\documentclass{article}
\title{Cartesian closed categories and the price of eggs}
\author{Jane Doe}
\date{September 1994}
\begin{document}
   \maketitle
   Hello world!
\end{document}
```

Figure 4.3: LaTeX.

4.5 Software and tools used for development

4.5.1 Sharelatex

ShareLaTeX is an online real time LaTeX editor which allows to work on text documents collaboratively without the need to setup additional software on your PC. It is an easy and convenient way to get started with LaTeX. [25]

We have chosen ShareLaTeX as we already had experience with it in our previous semester while using ShareLaTeX for our literature review assignment.

There is a lot of nice features in the premium version of ShaReLaTeX, like the ability to work on the same document for the group of people and possibility to upload your progress straight to GitHub. But we avoid the fees by sharing one account details between us and manually uploading our work to the GitHub repository.

4.5.1.1 History of document changes

ShareLaTeX saves the full history of all changes, so you can always track who made the change and when. Thanks to this you will always be aware of the work done by your collaborators. Errors occur regardless of whether you work alone or together with collaborators. You can always easily return to previous versions, which eliminates the risk of losing work or making undesirable changes. [25]

4.5.2 Visual Studio Code

Microsoft has released many development tools. But many developers will say that this is the best editor that Microsoft has built. Visual Studio Code working on three platforms, Linux, OS X and Windows. Without throwing all the functions of a mature Visual Studio IDE. Microsoft decided to rethink the approach on which the main toolkit of the programmer is built and started with the most important - the code editor. Visual Studio Code is an editor, but it has IDE functions that rely on extensions.

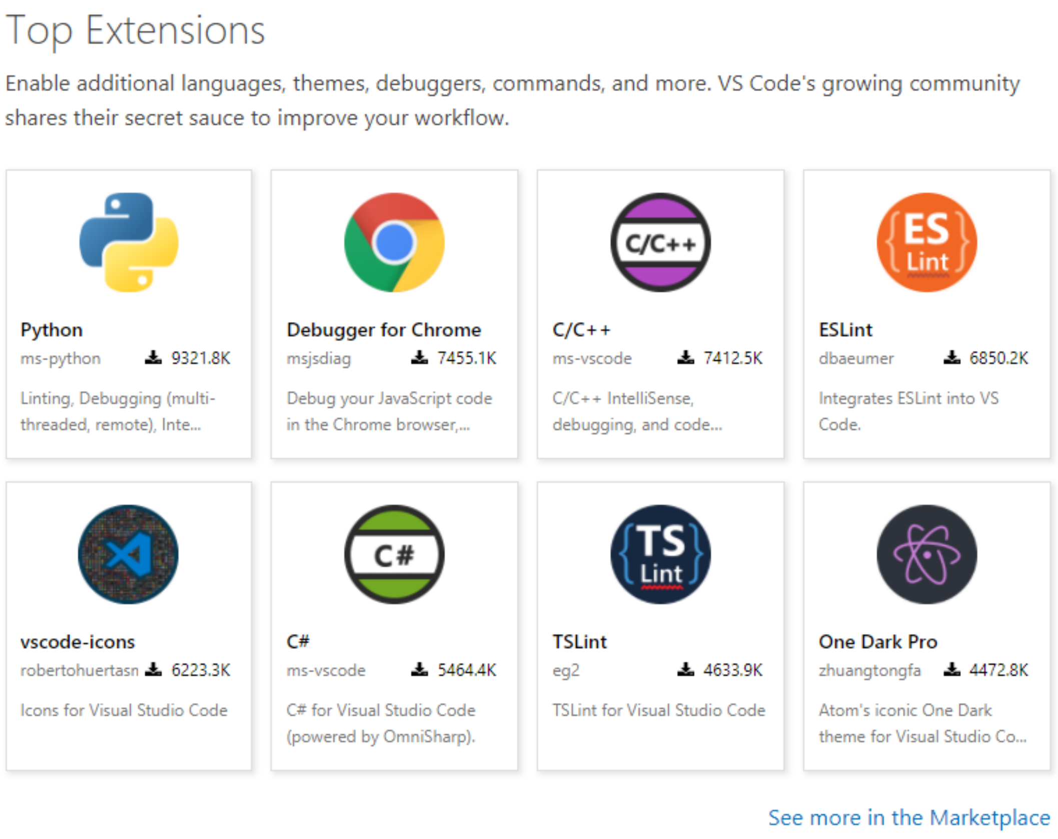
Now with Visual Studio Code is possible to use different languages such as JavaScript, TypeScript, C sharp, and so on. Also, you can use Visual Studio to create ASP.NET 5 or Node.js web projects, work with package managers npm and even carry out debugging. An excellent intellisense, support for code snippets, refactoring, navigation, multi-windowing, git support and much more. In some ways, Visual Studio Code even more convenient than the full version of Visual Studio and much lighter for the hardware requirements. [26]

4.5.2.1 VS Code extension

Without the Visual Studio Code extensions, it's only suitable for simple editing of files, for proper work you will need the accompanying tools that depend on your goals and tasks:

- git - version control system
- NodeJS (includes NPM) - a platform for creating scalable network applications
- Express - a framework for Node applications
- generator-aspnet - yeoman generator for ASP.NET 5 applications, run npm install -g generator-aspnet for installation
- hottowel - yeoman generator for fast creation of AngularJS applications, run npm install -g generator-hottowel for installation
- gulp - a toolkit for creating and performing tasks associated with the assembly of project tasks
- mocha - the framework for creating unit tests for JavaScript / Node
- bower - client package manager
- TypeScript - TypeScript language, adds modularity, classes and other nice things into your JavaScript code

- TypeScript definition manager - TypeScript definitions for popular JavaScript libraries, include support for IntelliSense in the VS Code [27]



Top Extensions

Enable additional languages, themes, debuggers, commands, and more. VS Code's growing community shares their secret sauce to improve your workflow.

Name	Publisher	Downloads	Description
Python	ms-python	9321.8K	Linting, Debugging (multi-threaded, remote), Inte...
Debugger for Chrome	msjsdiag	7455.1K	Debug your JavaScript code in the Chrome browser,...
C/C++	ms-vscode	7412.5K	C/C++ IntelliSense, debugging, and code...
ESLint	dbaeumer	6850.2K	Integrates ESLint into VS Code.
vscode-icons	robertohuertasn	6223.3K	Icons for Visual Studio Code
C#	ms-vscode	5464.4K	C# for Visual Studio Code (powered by OmniSharp).
TSLint	eg2	4633.9K	TSLint for Visual Studio Code
One Dark Pro	zhuangtongfa	4472.8K	Atom's iconic One Dark theme for Visual Studio Co...

See more in the Marketplace

Figure 4.4: Visual studio Code extensions

4.5.2.2 Files, folders and projects

VS Code works with files and folders containing the projects. In the simplest case, you can open the file for editing simply by running ./code index.html. A more interesting case is opening a folder. VS Code itself determines the type of project, depending on the contents of the folder. For example, if the folder contains the files package.json, project.json, tsconfig.json, or .sln and .proj files then the VS code includes many new features that include IntelliSence hints, code navigation, execution of commands and much more. [27]

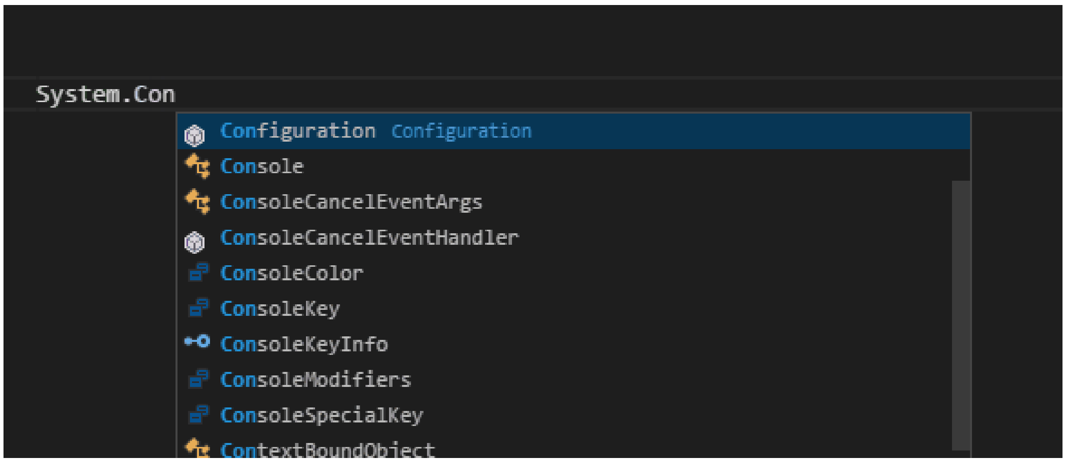


Figure 4.5: Visual studio Code IntelliSence

4.5.2.3 Command Palette

The most important tool for interacting with the editor in VS Code is the command palette. You can call it via the keyboard, by pressing the combination Ctrl + Shift + P. A lot of commands listed in the palette are also attached to the keys. [28]

4.5.2.4 Autosave

By default, VS Code works in the explicit save mode, which you can perform by pressing the Ctrl + S combination. This mode is compatible with most status monitoring tools. You can turn on the autosave mode by pressing Ctrl + Shift + P and type auto. [28]

4.5.2.5 IntelliSence Tips

Wherever you are in your code, clicking on Ctrl + Space will display the IntelliSence prompt. When typing the code, the editor will show it automatically. [28]

4.5.3 Postman

Postman is an API / web services testing tool that has a powerful HTTP client support. It has an easy-to-use query designer that allows you to write test cases and manage data and response time for effective testing and management of test cases API, simplifies the work with the API, supporting developers at every stage of its workflow and is available for Mac OS X, Windows and Linux.

Key features:

- Allows you to organize APIs in functions called Postman assemblies.
- Facilitates collaboration and sharing of API data and controls.
- Allows you to write logic tests in Postman Interface. [5]

In our project, we used postman for testing our back-end server functionality with our database. Most useful feature for us was the possibility of testing, at the early stage without having actual front-end application.

4.5.4 GitHub

GitHub is a hosting service which allows users to host, share and collaborate on projects together. It can be used freely for Open Source projects. Projects hosted on Github are called repositories. It is here users can view their projects, pull down a version of this project to their device or add to the project depending if you are either the owner of repository or a collaborator on the repository.

Collaborating on projects especially coding projects can get very complicated as others peoples work can conflict and disrupt the process. Luckily Github takes care of this with its distributed version control. Allowing the user to revert back to changes at any stage of the project, branch to different work trees and also merges branches back together.

In addition to being just a hosting project, GitHub is also a great social networking site for developers and programmers. It allows users to follow each other, subscribes to project updates, like them, giving them an assessment, etc. These functions allow users to receive updates for projects in which they are interested, or stay in touch with colleagues and employees. Github has been described as a portfolio for programmers work and many employers would ask potential employees for their Github profile link to review their work.

GitHub is not only used for programming and software development. It is also used by many other types of projects. For example, open source tutorials, documentation projects, training resources and other projects in which users can work together. [29]

Some sample examples of git commands:

- git add .

  Adds all modified and untracked files in the current directory

- git commit -m ”Commit message”
  To commit changes.
- git push origin master
  To send changes to repository.

Chapter 5

System Design

In this section, we will cover the overall design and architecture of our Application. We will do this through code snippets and visual aids to help you get an understanding of the Application design. The System Design chapter will be broken up into four parts. Data Tier represented by our Databases Mongo DB and Firebase, Logic Tier represented by NodeJS servers, Presentation Tier which will be represented by Ionic 3 App and our deployment of our app. The figure in 5.1 shows the overview of our Architecture.

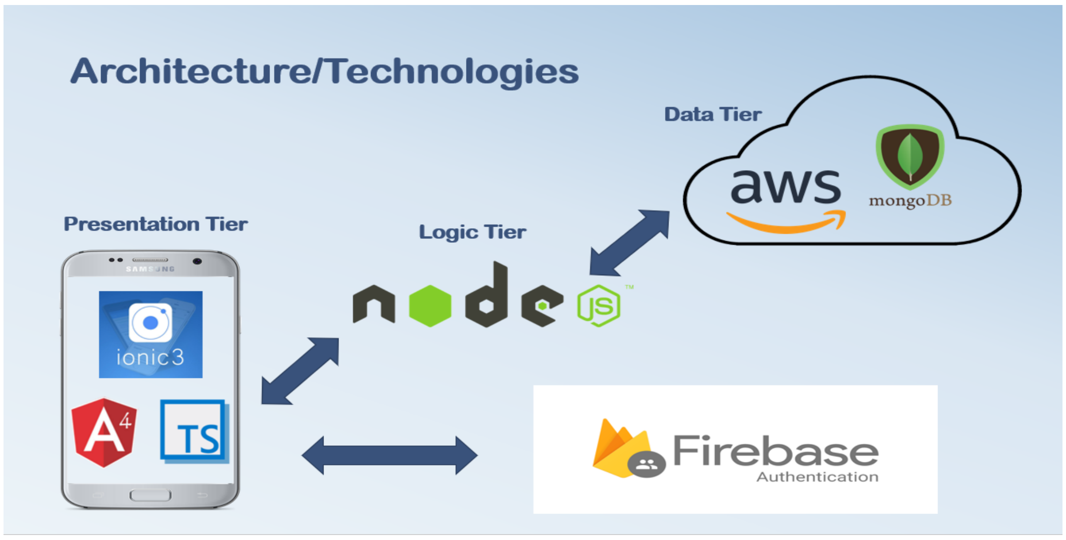


Figure 5.1: System Architecture.

5.1 Data Tier

The data tier represents the overall stored data and gives the ability to store, access, update and delete certain data. For our application we have two databases. One database is used to store our user accounts and the other is used to store our users published ads and forum posts.

5.1.1 FireBase

We chose Firebase to handle our user accounts as it was established in the ionic community for being a good database for login and registration functionality. We chose to use this free Authentication API service as it offered ready made functionality such as password reset, email and password authentication and Google login.

5.1.2 Mongo DB

We chose Mongo DB to store our clients ads, host ad images and forum messages as it offers a lot of flexibility. When developing a product no matter how much planning one does, requirements can always be subject to change. With this in mind we chose Mongo as it would allow us to change the structure of database as the project evolved. Mongo allows you to download and host your own Mongo DB server for free. This download comes with its own terminal which you can use to query your database which was very effective tool during development.

5.1.3 Hosting

To Host our Mongo DB we used AWS known as Amazon Web Services. AWS provides students with a free account which we have previously used for other project in our course. We created a AWS instance located on a server in Ireland and configured the in-bound and out-bound rules to allow communication between our node server and Mongo DB database. We simply could run our Mongo DB server on our instance by bringing up a terminal and navigating to our Mongo's bin folder and use the command below:

```
C:\Program Files\MongoDB\Server\3.4\bin>mongod
```

5.1.4 Collections

Our Mongo DB would be structured into 4 collections.

- properties - stores property models
- rooms - stores room models
- messages – stores message models
- images – stores and hosts our images

Our Mongo DB structures our models as JSON object in each collection. Here is a property JSON object example:

```
{
    "_id": "5aba78dab36ff80e38c85347",
    "UID": "test@gmail.com",
    "AdID": "2ec810ba29cc91c0ffc19913ef75397f",
    "PropertyType": "House",
    "SingleBeds": 3,
    "DoubleBeds": 1,
    "TwinBeds": 0,
    "EnSuite": 0,
    "NoRooms": 4,
    "Eircode": "H951CF",
    "Address": "Galway city ",
    "LocationDes": "City Suburbs",
    "Price": 2300,
    "Availability": "2018-09-27",
    "Email": "patrickmoran121@gmail.com",
    "Phone": 861234567,
    "Contact": "Phone",
    "Description": "Description house here",
    "Parking": "Yes",
    "Date": "2018-03-27T17:01:14.403Z",
    "__v": 0,
    "ImagesUrl": [
        
         ↪  "http://54.73.1.214:3000/images/5aba78d836ceeb168c045761"
    ],
    "College": [
        "GMIT",
        "NUIG"
    ]
},
```

I will cover how we connect Mongo DB to our NodeJS server in the next section.

5.2 Logic Tier

Logic tier refers to your apps business logic. It his here we program your applications ability to manipulate data. These include creation, storing , updating and deletion of Data. To handle our business logic we use NodeJS. NodeJS was the obvious choice for our 3 tier architecture not only does it link up well with our Mongo DB but also is easily integrated with our Ionic 3 presentation tier. NodeJS with its light-weight nature and its scalability due to its non-blocking I/O calls allowing tens of thousands of concurrent connections made it more than capable for our application.

Using Java script for our logic tier and using type-script for our providers in our presentation tier made it really easy to transition between the two because of their similarities. For our Logic tier we host two NodeJS servers on our AWS account. Our digs server would handle our ad data and our images server would take care of saving and hosting our images.

5.2.1 Connecting the Data Tier

For creating the connection between the Data Tier and the Logic Tier we used Mongoose. Mongoose is a JavaScript library which allows us to define a schema with strongly typed data. Mongoose then allows you create models which map to a MongoDB Document via the Models schema definition.[30]

Not only does Mongoose allow you to define strong data models and the ability to manipulate them before saving, it also offers functions to create an easy connection to your database and to validate, save, delete and query your data. In the code snippets below from our digs server you will find examples of this: Here we show how we connected to our Database using Mongoose:

```
var mongoose = require('mongoose');
var morgan = require('morgan');
var bodyParser = require('body-parser');
var methodOverride = require('method-override');
var cors = require('cors');

// Configuration
mongoose.connect('mongodb://localhost/digsdb');
```

Here we can see our room data model and how we declare each data type:

```
// Models
var Room = mongoose.model('Room', {
    UID: String,
    AdID: String,
    RoomType: String,
  College: [String],
  Address: String,
    Eircode: String,
    LocationDes: String,
    Price: Number,
    Availability: String,
    Email: String,
    Phone: Number,
    Contact: String,
    Description: String,
    Parking: String,
  ImagesUrl: [String],
  Date: Date,
});
```

5.2.2 Connecting the Presentation Tier

For the moment we are just dealing with the Logic Tier side. In the Logic tier we have configured our digs server to run on port 8080 and the images server to run on port 3000. It is on these ports where our servers will send and receive traffic to our presentation layer.
Using our IP address, port number and routing address, the presentation layer will be able to send http requests to our servers. Depending on the routing address provided will determine what action the server will take on our data tier. In the next section I will go through some of the functions and queries Mongo and Mongoose provide for accessing and querying of data.

5.2.3 Routing and Access of Data

Here we will show some of our routing and mongoose functions which allow to us to query our data tier Mongo DB.

Here we can see a Http Get request being performed on our Rooms Collection. This function is accessed though its routing address "/api/rooms" which can be seen below:

```
// Get rooms
app.get('/api/rooms', function(req, res) {

  console.log("fetching rooms");
  Room.find(function( err, rooms) {

    if (err)
      res.send(err)

    res.json(rooms);
  });
});
```

This function will return an array of room objects from the database and send them to our Ionic application in JSON format.

In our next function we can see our server performing a POST http request on our Properties collection. As you can see below the function is accessed through a similar routing address "/api/properties" except this time we use app.post rather than get.

```
// create properties and send back all properties after
 ↪  creation
app.post('/api/properties', function(req, res) {

  console.log("creating properties");
  let d = Number(req.body.DoubleBeds);
  let s = Number(req.body.SingleBeds);
  let t = Number(req.body.TwinBeds);
  let e = Number(req.body.EnSuite);
  total = d + s + t + e;

  Property.create({
    UID: req.body.UID,
    AdID: req.body.AdID,
    PropertyType: req.body.PropertyType,
    SingleBeds: req.body.SingleBeds,
    DoubleBeds: req.body.DoubleBeds,
    TwinBeds: req.body.TwinBeds,
    EnSuite: req.body.EnSuite,
    NoRooms: total,
    College: req.body.College,
    Eircode: req.body.Eircode,
    Address: req.body.Address,
    LocationDes: req.body.LocationDes,
    Price: req.body.Price,
    Availability: req.body.Availability,
    Email: req.body.Email,
    Phone: req.body.Phone,
    Contact: req.body.Contact,
    Description: req.body.Description,
    Parking: req.body.Parking,
    ImagesUrl: req.body.ImageURL,
    Date: req.body.Date,
    done : false
  }, function(err, property) {
    if (err)
      res.send(err);

    Property.find(function(err, properties) {
      if (err)
        res.send(err)
      res.json(properties);
    });
  });

});
```

As you can see above we create a Property object with the data received from body of our http request and post to our properties collection. After we do this we then perform a get request and return the new refreshed array of property objects to our Ionic application.

We now have shown how we can retrieve and create data from our Mongo DB. Next we shall manipulate existing items in our database through delete and update.

Here we demonstrate our delete function on a specific room object in our Rooms collection. As you can see below we use a delete request and our routing address passes through a parameter like so “/api/rooms/room-id”.

```
// delete a room
app.delete('/api/rooms/:room_id', function(req, res) {
  Room.remove({
    "_id" : req.params.room_id
  }, function(err, room) {

  });
});
```

Through this we can remove the room object using the mongoose remove function and the rooms unique identifier passed through in the parameter.

Next we show our Update function, here we use a put request. To perform this action we receive the object ID we wish to update as a parameter and also receive the new updated object from the body of the http request.

```
// update Room ad
app.put('/api/rooms/:UID', function(req, res) {

  let UID = req.body.UID;

  Room.remove({
    "_id" : req.params.UID
  }, function(err, room) {

  });

  Room.create({
    UID: req.body.UID,
    AdID: req.body.AdID,
    RoomType: req.body.RoomType,
    College: req.body.College,
    Eircode: req.body.Eircode,
    Address: req.body.Address,
    LocationDes: req.body.LocationDes,
    Price: req.body.Price,
    Availability: req.body.Availability,
    Email: req.body.Email,
    Phone: req.body.Phone,
    Contact: req.body.Contact,
    Description: req.body.Description,
    Parking: req.body.Parking,
    ImagesUrl: req.body.ImageURL,
    Date: req.body.Date,
    done : false
  }, function(err, room) {
    if (err)
      res.send(err);

    Room.find(function(err, rooms) {
      if (err)
        res.send(err)
      res.json(rooms);
    });
  });
});
```

After updating the object we do the same as our previous POST request and perform a get request on our Room collection in order to return the new refreshed array of room objects to our Ionic application.

Finally we have our search function. I will show an example below of searching for a property using the following criteria: College, Number of Rooms, Parking and Price.

```
//search function on college parking price and number of
 ↪  rooms properties
app.get('/api/searchProperties/:col/:numRooms/:parking/:price',

function(req, res) {
  let Col = req.params.col;
  let numRooms = req.params.numRooms;
  let Parking = req.params.parking;
  let Price = req.params.price
  console.log("fetching Search properties" + Col + numRooms);

  Property.find({"College": Col, "NoRooms":numRooms, "Parking":
   ↪  Parking, "Price": {$lt: Price }}, function(err,
   ↪  properties) {

    if (err)
      res.send(err)

    res.json(properties);
  });
});
```

As we can see above we perform another GET request but this time we take a list of parameters in our routing address in order to search our Properties collection. Using these and our mongoose find function we can search for property objects fitting the criteria passed through. For example : A property that is located beside GMIT, has 4 bedrooms, has no Parking and Price is less than 1200.

Here concludes our Logic Tier, next we will discuss our Presentation Tier.

5.3 Presentation Tier

In the Presentation Tier we will cover 4 sections. Firstly we will explain the Ionic 3 app structure, Secondly we will go through our Providers which we use to interact with our NodeJS server and through that our Mongo DB, we will then go through each of our Digs Apps Pages, displaying what it looks like and its functionality and finally how we styled our application.

5.3.1 Ionic 3 App Structure

Lets get started with the Ionic 3 Apps Structure. In the Figure 5.2 we can see the structure of our Ionic App.
We will briefly explain each of the main folders and files:

- Node-modules – Here contains the packages required to run and develop this ionic App
- Platform – Here contains the technologies needed for a ionic. Ie CSS, JavaScript, Angular
- Platforms – Here contains the platforms which our Application is being made for. Ie IOS or Android
- Plugins – Here contains the plugins needed to give the capability to interact with the phones native capabilities.

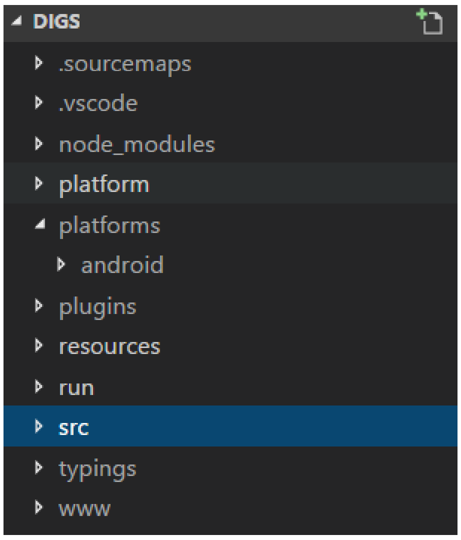


Figure 5.2: Ionic 3 Structure.

- Resources – contains the Apps resources such as images, splashscreens and readMe
- Run – enables us to run our application
- WWW – is where our index.html lies and is the root component where the app will load.
- Src – is the folder where we will do most of our coding for our Ionic 3 app. In here we will be using scss, typescript, angularjs and java script.

In the src folder Ionic 3 have structured our components into well thought out Folders as seen in figure 5.3 below.

- App – This is where we set up the base of our app. In here we provide access to components(such as other pages, providers, bootstrap etc) for various pages in our App as well as creating the App shell. For example whether our App will be a side Menu Application structure.
- Assets – This contains our images and icons.
- Models – In here we create our object models. For example our User Model which is made up of a String Email and a String Password.
- Pages – This is where the majority of our presentation tier code goes. Here we design each of our pages and its functionality. We will be going into each page in more detail below.

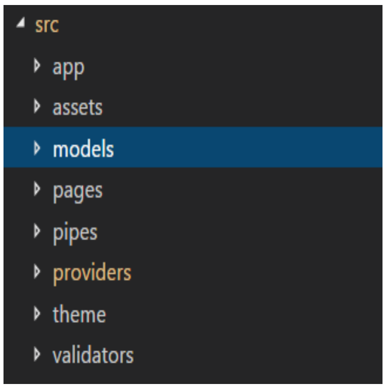


Figure 5.3: System Architecture.

- Pipes – In our pipes folder we link up with our list arrays to perform a search depending on price of a accommodation.
- Providers – This is where we link our Application to our Logic Tier Servers. We have 4 providers handling our messages, images, properties and rooms. In the next section we will show you how we accomplish this.
- Theme – Theme allows us to set the over all design and look of our app through global scss.
- Validators – In here we provide functions to validate whether users input was correct. Ie valid email address.

5.3.2 Providers

Providers gives our Application access to our data tier through our servers. In here we provide the logic and perform http requests to our NodeJS Servers to either return , create , update or delete data. Rather than repeating ourselves for each provider we will give examples of how our provider works using the RoomAd.ts Provider.

5.3.3 Pages

Ionic 3 Pages are broken up into 4 parts as seen in Figure 5.4:

Page.ts : TypeScript file which handles the logic of the page. Here we code our individual functions and bring in our imports needed for functionality. For example here is how we import FireBase for our Login Page.

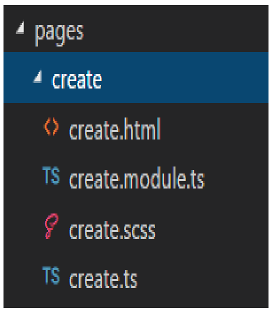


Figure 5.4: Page File Structure.

```
import { AngularFireAuth } from 'angularfire2/auth';
```

Page.scss : SCSS file which defines the styling of the page.

Page.ts : HTML file which creates the overall layout of the page. Adding forms, lists, buttons and images where needed. With the Ionic framework we can use angular directives on html elements such as buttons which offers us greater functionality and design. Here is an example of a angular directive on a html button:

```
<button ion-button large>Large</button>
```

Page.module.ts : This file is used for deep linking. In this file they create the underlying routing for our app.[?] This file is automatically created when a page is generated. We dont use this file for our coding of the app.

Below we will go through each of our apps pages features and functionality.

5.3.3.1 Login Page

Login Page is the first page the user sees when the Application is launched for the first time since install. It is here where they can Login to there account(if they are an existing user) with either there email and password or Login using there google account. The user also has the option to tap our menu button in the top left corner of our header and navigate through our features available for users who don't have an account. When entering the email and password, validation is performed to ensure that the email is the right format, the password contains the right characters and length and all field are filled.
The user is authenticated using Firebase Authentication which queries our Firebase API to check if we are an existing user or has a google account.

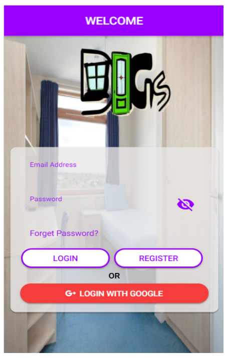


Figure 5.5: Login Page.

If the user is successful the API returns details on the user, a session Key to grant access to our Apps key features and navigate the user to the Home Page. If the user forgets there password they can tap on the Forget Password link provided and this will bring them to the forget password page. If the user is new and would like to register they can tap the Register button to navigate to the Register page.

5.3.3.2 Register Page

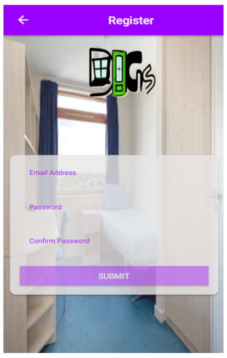


Figure 5.6: Register Page.

Here the user will be asked to enter their email and their password twice(to ensure correct password is entered) in order to register. As in the Login page the same validation is performed on the email and password fields. When the user taps Submit, the firebase API will store the user details, return a session key to grant access to our Apps Key features and navigate to the Home page.

5.3.3.3 Forgot Password Page

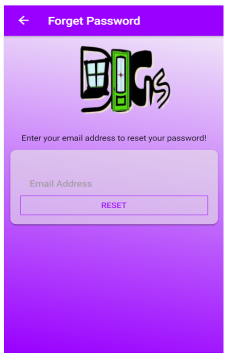


Figure 5.7: Forget Password Page.

Here the user will be asked to enter their email address and to tap the reset button. If the email is a registered email address the user will be asked to check their email account and follow the instruction to reset their password upon receiving an email from DigsApp.

5.3.3.4 Side Menu

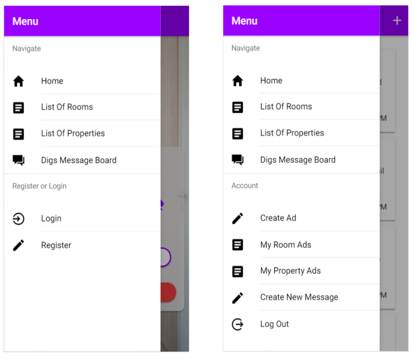


Figure 5.8: Menu Pages.

Our side menu allows navigation to our App's various pages. The menu is available for authorized users and unauthorized users. However, unauthorized users will have limited options available to them. For example, they will not be able to create room ads, create property ads, create messages for the message board and will not have access to the My Ads Pages.

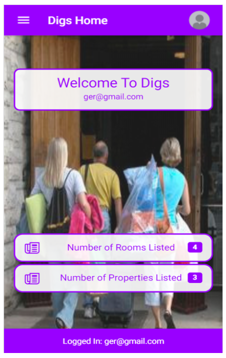
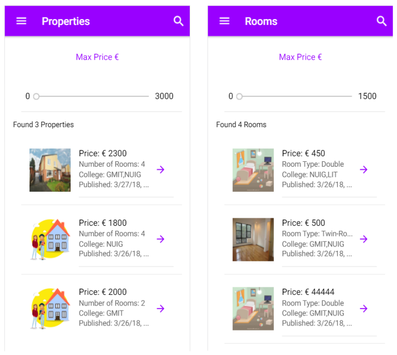


Figure 5.9: Home Page.

5.3.3.5 Home Page

In the header of our Home page contains our menu button in the top left corner our title and in the right corner if you are logged in with google it will show your profile picture. In the body of the page contains a welcome message to the users, two buttons to our property and room lists which display the number of ads in each and at the bottom to button to either Login or Register for our App.

5.3.3.6 Rooms List Page and Properties List Page



Found 3 Properties

Price: € 2300
Number of Rooms: 4
College: GMIT,NUIG
Published: 3/27/18, ...

Price: € 1800
Number of Rooms: 4
College: NUIG
Published: 3/26/18, ...

Price: € 2000
Number of Rooms: 2
College: GMIT
Published: 3/26/18, ...

Found 4 Rooms

Price: € 450
Room Type: Double
College: NUIG,LIT
Published: 3/26/18, ...

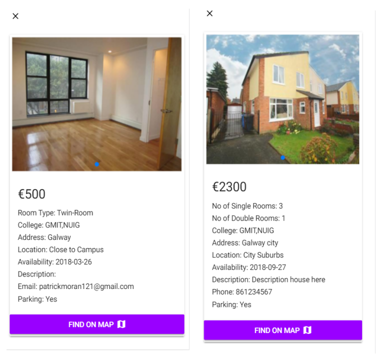
Price: € 500
Room Type: Twin-Ro...
College: GMIT,NUIG
Published: 3/26/18, ...

Price: € 44444
Room Type: Double
College: GMIT,NUIG
Published: 3/26/18, ...

Figure 5.10: Room and Property List Pages.

Both of these Pages are very similar so I will cover them both in this section. In the top right corner in the header of these pages, the user can tap the search icon and navigate to our search page. In the body of these pages the user is shown the list of ads available, showing the main criteria to the user to make an informed decision on which ad to tap on. The two pages differ on the criteria shown and information they hold. For example the property list item would indicate how many rooms the property has while the room item would display the room type. At the top of list the user has the option to search the list by setting the Max price they are willing to pay. Our Lists Pages imports pipes search.ts file to perform this action. To refresh the list of Ads at any time, pull downwards on the list of ads and the Page will refresh with the most up-to-date list.

5.3.3.7 Room and Property preview Modals


€500
Room Type: Twin-Room
College: GMIT,NUIG
Address: Galway
Location: Close to Campus
Availability: 2018-03-26
Description:
Email: patrickmoran121@gmail.com
Parking: Yes

€2300
No of Single Rooms: 3
No of Double Rooms: 1
College: GMIT,NUIG
Address: Galway city
Location: City Suburbs
Availability: 2018-09-27
Description: Description house here
Phone: 861234567
Parking: Yes

Figure 5.11: Room and Property preview Modals.

Firstly I will explain a Modal. A Modal is like a pop-up screen but offers the same functionality as a regular page. To access this page the user would tap on one of the ads from either properties or rooms. This would populate the modal and display the full contents of the ad. In here it will allow the user to side scroll through the ads images, read about the ad in more detail and, using the Find on Map button on the bottom of page, view the location of the accommodation on google maps.

5.3.3.8 Search Page

Here we have a adaptable search page depending on where the user has navigated from will determine on what criteria the user can search on. If the user taps on the search icon in the List of Rooms Page, the search page will show the search criteria to search for rooms. For example room type. If the user taps on the search icon in the List of Properties Page, the search page will show the search criteria to search for rooms. For example Number of Rooms. Each field is required to perform a search apart from price that is optional. When the user tap the Search Ads button the will navigate to the Search Result Pages.

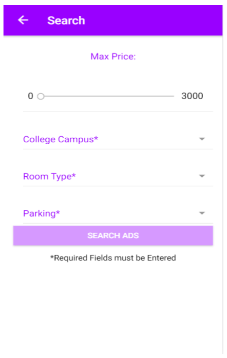
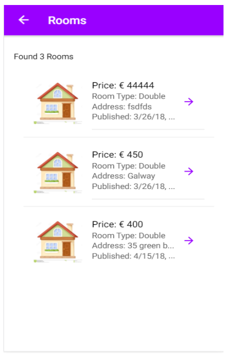


Figure 5.12: Search Room Page.

5.3.3.9 Search Result Property Page and Search Result Room Page



Found 3 Rooms

Price: € 44444
Room Type: Double
Address: fsdfds
Published: 3/26/18, ...

Price: € 450
Room Type: Double
Address: Galway
Published: 3/26/18, ...

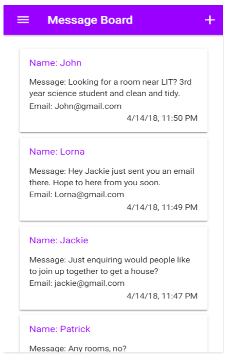
Price: € 400
Room Type: Double
Address: 35 green b...
Published: 4/15/18, ...

Figure 5.13: Search Result Room Page.

On these pages we display the results of our Searches. These pages are exactly the same as our List pages in Section 5.3.3.6 apart from they do not have a Search button or the ability to search by setting max price. If the user wishes to search on Price they can simply press the back button and they will be in the Search Page once again. The user can still tap any of the item in the list and our preview Modals will display the ad in more detail.

5.3.3.10 Digs Message Board Page

In the top right of the header of the Message Board Page you will see a button with a plus icon. If the user taps this button they will navigate to the Create Message Page. However this button is only available to the user if they are logged in. In the body of this page the user will be able to read and scroll down through the message board posts. To refresh the Message Board



Name: John

Message: Looking for a room near LIT? 3rd year science student and clean and tidy.
Email: John@gmail.com
4/14/18, 11:50 PM

Name: Lorna

Message: Hey Jackie just sent you an email there. Hope to here from you soon.
Email: Lorna@gmail.com
4/14/18, 11:49 PM

Name: Jackie

Message: Just enquiring would people like to join up together to get a house?
Email: jackie@gmail.com
4/14/18, 11:47 PM

Name: Patrick

Message: Any rooms, no?

Figure 5.14: Digs Message Board Page.

at any time, pull downwards on the list of Posts and the Page will refresh with the most up to date list. Above has shown all the available pages to the unauthorized user. The next pages will show pages only available for authorized users.

5.3.3.11 Create Ad Page

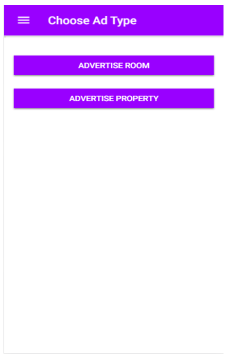


Figure 5.15: Create Ad Page.

As seen in Figure 5.8 displaying our Menu, the user can tap on the Create Ad, doing this will bring them to the page shown in Figure 5.15 where the user can choose whether to Advertise Room or to Advertise a Property by taping one of the buttons shown.

5.3.3.12 Create Room Ad Page

On this page the user is presented with a form (Figure 5.16). In this form the user will be required to fill each field. We also validate input on key areas such as phone numbers, email address etc. In the Eircode field we offer the user a link to the Eircode website, in case they don't know their Eircode.

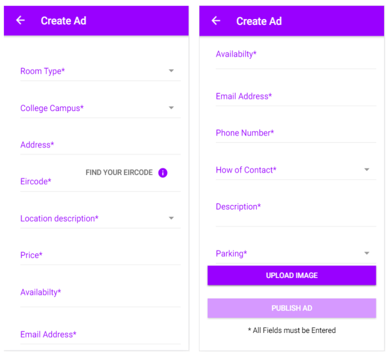


Figure 5.16: Create Room Ad Page.

At the bottom of the form there are 2 buttons. The first button called Upload Image when pressed will present an action sheet asking the user whether they want to upload an image from library, use camera or cancel this action. With the use of the cordova plugins we can gain access to the phones native functions and upload an image from the users image library or capture an image with the phones camera and upload it. After the user has uploaded an image the upload image button changes and displays Add another Image and allows the user to repeat the upload image step again. The button below is the Publish Ad button, when taped it will save your ad to the Rooms Collection in our database and redirect you to the updated Room List Page. The Publish Ad button is only tappable when all required fields have been entered.

5.3.3.13 Create Property Ad Page

As you can see from the Create Room Ad Page in section 5.3.3.12, the Create Property Ad Page is very similar. The only difference between the 2 pages is the input fields and that the Property Ad will be saved to the Properties Collection in the Database. In the Create Property Ad, we set the property type and ask the user for the number of each type of room in the property. We don't ask these fields in our Create Room Ad Page.

5.3.3.14 Create Message Page

Here the user can post a message on our Message board. As before, this form requires all fields to be filled with valid input. When the user taps the

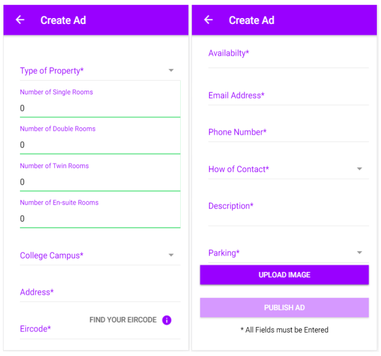


Figure 5.17: Create Property Ad Page.

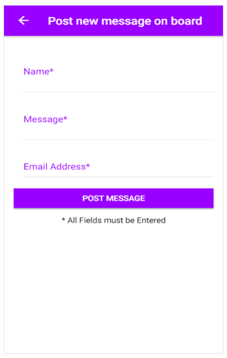
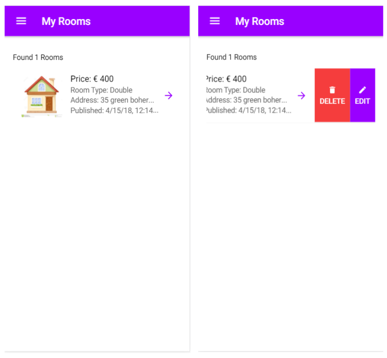


Figure 5.18: Create Message Page.

button Post Message, the Message will be saved to the Messages collection and you will navigate back to the new updated message board.

5.3.3.15 My Room Ads Page and My Property Ads Page

On these pages the user can view a list of ads they have created/posted on the App. Using the users unique email address which is used as a key for each ad published we can find all the ads created by one particular user. As seen in the previous Room List Page and Property List Page the user can tap on a item in the list and a Modal will present the Ad in full. The user can swipe left on each of their My Ads items and two buttons will become available, a delete button to delete this specific ad and a edit button to edit this specific ad. If the delete button is tapped the Ad will be deleted from our database and we will refresh the current page with the updated list. If the Edit Button is tapped the user will navigate to either the Update My Property Ad Page or Update My Room Ad Page depending on the Ad type being selected.



Found 1 Rooms

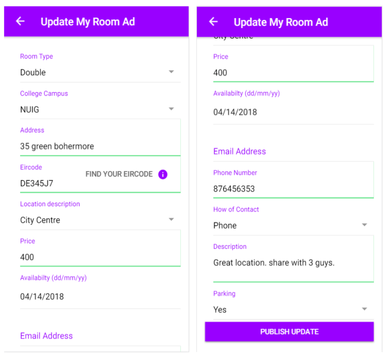
Price: € 400
Room Type: Double
Address: 35 green boher...
Published: 4/15/18, 12:14...

Found 1 Rooms

'rice: € 400
'oom Type: Double
'ddress: 35 green boher...
'ublished: 4/15/18, 12:14...

Figure 5.19: My Room Ads Page

5.3.3.16 Update My Property Ad and Update My Room Ad



Room Type: Double
College Campus: NUIG
Address: 35 green bohermore
Eircode: DE345J7
Location description: City Centre
Price: 400
Availabilty (dd/mm/yy): 04/14/2018
Email Address

Price: 400
Availabilty (dd/mm/yy): 04/14/2018
Email Address
Phone Number: 876456353
How of Contact: Phone
Description: Great location. share with 3 guys.
Parking: Yes

Figure 5.20: Update My Room Ad Page

Here the user will be presented the same form as before for the create property and create room pages in sections 5.3.3.12 and 5.3.3.13. The only difference is that the fields will already be populated with the information of the ad you wished to update. You can simply change the entries you wish to edit and re-publish the ad. This will update our collections in our database and redirect you back to the newly updated my ads list.

5.3.4 Styling

Our Ionic 3 application allows us to style our Digs app in many different ways. For the overall design scheme of the App we wanted to have clear cut lines and simplistic look. We didn't want to overwhelm the user with a to busy interface. So we kept the look and feel of the App simplistic.

For our colour scheme we went for white and purple. Having looked at many other apps trying to find inspiration. We found as a team that we were drawn to this colour combo. Having decided on our base colours, we were able to implement are colour scheme very easy with the use of the Theme folder in the Ionic 3 app. Here we have a variables.scss file which allows us to declare our most used colours under the headings of , primary, secondary, danger, light and dark. By doing this we now have access to this colour scheme throughout our app. Below is a code snippet of our variables.scss file:

```
$colors: (
  primary:     rgb(153, 0, 255),
  secondary:   rgb(17, 182, 30),
  danger:      #f53d3d,
  light:       rgb(255, 255, 255),
  dark:        #502274
);
```

To add more colours we could simply create another variable name and add it to colour variables.

Having set up our theme we then worked on our styling. We achieved this using in-line styling and our scss files for our pages. Ionic 3 comes with huge range of attributes for elements which helped us easily style buttons, ranges, forms and many other types of elements. For example here is some in-line styling performed on our register button in our home page:

```
<button class="login-but" ion-button color="light" round
 ↪  style="color: rgb(153, 0, 255); border: solid 2px"
 ↪  (click)="goToLogin()" float-start>Login</button>
```

As you can see Ionic allows you to set attributes such as color, round and float-start.

As explained in our Pages section 5.3.3 each page comes with their own scss file. In here we can style elements in our page by either their name or class. This allows us to keep our code tidy and better organised. Here is an example of our Login scss file:

```
.background{
    background-image: linear-gradient(rgba(255,255,255,.1),
     ↪  rgba(255,255,255,.1)), url('../assets/imgs/room.jpg');
    background-position: center;
    background-size: 100% 100%;
    width: 100%;
    height: 100%;
}
```

As you can see above we set the background to an image in our imgs folder and by using linear gradient and controlling the opacity we were able to control how much the image stood out in the background. We were also able to position and size the background to our specific requirements. We used a mix of these in-line and scss styling throughout our project.

We also used angular directives inside our html elements to control what was shown on our pages. This gave us great power and the ability to reuse pages rather than duplicating code. For example in our Search page depending on which type of list your searching we only display the fields for that particular search:

```
<ion-item [hidden] ="navFrom">
    <ion-label color="primary" floating>Number of
     ↪  Rooms*</ion-label>
    <ion-input [(ngModel)]="NoOfRooms" type="number" name =
     ↪  "NoOfRooms"></ion-input>
</ion-item>
```

We use the [hidden] directive to hide this element unless the Boolean variable navFrom is equal to true.

5.3.5 Deployment

To Deploy our project we hosted our Database and Servers on Amazon Web Service(AWS) instances.

To run our Mongo DB we would open a command prompt and cd to the location of our MongoDB bin folder and use the command Mongod as shown:

```
C:\Program Files\MongoDB\Server\3.4\bin>mongod
```

To run our digs server we would open command prompt and cd to the location of our DigsServer server.js file. In here we would run the command npm start. If this was the first time running this server we would type the command npm install before the command npm start.

```
C:\Users\gerar\Desktop\4th-Year-Final-Year-Project\DigsBackEnd\DigsServer>npm
 ↪  start
C:\Users\gerar\Desktop\4th-Year-Final-Year-Project\DigsBackEnd\DigsServer>npm
 ↪  install
```

To run our image server we repeat the same steps as for the Digs server.

```
C:\Users\gerar\Desktop\4th-Year-Final-Year-Project\DigsBackEnd\DigsImageServer>n
 ↪  install
C:\Users\gerar\Desktop\4th-Year-Final-Year-Project\DigsBackEnd\DigsImageServer>n
 ↪  start
```

This will have our data tier and logic tier up and running. Now the user can download our Digs App Apk to their phone and install it.

Chapter 6

System Evaluation

In this chapter we evaluate our system in the following areas:

- Robustness
- Testing
- Scalability
- Results Against Objectives
- Limitations

6.1 Robustness

Unfortunately due to time constraints, we have not been able to test our application fully for robustness. For example, load testing and stress testing . However, what we do know from testing our application, as laid out in the following section, is that we have not come across any errors, glitches or application failure so far.

6.2 Testing

Through continuous white box testing and regular black box testing we believe Digs has reached the expectations we set out at the start of our project. To achieve these desired results we tested our application using white box testing in the following ways.

1. Ionic Serve

2. Simulator Testing
3. Device Testing

6.2.1 Ionic Serve

Testing our application in a desktop browser was just a matter of running the following command in the project's root folder.

```
ionic serve --lab
```

As a result, a live-reload server of our project will load. When we make a change in our HTML, JavaScript or CSS, the change will automatically reload in the browser when the file bring edited is saved. Using the -–lab option on the ionic serve command, we can also view how each change made would appear on Android, iOS and Windows platform.

6.2.2 Simulator Testing

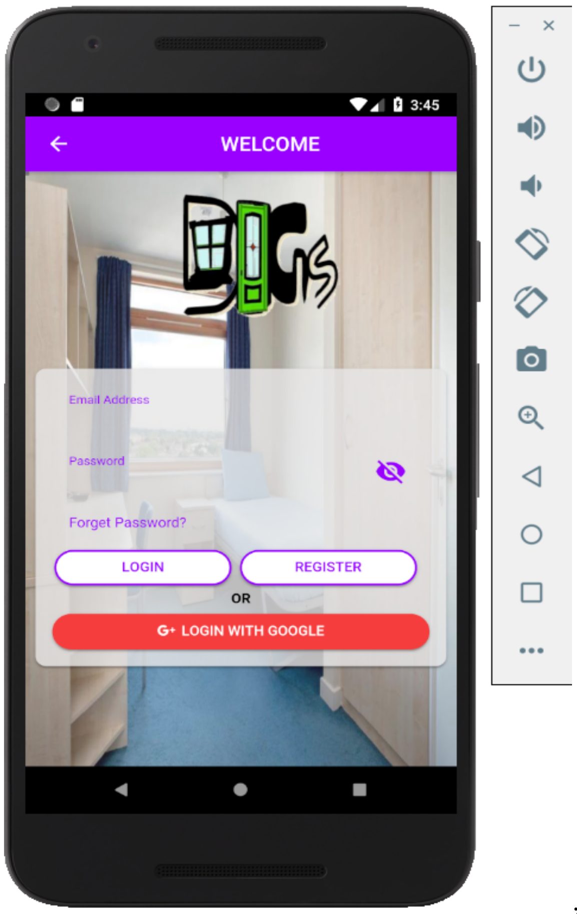
Another form of testing we used was simulator testing. This is where we tested Digs in an android phone emulator environment (Figure 6.1). The android emulator simulates an android phone and other devices in the android family. We chose to use android as our main testing platform as it is a widely popular platform and was the easiest to setup with no restrictions. For example, to test on an iOS emulator requires an Apple computer along with xCode installed and an Apple device in order to test it. The following commands were used to run our application with an android emulator.
The following commands was used to run our application with an android emulator.

```
ionic cordova run android
```

6.2.3 Device Testing

The last testing method we used was to test the application on an actual device. To test the application on an android device we first built an android apk file and then installed it on an android device. The command used to build an android apk file is:

```
ionic cordova build android
```



,

Figure 6.1: Android Simulator

6.3 Scalability

Our Digs application was designed to be very scalable. Currently, only store small quantities of data can be stored. This is because we chose to setup an AWS micro instance. AWS micro instances have 1 GB of RAM, 1 CPU, 30 GB of storage and are free-tier eligible. However, if the volume of data was to increase in the future, this can be easily scaled up to a larger AWS instance to improve performance and storage.

Another aspect of our project scalability is our NodeJS server in our logic tier. NodeJS is an ideal solution when considering scalability in your application. This is because Node can process many simultaneous connections due to its non-blocking, event driven I/O system.

6.4 Results Against Objectives

When we compare the objectives laid out in Chapter 2.1 to the application we have created, we believe we have met these requirements. Regarding helping students in the accommodation crisis we cannot confirm this until our application is launched for public use and a business model created.

We also wanted to create an app that was student-orientated and user-friendly. To do this we implemented college campus as both a field and a search option for our listings. We also created a message board for students to communicate with each other to find accommodation. These aspects differentiate our application from other accommodation apps available.

6.5 Limitations

The following are some limitations we experienced while developing our Digs app.

6.5.1 Native look and feel

Even though developing cross-platform applications with the Ionic framework has become more refined in the last few years, it is still quite difficult to create an application with a native look and feel to it, for example developing an Android application with Java and Android Studio.

6.5.2 Lack of Apple Products for Testing

As discussed in the previous section 6.2, we had some limitations testing our cross-platform application on the iOS platform. This was because no one in our team had access to an Apple mobile or computer. Implementing this in future would increase the potential reach of our app to many more clients.

6.5.3 Service Limitations

In our application we are using firebase and AWS as services for free. We are using these free services, which both have limitations explained in 6.3, as we are currently only developing the application as a prototype and because we are all students.

Chapter 7

Conclusion

We can honestly say as a team, that this project was a journey. Not in all our four years in Galway-Mayo Institute of Technology have we taken on a project of such scale and complexity. We have had ups and downs but over all our project has run pretty smoothly.

Our App objective was to develop an accommodation app specifically designed for students to find accommodation for themselves and for letters/-landlords and students to advertise accommodation specifically for students. We created an app which does just this.

What we have accomplished:

- Users can view our accommodation list
- Users can search our accommodation lists
- Users can view our Digs Message Board
- Users can create their own Account
- Users can Log into their Account
- Users can reset their password if they have forgotten theirs
- Logged in users can create their own accommodation ads for the app
- Logged in users can post their own messages on the Digs Message Board
- Logged in users can view ads they have created
- Logged in users can edit and delete ads they have created

We believe this app can help alleviate some of the stress and burden involved in finding student accommodation, particularly for those who are arriving to a new area. We hope this will save them time, stress and hopefully enable them to find suitable accommodation with new-found friends, as well as helping landlords/letters to easily advertise to this specific market.

As stated above in our Introduction chapter our overall objective was to create a project “that highlighted our existing skills and allowed us to learn new skills and develop as part of a team”. Through bringing technologies such as Mongo DB, NodeJS, Firebase and Ionic3 together in one application we have been able to showcase our existing skills developed over the last four years in GMIT and learn new ones which we can add to our portfolio.

As for learning how to work as a team not only through technology but also as people we found this project a great learning exercise. We all had our strengths and weaknesses which is common in all teams but through good communication and management of our time and resources we worked to produce a project that all three of us are proud of.

7.1 Future Development

We would like to see this app develop to its full potential by implementing a business development plan whilst also enhancing the technical capabilities of the app.

7.1.1 Business Development

As our Application is solely geared towards the student market we thought it would be a good idea to see if the Students Union would be interested in getting involved in our App. Currently we are still waiting on a response from our own Students Union President. We were hoping to create a student union page to our app where the students union could promote issues, services and competitions to the students of Ireland. Also the students union currently holds an accommodation list for properties around colleges and would be able to advertise these on our app.

7.1.2 App Development

- Save Ads – We would like the user to be able to save an ad as a favourite in order to view it again later.

- Further testing – As mention in chapter 5 System Evaluation, we would like to further test our app for robustness which means load testing and stress testing our application.
- Get app on the store – Like every App developer our end goal is to get our App on the store and into public use. We would aim to get our Digs app on the Android store first, as requirements are not as stringent and eventually make it available for the Apple Store.
- Create a Web application – So not to limit ourselves to just Android and Apple user, we would also like to create a web application with the same functions and features as our Digs app.

Chapter 8

Appendix

Project Source Code Link:
https://github.com/gerardnaughton7/4th-Year-Final-Year-Project

Bibliography

[1] B. Fallon, "Daft accommodation." Available: `www.Daft.ie`, [Accessed: 28- October- 2017].

[2] J. Morony, "Building a review app with ionic 2, mongodb & node." Available: `https://www.joshmorony.com/building-a-review-app-with-ionic-2-mongodb-node/`, [Accessed: 12-October-2017].

[3] Riron, "ionic-img-viewer." Available: `https://github.com/Riron/ionic-img-viewer`, [Accessed: 28- October- 2017].

[4] Devdactic, "Image upload back end." Available: `https://devdactic.com/ionic-image-upload-nodejs-server/`, [Accessed: 20- January- 2018].

[5] "Postman," 2018. Available: `https://www.getpostman.com/`, [Accessed: 15- April- 2018].

[6] Devdactic, "Image image upload." Available: `https://devdactic.com/ionic-image-upload-app/`, [Accessed: 08- February- 2018].

[7] Cordova, "Launch navigator plugin." Available: `https://github.com/dpa99c/phonegap-launch-navigator`, [Accessed: 12 - March - 2018].

[8] E. Horowitz, "Mongodb," 2018. Available: `https://www.mongodb.com/blog/post/multi-document-transactions-in-mongodb`, [Accessed:9- April- 2018].

[9] "Mongodb," 2018. Available: `https://docs.mongodb.com/manual/reference/bson-types/`, [Accessed: 8- April- 2018].

[10] "Firebase documentation," 2018. Available: `https://firebase.google.com/docs/auth/`, [Accessed: 9- April- 2018].

[11] V. Gudivada, D. Rao, and V. Raghavan, “Nosql systems for big data management,” 2014. Conference: 2014 IEEE World Congress on Services, At Anchorage, Alaska, [Accessed: 20- March- 2018].

[12] L. Okman, Y. Gonen, E. Gudes, , and J. Abramov, “Security issues in nosql databases trust, security and privacy in computing and communications (trustcom).” 10th International Conference, 541-547, 16-18 [Accessed: 22- March- 2018].

[13] “About node.js®,” 2018. Available: `https://nodejs.org/en/about/`, [Accessed: 10- April- 2018].

[14] R. Camden, “My perspective of working with the ionic framework,” 2014. Available: `https://www.raymondcamden.com/2014/07/28/my-perspective-of-working-with-the-ionic-framework/`, [Accessed: 27- March- 2018].

[15] “ionicframework,” 2018. Available: `https://ionicframework.com/docs/components/#overview`, [Accessed: 27- March- 2018].

[16] S. Reimler, “How to get started with ionic,” 2018. Available: `https://ionicacademy.com/get-started-with-ionic/`, [Accessed: 27- March- 2018].

[17] “Cordova introduction,” 2018. Available: `https://cordova.apache.org/docs/en/latest/guide/overview/index.html`, [Accessed: 10- April- 2018].

[18] W. Chegham, “Angular: One framework,” 2017. Available: `https://jaxenter.com/angular-one-framework-134755.html`, [Accessed: 11- April- 2018].

[19] “Typescript tutorial,” 2018. Available: `https://www.tutorialspoint.com/typescript/typescript_overview.htm`, [Accessed: 11- April- 2018].

[20] M. Birnbhaum, “Psychological experiments on the internet,” 2000. Available: page: 64 `https://books.google.ie/books?id=Hrp8peu1FCAC&pg`, [Accessed: 27- March- 2018].

[21] A. LaPAUGH, “Simple javascript programs,” 2010. Available: `http://www.cs.princeton.edu/courses/archive/fall10/cos109/JS_topost/`, [Accessed: 28- March- 2018].

[22] "Html," 2017. Available: `https://www.computerhope.com/jargon/h/html.htm`, [Accessed: 11- April- 2018].

[23] O. Briggs, S. Champeon, E. Costello, and M. Patterson, "Cascading style sheets: Separating content from presentation," 2004. Available: `http://www.yourhtmlsource.com/books/extracts/cssscfpfoundation.html`, [Accessed: 28- March- 2018].

[24] "Latex-project," 2018. Available: `https://www.latex-project.org/about/`, [Accessed: 14- April- 2018].

[25] "About sharelatex," 2018. Available: `https://www.sharelatex.com/learn/Learn:About`, [Accessed: 14- April- 2018].

[26] S. Basu, "Features of visual studio code," 2015. Available: `https://developer.telerik.com/featured/10-awesome-features-of-visual-studio-code/`, [Accessed: 15- April- 2018].

[27] "Visual studio code docs," 2018. Available: `https://code.visualstudio.com/docs`, [Accessed: 15- April- 2018].

[28] "Visual studio code tricks," 2018. Available: `https://code.visualstudio.com/docs/getstarted/tips-and-tricks`, [Accessed: 15- April- 2018].

[29] K. Brown, "What is github, and what is it used for?," 2016. Available: `https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/`, [Accessed: Accessed: 01- April- 2018].

[30] C. Software, "Mongoose." Available: `http://mongoosejs.com/`, [Accessed: 02- November- 2017].