In-Touch, A RESTful Based Social Media App

Programme: Bachelor of Science (Hons) in Software Development

April 26th, 2019

Programme Code:GA787

Authors: Garry Cummins, Hugh Brady

Advised by: John French

adfa, p. 1, 2011.

Contents

Table of Illustrations

Abstract

For this project we decided on creating a social media server that can store a user information on a database in a cloud-based application. In order to achieve this, we need to establish a three-tier architecture to make a connection between the front end and back end of our project.

Firstly, we had to discuss and ensure with our supervisor that we were developing the project to meet the standards of a level 8 project. Due it being three tier architecture, it met the requirements that were needed to fulfil with the project. The front end was developed using Angular with the Ionic Framework. Users

will be able to view the club and user's collection in the remote mongo database using a Spring based API.

1 Introduction

1.1 Project Discussion

Nowadays there are a huge plethora of hobbies and interests that people have. Around the world there are entire communities that get together to practice their passion, whether that is sports, playing video games or watching movies. However, one thing we noticed was that a lot of these communities can be very insular, which is often not the fault of the community members but rather the natural result of a lack of an online presence. Many of these communities operate using private Facebook or Discord groups that can be hard to find and often don't share users. We decided that we wanted to create an application specifically tailored for finding clubs and events for any activity they have an interest in.

Once we had arrived at the objective for our project, we had to decide what technologies we would use. We wanted to use an array of technologies that worked in tandem together in order to show our capabilities as well as gain new skills and develop as part of a team. After some deliberation, we settled on using Ionic as our presentation layer, Spring as our data access layer and MongoDB for our database.

Originally our group consisted of 3 people, but unfortunately our third member had to defer the year. With the two of us left, we divided the work between us with Hugh Brady working on the presentation layer and Garry Cummins working on the data access layer.

1.2 Technologies Used

Below is a list of all the technologies used in the making of this application and a small description of what they were used in. They will later be expanded upon in the Technology Review section of this document.

- Ionic 4 – used to develop the web application.
- Spring - used to develop the data access layer.
- mongoDB – the database we used for the project.
- Heroku – a platform as a service used to host the database and data access layer.
- Mlab – Database-as-a-Service used to host the database on Heroku.
- Visual Studio Code – IDE used to develop the Ionic application.
- IntelliJ – IDE used to develop the Spring Boot application.
- Java – Language used to write Spring Boot application
- HTML – Language used to create web pages for the Ionic application.
- TypeScript – Language used to code logic for Ionic application.
- Node.js – JavaScript runtime environment used to create dynamic web pages.
- RxJS – Used by mobile app to read data incoming from the data access layer.
- Postman – used to test Spring Boot application's functionality.

1.3 Objectives

The primary objective of this project was to provide a single point where people could find clubs and events for any interest or hobby that may have a following.
As this project is divided into an applied project and a minor dissertation, our objectives for both will be laid out below.

1.3.1 Minor Dissertation

- Introduce the concept of the project, providing the reader with what we have set out to do with regards to the project's planned functionality and our motivations for this project.

- Provide a detailed explanation of each of the technologies that were used in the development for all layers of the project.
- Describe thoroughly the applied part of this project, with relation to our approach to the project, our evaluation of the system, issues encountered in the project's development cycle, how some issues were resolved and our final thoughts on the project.

1.**3.2 Software**

We needed to create a RESTful web application that interacted with data on a remote database. The user would be able to create an account, log into the application, search an area for groups or events, and join those groups or events.

- A user login and registration system.
- Users can create clubs.
- Users can search for clubs by area or hobby.
- Owners of clubs can create events.
- Event interactions such as
    - Mark as interested or attending.
    - Inviting other users.
    - Deleting event.
- Host the data access layer on a server.
- Work with the given learning outcomes for the project in mind, with regards to research, methodologies
- Gain experience working as a team in a professional manner.

1.4 Metrics for Success or Failure

Below is a list of metrics we will use for determining the success or failure of this project, with regards to the software, dissertation and our work as a team.

- Dissertation – A comprehensive explanation of our project with regards to our intentions, objectives, technologies used and critical evaluation of our work. The dissertation should be easy to comprehend even for people who are unfamiliar with the field.
- A simple, efficient mobile application whose functionality standards meet those of our objectives set above.
- A data access layer that can perform RESTful interactions with a hosted database.
- Our success as a team will be reflected by how we conducted ourselves under stress and resolved issues that arose throughout year.

1.5 Chapter Information

This paper is divided into different chapters that cover different topics ranging from planning, design, development and evaluation. This section will briefly outline what each chapter in this dissertation deals with.

2 Methodology

In this section, we will discuss what development method we took in order to develop this project, with regards to our research methods and our meetings with our project supervisor.

3 Technology Review

This section provides detailed descriptions of each of the different technologies we used in the development of this project.

4 System Design

Here we will cover the system architecture and design of our project, detailing the different components present, how they were implemented and how they communicate with one another.

5 System Evaluation

In this chapter we will evaluate our system in the areas of robustness, testing, scalability, and the results of the project against our objectives and limitations.

6 Conclusions

In the conclusion we will summarise the development of this project, discussing what went wrong and what we would do differently if we were to do this project again, as well as how we would continue development.

2 Methodology

This section will discuss the stages of development during the project life cycle that was undertaken during the development of the In-Touch Application. This will include the methodologies and processes we carried out which will be discussed and explained in depth. This section will also include how we used necessary tools to plan out are development process as well as problems we faced that were solved.

2.1 Agile – Planning and Development

When we first began developing the project, we took an agile approach towards research and development as it would allow us to change a component if it was needed. Agile is an iterative approach to project management and software development to help deliver results and avoid as many problems as possible. It also helped with the product requirements to make sure that the project was meeting the standards we set out for. We began by creating sprints which are short development cycles which ensured that we didn't stick to one process for too long. This helped with breaking down a feature that we found to be of use with the project or how we could get information from a database and how that could be implemented. We set the planning stage of the project to 8-10 weeks as we knew it would be critical to the projects flow. Planning included research of the three key aspects of our project:

1. Front End – Being able to create an application for the user to interact with (The UI of the project
2. API – Setting up an API to handle REST based requests (Create, Read, Update, Delete) between the front end and the backend.
3. Backend – Having a cloud-based application that stores a type of database that is open to manipulation

Researching these helped with discovering the requirements that we had to achieve going forward in the project life cycle. For each review of the technology we had to make sure it contained necessary tools we needed to achieve our goals.

Unfortunately, we lost one of our teammates during the research cycle of are project which caused deadlines we set out to finish to be pushed back until after Christmas. Coding than began early January.

2.2 Planning

In the planning phase, we held group meetings to discuss are objectives and requirements for the project. Features would also be discussed and what software's that we would need to about implement them for our three-tier architecture discussed above. We decided on having meeting once every week to discuss our findings and how it

could work with the project. This helped ensure that the project met with the level 8 standards. Meetings with our supervisor John French were also scheduled every Thursday at 4:00pm. These meetings helped keep us in check with what we planned as well as receiving feedback on our project to help with future progress. We created a Facebook message group to keep communication with each other outside of college hours that was strictly based on conversations on the final year project. This helped in the case of one of the team members not being able to attend are weekly meetings to be able to pass on their information as well as keep everyone informed when we would decide to meet up.

We made a lot of time during the meetings to have brainstorming sessions to help with visualising the designs, architecture, and sharing ideas with each other. Each member had their own time to discuss their thoughts and points with the project which helped with identifying features we needed to tackle and what obstacles that we might face. Technologies were reviewed also and analysed to determine whether it was suitable for what we wanted and that it was compatible with the software we wanted to implement it with. We identified tasks that each of us would need to do and was discussed and agreed with the team who would do what task (each teammate was given the freedom to choose task they wished to do) We used the Zoho Projects application to help setup tasks that each individual was assigned to complete within a certain deadline.

.

2.3 Development

During the development phase, we had to take a waterfall approach with are coding due to losing a member. We also had to decide how to split the work between two members rather than three. We first approached this phase by seeing how are architecture design could be implemented. We would discuss what tasks would need to be completed and what should take priority whether it be the database, api, etc… We also tried to set objectives to achieve at certain dates but some of the work conflicted with each other, so some tasks would not be completed until later as we didn't identify all the problems, we would face with the tasks we wanted to achieve.

2.4 Testing and Validation

When discussing with are supervisor the best way we could test our CRUD operations with the API, we decided on using postman. Postman helped us with creating simulate connections to the database which enable us to try requests so as GET, POST, PUT, and DELETE requests. We took the approach of familiar technology we used before but adding a new element with having them connect with each other which we haven't done before. We also decided on trying a new database that we haven't much

experience in before being MongoDB. We wanted a way of accessing a database for full manipulation and as MongoDB uses NoSQL, thought it would be the best database to use for what we planned. Ionic was chosen from experience for the front end but with ionic 4's release in January, took us a while to get used to its new features but was the only application we had some experience in that we could use as a front-end application. Spring was chosen for the API so that we could get a better understanding of the framework. We only briefly used it throughout the course and found its functionality very interesting and using spring tools to help with developing the API was worth researching.

2.5 Version Control (Github)

Throughout the development of the project, Github was used to help store are source code. This helped us split the project into four sections that we envisioned when planning the project being the front-end application, API, Cloud Database, and Documentation. Both members would work on individual parts of the project by using branches to keep work separated and merge them again for testing purposes. We would commit any changes we made throughout each week. The progress of the project during each commit and push to the repository is saved in local storage of the repo's directory. Team members ensured that the correct version control was managed by using the git pull commands to have the latest version before making any changes to the master branch. Github was a great help to the project life cycle as it gave us a way of checking individual changes and seeing issues that may have occurred to solve with the team. It also allowed us to revert back to any older version or commit of the project in case of the source code being completely broken. This can be done as long as one of the team members has access to the commit code and the name of the branch to perform rollback too.

3 Technology Review

3.1 Database Model – MongoDB

3.1.1 What is MongoDB?

MongoDB is a NoSQL database platform that is used for cross platforms. It uses JSON documents with schemata. It was developed by MongoDB Inc.

Mongo allows teams to organize their data easily and in real-time. It uses a flexible document data model which allows users that are building apps to easily access its data due to how intuitive working with the data is. Data can also be placed globally to help keep performance at an optimal level in any location and compliance with regulations. It can be run locally or on the cloud due to its deployment flexibility and data layers that can be accessed publicly.

3.1.2 Brief History

A software company called 10gen started development of MongoDB in 2007 as a planned (platform as a service) product. In 2009, they decided to change it to a (open source development model) and 2013 10gen changed their name to MongoDB Inc.

3.1.3 M-Lab

The project uses M-Lab to store the mongo created database collections stored in Heroku cloud. MLab is used as a fully managed cloud database service that hosts MongoDB databases. Most cloud providers so as the above mentioned Heroku (PaaS – Platform-as-a-Service), as well as Google and Amazon support and partnered with them.

3.1.4 M-Lab with Heroku

It becomes easily scalable as it does not require software install due to it being a add on and provides a connection string. It provides 100% SSD-backend disks and dedicated MongoDB processes. Admins are established if any issues arise with the database and to keep it up-to-date and functioning. They use a replica set cluster plan which offers high availability across multiple availability zones.

3.1.5 Mongos Main Features

- **Indexing –** The fields located in a MongoDB document can be indeed with primary and secondary indices
- **Ad hoc queries –** Mongo uses multiple search queries such as range queries, and regular
- **Load Balancing –** MongoDB scales horizontally using sharding which is referred to as an individual partition. The user can choose a shard key which can determine how the data in a collection will be distributed. The data (depending on the shard key) can be split into ranges across multiple shards. The shard key can also be hashed to a map which allows for an even data distribution.
- **Replication –** Replica sets are provided with mongoDB. A set usually contains of two or more data copies. Each member of the set replica can act in the role of primary or secondary replica at any possibility. The writes and reads are done on the primary replica by default. All secondary replicas maintain a copy of the primaries data using built-in replication. If the possibility of the primary replica fails, the replica set will enter a process called (Election process) which helps with determining which of the secondary replicas should become the new primary. The secondary replicas can optionally serve read operations, but ultimately that data will eventually be consistent by default.
- **File Storage –** MongoDB enables the functionality of being used as a file system (GridFS). It allows for load balancing and data replication features which can be done over multiple machines for storing files. The function for this operation is called grid file system which takes advantage of many

smaller file storage areas. It is included as a MongoDB driver. MongoDB helps with exposing the functions for file manipulation and its content.

- **Aggregation -** Used for clusters, MongoDB provides three ways for performing aggregation:
  (1) Aggregation Pipeline
  (2) Map-Reduce Function
  (3) Single-Purpose Aggregation

  The aggregation pipeline provides good performance for most operations while Map-Reduce can be used for batch processing of data and operations.

- **Transactions –** ACID (Atomicity, Consistency, Isolation, Durability) transactions are supported by MongoDB in June 2018 with the release of the 4.0 version.
- **Capped Collections –** MongoDB supports fixed-size collections called capped collections. The collection maintains an insertion order and once the specified size has been reached, will behave like a circular queue (Data Structure).
- **Server-side JavaScript Execution –** In queries, JavaScript can be executed for aggregation functions (Single Purpose, MapReduce) to be sent directly to the database to be executed.

3.1.6 Mongo Architecture

Serverless Access

MongoDB added MongoDB Stitch to help provide serverless access to MongoDB and other services. The client libraries can be accessed by JavaScript, Android and iOS devices.

Management and Graphical Frontends

This is used as the primary interface of the mongo database which is used in mongo shell. Version 3.2 introduced Mongo Compass which is used as the native GUI.

Programming Language Accessibility

Programming languages can be used in development environments due to MongoDB having official drivers for them. Unofficial drivers are also used by users for different frameworks and languages.

3.1.7 Issues and Problems with MongoDB

Security Issues
MongoDB allows anyone access with the full database due to its default security configuration. This caused a lot of mongo installations to be stolen, but fortunately new updates have made all networked connections to a database to be denied unless its explicitly configured by an administrator.

3.1.8 Technical Hiccups
Some scenarios may have an application access two distinct MongoDB processes, but these processes will not be able to access each other. Due to this, MongoDB may return reads that are stale (Returns incorrect values).

3.2 Application Model

In this section we will discuss the technologies that were used in the development of the front-end application.

3.2.1 Ionic 4 Framework
Ionic is a complete open source SDK for developing cross platform apps for native iOS, Android and the web from a single codebase. While earlier versions of Ionic were built on AngularJS, Ionic 4 was the result of experimenting turning Ionic into a set of Web Components. Web Components are a set of web platform APIs that allow custom components and widgets to work across modern browsers as well as used with any JavaScript library or framework that works with HTML.

Ionic uses Cordova to build and deploy its apps as native applications. Its applications are created primarily through the command line interface.

3.2.1 Visual Studio Code
Visual Studio Code (VS Code) is an open source code editor with powerful developer tools developed by Microsoft VS Code was developed as an alternative to Visual Studio IDE, which comes with more built in features but suffers from a large installation size and can be CPU intensive.
In contrast to this, VS Code is lightweight, with the option to simply download any extensions or tools needed. Built in support for both JavaScript and TypeScript and has support available for Java, C#, Python and many more.

3.3.3 TypeScript

TypeScript is an open source programming language developed by Microsoft in 2012. TypeScript is backwards compatible and is a superset of JavaScript. The code of the compiler that translates TypeScript into JavaScript is distributed under the Apache licence.

https://github.com/Microsoft/TypeScript

TypeScript differs from JavaScript in that it is an object-oriented programming language whereas JavaScript is a scripting language. TypeScript has the ability to explicitly enforce static types, supports, the use of classes and support for connecting modules. This results in improved development speed, readability, refactoring and reuse of code. TypeScript also accelerates the execution of programs. It is the standard language used in the Ionic framework.

3.3.4 HTML

HTML (HyperText Markup Language) is used for website creation Using HTML, you can insert paragraphs, buttons, checkboxes, form elements and much more. Hypertext is the method by which you move around the web; by clicking on special text called hyperlinks which bring you around the page. Mark-up is what HTML tags do to the text inside them e.g. control font size or make text italicised.

3.3.5 Cordova

Apache Cordova is a platform for developing open source mobile applications that is used by Ionic. It allows you to use the standard web technologies HTML, CSS and JavaScript for cross platform development, avoiding the native development language for each of the mobile platforms. Cordova's plugins can access a device's capabilities such as sensors and network. Cordova applications execute within wrappers targeted to each platform and rely on standard APIs to access device sensors, data and network status.

By default, Cordova provides basic browser capabilities that are available on a mobile device, but it also 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 on browsers in different plugins. For example, if the CSS or JavaScript functions differently on different platforms, Cordova tries to provide unified functionality for all supported versions of mobile operating systems.

3.3.6 Angular

Angular is a platform used to build dynamic mobile and web applications. Angular works in tandem with other projects, most notably Node.js, TypeScript and RxJS. In fact, Angular uses TypeScript as its standard programming language.

Previous versions of Ionic were built on the Angular framework, however Ionic 4 allows you to choose your user interface framework from Angular, React and Vue.js. While Angular is commonly related to single page applications, you can use Angular to build any kind of app, taking advantage of features such as two-way binding, RESTful API handling, dependency injection and more.

3.3.7 Node.js

Node.js is a JavaScript runtime environment for executing server-side JavaScript code. It allows developers to run server-side scripts to produce dynamic web content before the page is received by the client. What separates Node.js from other server-side technologies is that it is asynchronous. While PHP or ASP would handle a file request by sending the task to the computers file system, waiting for the system to open and read the file and return the content to the client and is then ready for the next request, Node.js would send the task to the computer's file system and is immediately ready for the next request, only returning the content to the client when it's ready. This means that Node.js can process more queries, faster. Asynchronous calls are especially important for webservers and are now seen as the standard for modern web applications. They allow the server to cope with thousands of parallel requests with little overhead as opposed to creating separate threads for each connection.

3.3.8 Reactive Extensions for JavaScript

Reactive Extensions for JavaScript (RxJS) is a library for reactive programming using observables that makes it easier to compose asynchronous or call-back-based code. RxJS exports a type of class called Observables. These classes also return an Observable. Unlike simple variables, observable act almost as functions, in that functions do nothing until called, and Observables do nothing until they are subscribed to. Observables are abstractions that help you deal with asynchronous operations. They can be represented as a stream of events that can be handled with the same API.

https://angular.io/guide/rx-library

3.3 Cloud Model – Heroku

3.3.1 What is Heroku?

Heroku is a cloud platform as a service which supports several programming languages. It was one of the first ever cloud platforms developed as early as June 2007. In only

supported the Ruby programming language when it was first introduced but now supports multiple languages such as Java, Node.js, Python, etc…

3.3.3 Heroku vs Google Cloud

Heroku and Google Clouds app engine are very close competitors. They both offers good and bad points. Google missed opportunities for having their AppEngine be a closed-in and be proprietary in nature which caused a lack of platform choices. Heroku offers very few restrictions on what can and can't be done on the hosted app, while also providing a huge amount of space for the user's application. Google continues to view its AppEngine as a way to tie in its customers and because of this Heroku is the better platform overall.

3.3.4 Review Heroku

Due to its ease of accessibility, Heroku was very beneficial for setting up the Mongo database remotely with only a few changes needed to be made for it to be accessed by the API. M-Lab also needed admin privileges so it can only be accessed by use which was done through the application.properties file in spring.

3.4 API Model - Spring Boot

3.4.1 What is Spring Boot?

Spring Boot is an open source Java-based framework which is used to create a micro service. It was developed by Pivotal Team and is used to build stand-alone and production ready applications. It has embedded Tomcat, Jetty and Undertow directly so it makes setting up RESTful web services much easier. Spring Boot based REST services are the exact same as Spring based REST services, with the only difference being how bootstrap underlines the application. JSON REST service enables the application to render JSON responses on the classpath. This allows applications like ionic to access the Spring REST services.

3.4.2 Advantages of Spring Boot

- Different to Spring, Spring Boot comes with no need to create a boilerplate configuration
- There are a lot of Spring Boot starters on the web such as Spring Intializr, to allow the creation of applications much easier and to be able to get up and running quickly with the code.
- Embedded Tomcat, Jetty, and Undertow support as mentioned
- It enables easy customization in application properties such as setting ports or Uri's for data flow.

3.4.3 Spring Initializr

Spring Intializr is a web application that can generate a Spring Boot project structure for the user (Bootstrap). It allows for the creation of project dependencies also for the pom.xml (Project Object Model) so it's easy to setup rest service dependencies.

3.5 Postman

3.5.1 What is Postman?

Postman is an API environment that can be used for testing requests through the API life cycle. All the tools that Postman possesses is based on the Postman collection format which includes: Mocks, tests, documentation, monitors, and publishing. Postman Workspaces enhance team collaboration. Changes made to a team workspace sync in real-time, so every team member is always working off of the most up-to-date version. Team admins can set permissions and manage contributors across multiple workspaces.

3.5.2 Positives and Negatives of Postman

Positives

- When using postman, it is very easy to create test suits. These suits can contain multiple integration tests.
- Test cases can be dependent of each other as some test cases may sometimes require data from others so postman can store data for them.
- Different environments may be needed to run different test cases. Postman keeps information for them to allow them to be executed.
- Postman can integrate different tools if needed such as Jenkins (Self-contained java-based program)

- Test cases can be moved from easily from one system to another.

Negatives

- Postman has some problems when it monitors test cases.
- It doesn't provide codeless web service testing

Overall postman is an excellent testing tool for API requesting testing.

3.6 IntelliJ IDEA

3.6.1 What is IntelliJ IDEA?

IntelliJ is a Java integrated development environment for developing computer software. Developed by JetBrains, it is a super slick and smooth environment which gives the user relevant suggestions in every context i.e., instant and clever-code completion, on the fly code analysis, and reliable refactoring tools.

3.6.2 Built-in tools and Frameworks

IntelliJ has a lot of already pre built-in tools that allows the user to get up and running as soon as possible. Tools such as version control, decompiler, databases, etc… It also comes with framework support such as Spring, Java EE, GWT/Vaadin, and so on to allow the creation on multiple different software's such as API's and Apps.

3.7 GitHub

3.7.1 What is GitHub?

GitHub is a web-based hosting service to create repositories to store code for version control using git. It provides all the distributed version control and source code management functionality of Git as well as adding its own unique features. It uses access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project created.

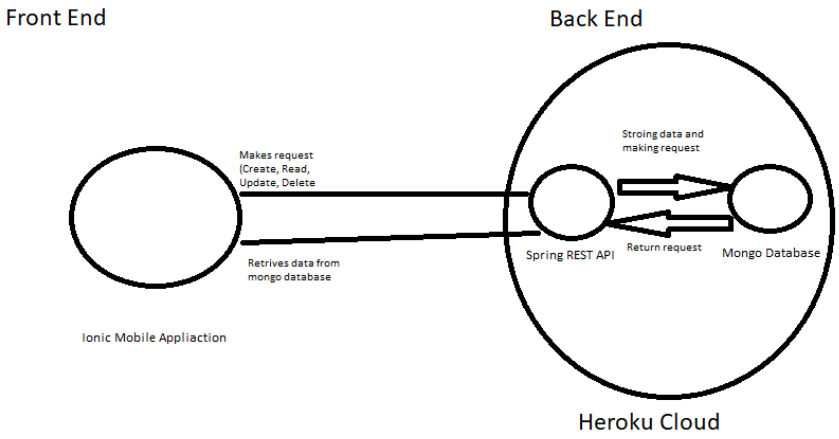
3.7.2 How GitHub Works

When accessing and signing up with GitHub for the first time, the user will notice the option to create a repository. These repositories can contain folders, files, images, videos, and datasets – anything that is required for the project being developed. During this process, GitHub will recommend creating a README file and s license. A README usually contains information on a project, such as technology and language used as well as how the project works and how to run it. The license file can determine the language that is being used for the project(These can be easily added later during the development cycle) When the repository is created, it can be pulled or cloned down to the user machine using the command line GitHub commands(git pull, git clone) in the selected directory. From there the user can add their code in that directory and move that code up to the repository (git add., git commit -m “commit message”, git push)

4 System Design

Our project consists of three parts or layers; the presentation layer, the data access layer, and the cloud-based database. The design of these parts can be viewed in screenshots below and the database structure is laid out also. A UML diagram is also provided to visualize the design or the system. In this chapter we will go through all the tiers and explain the overall design and architecture of our project.

4.1 System Architecture



Above is an image of the project architecture. Heroku cloud contains two applications that contain the REST API through Spring and the Mongo Database using mLab. These are both connected to each other to allow the DAL (data access layer) in the API to create the collections and store them in mLab to be then accessed by the ionic application that is connected to the API. The architecture is simple in design, but the spring API needs to ensure that the data is created and is able to be accessed remotely.

4.2 The Presentation Tier

The presentation tier describes the mobile application; the layer that the user will interact with. This part of the project was developed using Ionic 4. In this section we will explain the structure of our application, as well as detail the services that was used to send REST requests to our Spring Boot application.

4.3.1 Ionic 4 App Structure

We started our Ionic 4 application by using Ionic's CLI to generate a blank starter app. This installs dependencies and sets up the project. Here we will briefly go through each of the folders in the root directory of our Ionic 4 project and explain their purpose.

- node_modules – this folder contains the packages required to run the app
- plugins – this folder contains the plugins needed for interacting with a mobile phone's native capabilities.
- resources – contains the app's resources such as images and the README file.
- src – this directory contains our code, where most of the work for the Ionic app will take place. The TypeScript code in this folder is transpiled into the correct JavaScript version that the browser understands.
- www – contains the compiled code, which is erased and rebuilt every time the project is compiled. The code in this folder is the TypeScript from the src folder that has been transpiled into JavaScript.
- package.json – lists the dependencies for the application.

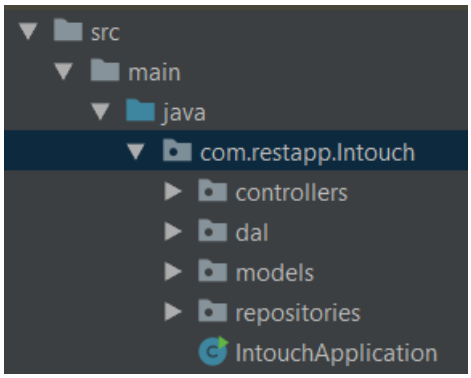
The src folder is where most of our work is done; especially in the app directory. The app folder is where we write the code for our app and contains a number of critical files and folders.

- pages – this folder contains files for designing each of our pages and their functionality.
- providers – this is where we connect our Ionic application to our data access layer.
- app.component.ts – root component for the project.
- app.component.html – root component that contains an ion-app tag and ion-router-outlet tag. These tags will be where or views and pages will be injected, displaying the pages on the app.
- app-routing.module.ts – this file sets up the routes for navigating through pages.
- index.html – injects the application into the web page.

4.3 Spring Boot Application Structure

The Spring Boot Application is our data access layer, which is used to manipulate data in the database, specifically through performing CRUD (create, read, update & delete) operations.

The source code is all located in the package com.restapp.Intouch. Inside this package resides the main class, IntouchApplication.java, as well as four other packages.



4.4.1 Dal Package

The UserDal and ClubDal classes are interfaces that define the CRUD methods for both models. This package also contains the implementation classes for these interfaces.

4.4.2 Models Package

The model classes define the variables that documents have in their collections in the database. This class has constructor methods, as well as getter and setter methods for each of the variables that the user or club has.

4.4.3 Repositories Package

The classes in this package are interfaces that extend the MongoRepositroy interface. These classes allow Spring to access the Mongo database.

4.4.4 Controllers Package

The controller classes contain the API's functionality. These classes are the ones that perform the CRUD operations. It contains the methods;

- getAllUsers – returns the entire collection of users from the UserRepo class.
- createUser – adds a new user to the database by passing the input to the save method in the UserRepo class.

- getUserById – searches for a specific user by their ID, calling the findById method in the UserRepo class.
- updateUser – finds a user using the findById method and maps new data to the document by calling the set methods in the model classes.
- deleteUser – searches for a user using the findById method in the UserRepo class and delete's that document from the database.

There are corresponding methods with the same functionality in the ClubController class.

4.4.5 IntouchApplication (Runner class)

This class is the main class, or runner of the Spring Boot app. This is the class that is pointed to by the MANIFEST.MF file so that it can be executed through a JAR file, which is important for deploying the app to Heroku.

4.4 Heroku

Heroku is where we will host our database as well as our Spring Boot application. To start off, after creating a Heroku account and a new app in the Heroku dashboard, we installed the mLab MongoDB add on to this new app. Using mLab we were able to get the URL for the database and populate it with a basic collection to give us some data to run operations on.

In order to deploy the Spring Boot application to Heroku, we must first define the main class in the MANIFEST.MF file and export the project to a JAR file. This is so that when we run the JAR file it knows which class to run the project from. From there, we logged in to the Heroku CLI and ran the following command:

```
$ heroku deploy:jar out/artifacts/InTouch.jar --app intutch
```

The Heroku CLI then deploys the Spring Boot app to the intutch app on Heroku.

4.5 mLab MongoDB

The Mongo database contains two main collections: club and user. The club collection contains information on each unique club created in spring which can be viewed in ionic. The user collection contains predefined records which can also be accessed, changed, and deleted. The database is hosted on Heroku Cloud Service.

4.6 Database Model

Club Collection

- clubId – Clubs id, String which is unique to each club record.
- clubName – Club name, String. Can share the same as other clubs.
- clubAddress – Club's address.
- clubEmail – Club's Email.
- clubMobile – Club's Mobile number.

User Collection

- userId – Users id, String. Unique to each user record.
- userName – Users name, String. Can share the same as other users.
- userDOB – Users date of birth.
- userEmail – Users email address.
- userNumber – Users mobile or telephone number.

5 System Evaluation

This section will evaluate each component of the project which will include: The Ionic Application, Spring Boot API, and the Mongo Database. These applications will all be evaluated individually as well as how they are interconnected with one another as part of the architecture discussed in the System Design section. Evaluating each component of the project will be broken down into 3 different sections:

- Scalability
- Robustness
- Maintainability

5.1 Scalability

Scalability involves the process of adding more resources to a program. The two types of scalability are: Horizontal Scaling (Scale in/out) and Vertical Scaling(Scale up/down) Horizontal may involve the adding or removing of nodes when the system is already established, while Vertical adds resources to these nodes which usually

involves the addition of a single computers memory. The project is very scalable, as database allows for as many users and clubs to be created in their collections as well as the creation of other collections if necessary. The Spring Boot API is designed in a way that allows multiple ways that the data can be passed between the front end and the backend to keep in scalable. Finally, with ionic, although with the limited functionality we got working is still quite scalable due to it being able to it being able to have access to the data.

5.2 Robustness

To test robustness of a product would involve any quality insurance method that focuses on testing the strength of the code in the products software. This helps with ensuring that the code is working correctly and how it is being processed.

5.2.1 Spring Tools – API

The API application for REST services is designed to perform well with a stable network connection. It is designed to work without any errors or issues. Testing of the CRUD based requests was done through postman which will be discussed more in depth in the performance testing section later.

5.2.2 MongoDB - Database

The Mongo database, which was created using the Heroku mLab application, was designed in a robust manner. Due to it being a NoSQL database it becomes more flexible compared to a MySQL database due to Mongo's schema being able to enforce any data type. Due to it being stored in the mlab application in Heroku, it is secured in a remote sever which helps reassure that the database will always be running and if any issues were to occur, would be handled by the hosting service.

5.2.3 Ionic – Mobile/Web Application

The ionic application was designed in a way that would allow it access to a remote database and being able to display and edit that data in order to be robust. It performs well under a good network environment so that it can handle different amounts of data being passed to and from it. The application was tested using Chrome and

Mozilla Firefox web applications and postman to test that the requests were functioning properly.

5.3 Maintainability

Maintainability helps with defining how well a system can perform and how easy it will be to maintain the system. This can cover aspects such as the change of code as well as how it can be understood when analysed by someone. Overall the system is very maintainable as no issues will occur when running the project. The classes in spring and ionic were designed in a way to not be extensive as well as the code to keep it easily readable for users. The code is also commented decently to explain functionality of the methods, interfaces, data accesses layers, and the models.

5.4 Performance Testing

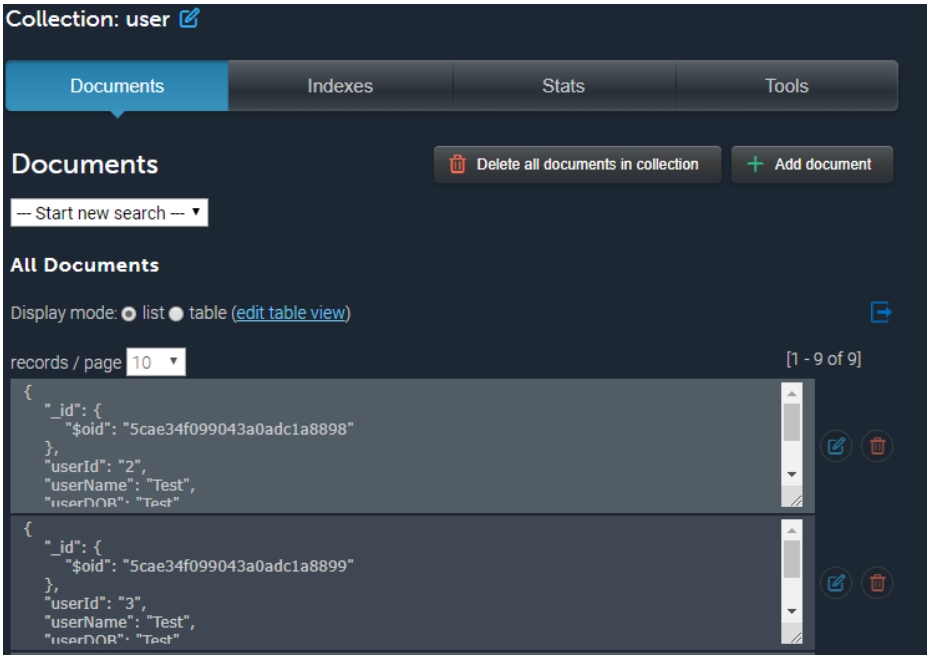
5.4.1 API and MongoDB

When testing the Spring API, we used postman to test the requests we developed in each collection's controllers. For this example, the User collection will be tested for GET, POST, PUT, and DELETE requests. These were all tested locally to ensure that it was working.

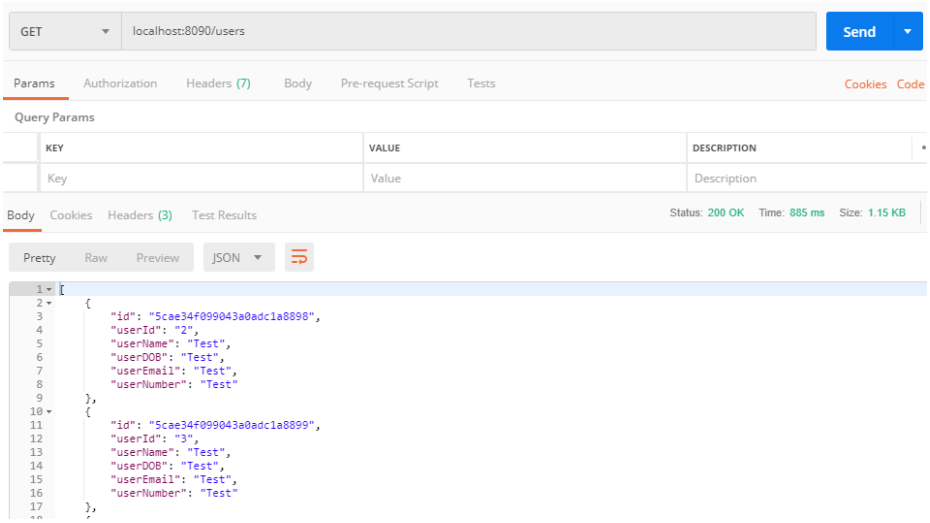
5.4.2 GET

```
// Get users api
@GetMapping("/users")
@ResponseBody
@CrossOrigin(origins = "http://localhost:8200")
public Collection<User> getAllUsers(){
    return userRepo.findAll();
}
```

Setting the mapping request to /users would return all the users that were currently in the repository.



MLab view of user collection containing nine records.

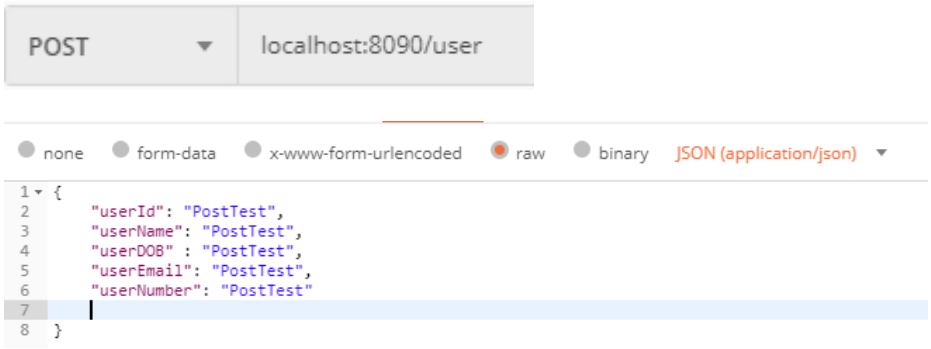


Finally testing the request in postman which returns the nine users.

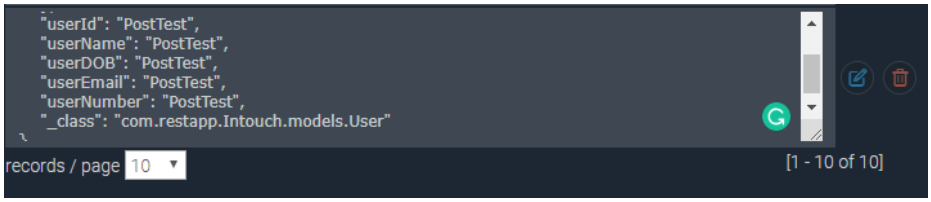
5.4.3 POST

```
// Create user
@PostMapping("/user")
public User createUser(@Valid @RequestBody User user) {
    return userRepo.save(user);
}
```

Create user method which makes a request body of the user model and returns all attributes of a new user record.



Testing post request by passing in a new record through json.

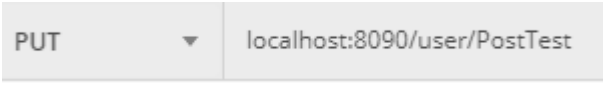


Request works correctly and PostTest gets added to the mLab user collection as its tenth record.

5.4.4 PUT

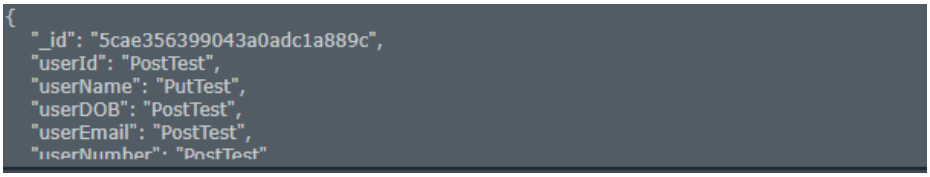
```
// Update user
@PutMapping(value="/user/{userId}")
public ResponseEntity<User> updateUser(@PathVariable("userId") String userId,
                                       @Valid @RequestBody User user) {
    return userRepo.findById(userId)
            .map(userData -> {

                userData.setUserId(user.getUserId());
                userData.setUserName(user.getUserName());
                userData.setUserDOB(user.getUserDOB());
                userData.setUserEmail(user.getUserEmail());
                userData.setUserNumber(user.getUserNumber());

                User updatedUser = userRepo.save(userData);
                return ResponseEntity.ok().body(updatedUser);
            }).orElse(ResponseEntity.notFound().build());
}
```

Update user method which fins user by ID and returns any change to the user's attribute made.




```
{
    "userName": "PutTest"
}
```

The URI must contain the users correct ID for it to be updated, for this example it will be the PostTest ID. PostTest username will be changed to PutTest.

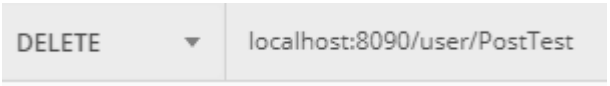


Username has successfully been changed in the mLab collection.

5.4.5 DELETE

```
// Delete user
@DeleteMapping(value="/user/{userId}")
public ResponseEntity<?> deleteUser(@PathVariable("userId") String userId) {
    return userRepo.findById(userId)
            .map(todo -> {
                userRepo.deleteById(userId);
                return ResponseEntity.ok().build();
            }).orElse(ResponseEntity.notFound().build());
}
```

Delete user request which searches for user's id to be deleted in the collection.



When the user ID is specified in the URI, that record is deleted and from mLab can see that that the records have returned to nine.

5.5 Objectives vs Outcomes

5.5.1 List of Objectives

- A user login and registration system.
- CRUD operations working with the spring AP
- Users can create clubs.
- Users can search for clubs by area or hobby.
- Owners of clubs can create events.
- Event interactions such as
    - Mark as interested or attending.
    - Inviting other users.

    - Deleting event.
- Host the data access layer on a server.
- Work with the given learning outcomes for the project in mind, with regards to research, methodologies

When reviewing the set objectives, we wished to achieve during the project life cycle and comparing them with the outcomes, that unfortunately most of the objectives that have not been fully implemented or haven't been fully accomplished. The InTouch application has a functioning API that works with getting information down to the ionic application locally, but unfortunately is unable to perform the extra functionality we wished to achieve with it remotely. The applications are easy to understand and access but not to level we wanted such as allowing users to be able to create clubs and join them, hosting the data access layer on the server, and having a event interaction system.

5.6 Limitations with the system

As discussed, when comparing the objectives and outcomes, unfortunately a lot of the functionality on the ionic side is not working as intended. With little documentation on ionic 4 as it was newly introduced this year, as well as problems faced with the spring api with mongo, caused the development cycle to be pushed backed that it began clashing with other objectives we needed to complete. This was unfortunate but have learned from mistakes we made as we were not familiar with setting up a three-tier architecture before. Internet connection was seen as another limitation that was faced, as not having a stable network connection would not allow for the transfer of data from the mongo database.

The system was created in the mindset the application would have access to a internet connection, and although the API those work locally, it will not be able to access the mlab application on Heroku. Having the system have the ability to work remotely and locally with alterations with the REST API and MongoDB's location, would allow the mongo database to be run using Local Area Network (LAN).

6 Conclusion

To conclude with this project, we have gained a lot of experience with how we were able to setup connections with vastly different technologies and although we didn't reach the standard that we set out to achieve, we learned from the experience and what we can from that in the future.

When we set out at the beginning of the project, we broke down the project into three separate sections that each team member would take. Unfortunately, from unforeseen circumstances of losing one of our members midway through the project, we had to split it between two members which altered our plans and objectives. We expanded the researching section until January and only started development in January. This was poor planning on the team's part as we should of discussed more and finalised the technologies we would use rather than leaving it till later with technologies we were sure about.

With these changes to the planning and development, we lost sight on the objectives we set out to achieve causing the development to take longer than we anticipated. Time management was a key issue we faced as when we first approached the project through a agile methodology, tasks began to not be achieved at set deadlines. Some days had disagreements with the team members, but these must always be taken into consideration as without these learning curves, no progress would be achieved. The team had an overall design in place in the beginning weeks of planning but had no subject to base it off. Fortunately, with some time allocated to workshops and brainstorming we came up with the idea of the InTouch application. We setup milestones for are planning, but we began to not meet them in the development cycle and began taking a waterfall approach. GitHub was useful for helping us in terms of production and setting up branches to keep work organised and to not conflict we each other.

From looking at the technology review chapter, we gained a lot of knowledge with tools we used. Learning about Heroku and the ways we could access, and use built in applications in it was interesting and flexible to configure. The majority of the code was designed by us with help for certain sections found on the web referenced. IntelliJ helped with having a built-in command line runner which helped monitor the data being passed, and the requests being made. The command line was used for updating are GitHub with git pull and push requests. The architecture we designed went through a few processes as we were unsure of what technologies we ultimately wanted to use. Ideas of using neo4j for the database and Android Studio with Android Development Kit for the front end came to mind, but were scrapped with technologies we had some knowledge in.

6.1 Learning Outcomes

We achieved a lot of outcomes with the planning and development of the cycle:

- Teamwork: Being a part of a team was a very important experience for us as it helped with are time management skills as being able to setup meetings with each other and allocating work to achieve help and help us get a better understanding on how it would be like to work in the industry. Not every decision made or tasks set was always agreed on and disagreements would occur, it still gave us good experience as problems will always occur with group projects and the ways they are dealt with as we will not always have the opportunity with people we want to work with
- Communication – Having weekly meetings with are supervisor as well are own scheduled meetings, helped improve are communications skills. It is important, especially at team to ensure that every teammate had a say on how they wished the project to progress. Having to discuss are plans with are supervisor helped with expressing the thoughts we had with the project and to have someone understand the methods we were going to take to achieve our goals helped greatly. We found when we met and sat down that work was achieved very often. This module helped us understand that communication is necessary in software development as work can be achieved at a much faster pace and more efficiently.
- Technologies – We choose technologies that we briefly used in modules throughout the course to help us get a better understanding of their frameworks. The experience was both interesting and challenging at the same time, as we applied these technologies to do tasks that we haven't used them for before. There was a curve of returning to something familiar but to also learn something new with that technology. The technologies used were of modern design while still being innovative and new especially with ionic. With the research conducted of these technologies, we were all up to date with the latest versions and understood what we could do with them.
- Problem Solving – With any type of software there's always problems or errors that occur that need to be solved and figured out. Finding tools outside the technologies we were using such as postman, helped with finding out the problems that we may not have necessarily seen or expected. Researching how the technologies could be used helped with understanding how they could be compatible with the project.
- Documentation – Making sure that the code was understandable and to give the knowledge of what each method, function, etc was doing was a key skill when designing the project. Team members would make notes in are social media chats of what tasks were accomplished and whenever time would pass with the project, we would always know what we would need to achieve next.

6.2 Final Thoughts

As we finalise and submit are project, we reflect on the project life cycle. Although we did not fully achieve what we set out to accomplish, the knowledge we gained and the experience we shared help us shape are software skills and helped improve upon them. We learned how to work as a team and kept a clear focus on how we envisioned the project.

7 Appendix

7.1 Github URL

https://github.com/yrrag5/Final-Year-Project-

7.2 Running the Project

Once the repository is cloned open your IDE in this case IntelliJ and import the Intouch spring project. When the project has been selected with Maven import project the snapshot should be able to be imported. JDK 8 was used for this project so ensure this is the version selected. Once that's completed the project will open correctly. Select the IntouchApplication class and click play to run the Spring API. This will get a handle of the remote mongo database on Heroku and open it to crud operations. With Ionic, the latest tools and plugins will need to be installed for it to run on visual studio code. The application will need to run the ionic serve command in the command prompt for it to execute. Once done the application should open a web browser where the mongo database can be viewed.

8 Bibliography

1. Phillip Webb, Dave Syer, Josh Long, Stéphane Nicoll, Rob Winch, Andy Wilkinson, Marcel Overdijk, Christian Dupuis, Sébastien Deleuze, Michael Simons, Vedran Pavi, Jay Bryant , Madhura Bhave. Spring Boot Reference Guide – 2.1.3 RELEASE
2. Matt Raible. Develop a Mobile App With Ionic and Spring Boot
3. Joyce Lin. The Ultimate API Publisher's Guide