Spot Finder Android App

Shane Moran

Sean Moylan

B.Sc.(Hons) in Software Development

May 10, 2020

Final Year Project

Advised by: Martin Kenirons

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

Contents

Chapter 1

Introduction

The idea of the project is based on the problem that BMX riders, skaters and other outdoor sports do not always have available locations for recreational use at convenient times and infrequently summer.

1.1 About the Project

Skaters, for the sake of innovation, transparency and citizen participation are interested in outdoors sports such as BMX riding and skateboarding. This innovative technology allows for the decentralization of pay to play organised sports, allowing citizens to interact in an immutable, safe, and reliable environment. Throughout this study we will see the problems detected during its deployment phase, the solutions adopted and the conclusions for its use recreationally and commercially. The result will be achieved with the elaboration of a proof of concept prototype.

The purpose of this Final Year Project is to build an application using a new technology that the developers have never used before. Therefore, it allows for research to be done, learning, coming up with technologies and discovering new ways of developing software. The front-end technology being used to create the mobile application is Android Studio. Android Studio was chosen because of its new and gives the opportunity to learn a new Mobile Application framework. It will be connected to a back-end server running Flask that accesses a Mongo Database. This way, the developers will be able to show they have worked and have a fully functioning app that will satisfy what was required from them. The technologies used will be explained and why they were used throughout. This document will contain steps taken in creating the project and they will be documented clearly. The research and

development will all be detailed and allow others to understand why specific software was chosen. It will in detail compare similar technologies that could have been used. These sections will demonstrate the similarities between specific frameworks helping to explain what technologies were needed for the development application. This project is implemented using technologies that have not been used together too often which may lead to some issues. This means any problems faced or certain aspects found to be new will be documented along with the development.

1.2 Context

The context of this project revolves around having fun with action sports anywhere, wishing to find out more spots to BMX or skate. Opening the app on the user's phone, they can view the various spots via Google Maps. This application prevents the user getting lost and gives them accurate information about each location. The information on the application will be provided by the users, to provide variety. Reading the data and retrieving images from the database needs to be very fast, as any delayed hang in performance could lead to a bad user experience. The project will be developed as an android application which provides users with information, and a map of the various spots. Users of the application can find their nearest spots easier in relation to their current location at any given time. When the user decides they wish to learn more information about a specific location, they can retrieve text about the spot on the application. Along with the mobile application a web application will be developed for use, to update, delete or add information to the spots database.

1.3 Objectives

The project will require a number of objectives to be accomplished in order to provide a solution that works and is suitable for use by outdoor sports enthusiasts. [1]

- A Mongo database will be used to store the text which appears in the application. This database will need to be setup and hosted in such a way that it can be accessed from clients through the android application.
- Pull information from the MongoDB using flask.

- A client mobile application will be the main product / asset for the project. This application will be able to locate the user via the GPS on their mobile device. The app will then show the user the locations of each spot. This will provide the user an idea of how far they are from the chosen spot and will allow them to see when they are near their destination. Each location is specified on the application as a list on the home page. Then chosen images and information will appear for each location.

- An additional requirement is to develop a web application for the use of the company to allow them to modify the information shown to the user. The web application is a simple website that can access the database to create, update and delete information from the database.

- Using the Google Maps application programming interface (API) within the mobile application to show the user their location along with the specific spots. Each spot is indicated by a marker in the map. When the user clicks on a specific marker, the name of the location appears as a label.

1.4 Overview

Spot Finder is an Android Application designed to help skaters locate popular spots around the globe that they can skate at. From a city plaza to a stair set located behind a rural building, this app is made to help the skater locate areas for them to skate. This means if someone was to travel to a new country/city they would be able to easily locate areas that local skaters have saved on the app without having to painfully search around massive areas for a certain type of spot they are looking for e.g skatepark, outdoor skatepark, stair set, hand rail, plaza etc. [2]

Chapter 2

Technology Review

2.1 Collaboration

The idea came through Sean Moylan for making a spot finder app. This application will enable the easy accessibility of information for outdoor sports people. All images and information included in this application was provided by users. We began the planning stage of the project with an initial meeting with our mentor (Martin Kenirons). The initial meeting took place in GMIT in October 2019. During the meeting, we discussed exactly what was wanted as an outcome of this project by Martin as set out in the objectives. Meetings took place once every two weeks from this point to mid-February, but due to the unforeseen circumstances surrounding the spread of the Coronavirus; we were unable interact in person and this led to meetings having to take place online via Microsoft Teams. that in order to iron out all the details in relation to design, requirements and any problems regarding the project. From January the development of the project, during these meetings the developers produced a working version of the application. This allowed Martin to give immediate feedback and change any of the requirements. Martin began to give the developers the information needed for the application in December and January in order to begin population of the database.

2.2 Technology

During the initial project planning, such frameworks that were considered included Android Studio, Ionic, React native and Flutter. But we chose android studio. Android studio provides a flexible Gradle-based build system. It builds variants and multiple APK generation. It has expanded template support for Google Services and various device types. It provides a rich

layout editor with support for theme editing. Line tools to catch performance, usability, version compatibility, and other problems. It has ProGuard and app-signing capabilities. Its built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine. [3]

The back-end services researched include Docker, MySQL, MongoDB, Cassandra, Flask, NodeJS and Django. After much research and reviewing of examples Flask and MongoDB were chosen.

2.2.1 Android vs Ionic

Android Studio and Ionic are similar in what they offer with regards to prebuilt components in Android Studio and a comprehensive suite of built in widgets in Ionic [4]. With Ionic, you create a real native app but you do this by creating a web app (with Hyper Text Mark-up Language, JavaScript and Cascading Style Sheets) which will be wrapped by a real native app that hosts a web view while Android studio you tend to write Java (or Kotlin) code which can be compiled on the target device. The main reasoning for using Android studio over Ionic is that it is reliable and resourceful. It is powered by Android and therefore is well documented and although it is still relatively old there are loads of interactive talks online about its upcoming, current and new feature. Ionic has numerous third-party built-in tools. It has thousands of threads on Stack overflow and packages on NPM (node.js package manager) to help developers along the way. Ionic provides all the functionality which can be found in SDKs of native mobile environment. Ionic application enables inventor to build their applications and customize them according for different platforms such as for iOS, for Android, and deploy through Cordova. It includes various mobile components, extensible base themes and typography. Ionic uses framework of JS i.e., Angular, which offers custom components and method to interact. Therefore, Ionic is a backdrop support which operate on top of Cordova to build hybrid apps. It allows to develop an application interface which are comparable as they were web pages i.e., using HTML, CSS, JavaScript. And these apps run inside WebView of native application. Virtue of ionic framework is – it is an open source framework in which code needs to be written once and execute on numerous mobile devices. The use of only one programming language enables the app on all the different mobile OS. Availability of various plugins to use hardware and other extra services for mobile app. Ionic apps has the well build graphics. This is the developer's favourable framework where simple technologies are used to build the entire interface of the application. [5]

2.2.2 Android vs React Native

React Native is similar to Android development and compiles to native application by default. A basic React Native application is given a basic set of components. The developer must style most of them separately for each platform. This creates more work for the developer and increases time of development and cost of development for a company. React Native is developed by Facebook, while Android was various developers, but most notably Google. Each company is well developed and looked upon favourably by smaller businesses. React Native applications are developed using JavaScript and React libraries to build user interfaces. React previously existed to create web applications. Android is developed using various programming languages such as Java, C, C++, Kotlin and is solely used for mobile application development. Android is a very popular platform with over 980,000 results on GitHub in comparison to 145K for React Native. This is because Android is older and developed by Google, therefore it is becoming more and more easier to develop in each day. Both are primarily classified as "Frameworks (Full Stack)" and "Cross-Platform Mobile Development" tools respectively. [6] The key thing about React Native is that it's still in development and we are yet to see its full potential. In the future, it may be more powerful and efficient and allow for even more use cases, but for now it cannot fully replace native mobile development. However, its write once, use everywhere paradigm can be a tremendous time and money saver if used on the right projects. [7]

2.2.3 Android vs Flutter

The original reason for using Android was the fact that Flutter is so new and the idea and challenge of learning a new cross platform mobile application platform did not sound appealing. Also, learning a programming language (Dart) with less resources to that of Android, would not be ideal. Android provides you with the API libraries and developer tools necessary to build, test, and debug apps for Android. It also provides a rich application framework that allows you to build innovative apps and games for mobile devices in a Java language environment. Flutter is Cross-platform mobile framework from Google as well and is a mobile app SDK to help developers and designers build modern mobile apps for iOS and Android. Android SDK belongs to "Frameworks (Full Stack)" category of the tech stack, while Flutter can be primarily classified under "Cross-Platform Mobile Development". "Android development" is the top reason why over 280 developers like Android SDK, while over 13 developers mention "Hot Reload" as the leading cause

for choosing Flutter. Flutter is an open source tool with 69.5K GitHub stars and 8.11K GitHub forks. Here's a link to Flutter's open source repository on GitHub. [8]

2.2.4 Comparison

Native applications for Android are built in Java or Kotlin, and native applications for iOS are generally build in Swift. For example, when developing in Flutter there is a single code base; this means the application is written once and it works for both iOS and Android. Third Party libraries are widely available for native language applications. This is due to the popularity of their language and therefore there are very few problems that cannot be solved by referencing websites such as Stack overflow, etc. , but, Native languages and Flutter both give a native application appearance. [9]

2.2.5 Why Android?

We opted to develop an Android app for many reasons, other than the fact that it has the most resources. After a good deal of researching other options of Mobile application development, Android came out on top by far. Firstly, the main reason is the use of java as a programming language makes it easy to port the app to multiple operating systems like Symbian and Ubuntu. Thus, businesses can target multiple platforms with Android app development. Android apps have a rapid development cycle lasting a few hours. It offers a competitive edge to companies who wish to have a quicker go-to-market for their new idea. Reduced Time to Market (TTM) is, thus, one of the best benefits of Android development. Another one is its availability of the Android SDK. The development teams can use the material design from these SDKs to build interactive apps. However, developers are required to pay a one-time registration fee for application distribution. After that, they can leverage any computer device to build and test the product for their smartphones, ensuring low investment and increased user engagement. In turn, the end users, are benefited by an interactive app, and the enterprise gains a higher return on investment. Android P introduced several additional and in-built security features. It will help with the protection against malware and viruses. Thus, safety and reliability are exceptional benefits of android application development. To conclude, with over 75 per cent of Android device users today, developing an app on this platform is a value proposition for organizations globally. It helps them address a higher range of audience and gain control over the market. [10]

2.3 Server

Flask is a micro web framework written in Python. A framework, in the simplest terms, is a library or collection of libraries that aims to solve a part of a generic problem instead of a complete specific one This means flask provides you with tools, libraries and technologies. Micro-framework is normal framework with little to no dependencies to external libraries. Whereas, Flask being a micro framework because it implements only core functionality such as routing; It then leaves more advanced functionality such as authentication and database ORMs to extensions. The result of this is less initial setup for the first-time user and more choice and flexibility for the experienced user. This is in contrast with larger frameworks, such as Django, which dictate their own ORM and authentication technologies. [**?**]

2.3.1 Pros

Pros would be that the framework is light, there are little dependency to update and watch for security bugs,

2.3.2 Cons

Cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins. [11] It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks. Flask offers suggestions but does not enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy. [12]

2.3.3 Python Flask vs Node.js

Choosing a language back end is an important step to the development planning phase of any project. While flask is written is Python, node.js is JavaScript which allows JavaScript run on the server side and not in the browser because it is an environment. Flask is a micro framework for Python based on Werkzeug, Jinja2 and good intentions. The comparison of Flask and node.js breaks down to a comparison between Python and JavaScript. The performance at the back end of any application is important for the speed and

response time of the application. Both flask and node.js have good performance however node.js has better performance this is because it is based on chrome V8, a fast and powerful engine. Flask is sufficient once application is not memory intensive. Scalability of the application when running on node.js is optimum as node.js supports an asynchronous application. This ensures for smooth, fast scaling. Python's Flask does not support traditional asynchronous programming, but this can be mimicked with co-routines, making asynchronous processing achievable.

2.3.4 Python Flask vs Python Django

Django is a full-stack web framework for Python, whereas flask is a lightweight and extensible Python web framework. Django is focused on the final product alone, it is an all-inclusive experience. Flask focuses on the experience and the learning opportunities. It provides simplicity, flexibility and fine-grained control. Although Django comes with everything built in, it is difficult to change predefined platforms, libraries etc. Although Django was released first in 2005 in comparison to Flask in 2010 the latter has outgrown Django and is a more popular framework today. Although both are easy on the eye, some would say that Django has underpowered templating and Object-Relational Mapping. [13]

2.3.5 Python Flask vs PHP

Another alternative was the popular general-purpose scripting language that is especially suited to web development, known as PHP. Its Fast, flexible, and pragmatic, PHP powers everything from your blog to the most popular websites in the world. But its inconsistent API, fragmented community and lack of a routing system lead to us choosing Flask. [14]

2.4 Database

2.4.1 MongoDB

MongoDB is one of many non-relational databases. It is a NoSQL database that is used across many areas in the technology and business sector. MongoDB is a free open source database management system. As it has evolved over the years it is now one of the most popular document-oriented databases. It is run on a Mongo server and can be created and controlled from your command prompt. Data in MongoDB is stored in JSON like documents.

A format called BSON which is JSON in binary style issued for document storage in the database. This allows for an easy to read format of the data. Dissimilar to relational databases for example MySQL which uses rows and tables as a database structure MongoDB is formed of collections and documents. Each database in Mongo can have multiple collections. " A collection is a group of documents". These documents can have many fields. The Documents in the database are a set of key value pairs. An id or in our case, a user will be assigned to the documents making your database easy to edit. As MongoDB is a NoSQL non-relational database it differs in many ways from a relational database management system. A relational database is row and column based as opposed to document and field based. MongoDB is considerably easier to set up and is not vulnerable to SQL injection. A RDMS can be more challenging to understand and lets down the database in terms of its hierarchical storage not being as good and how it is open to SQL injection. One of the more popular relational databases is MySQL. Compared to MongoDB, it is quite inflexible in terms of the database schema. MongoDB is known for its ability to" handle large amounts of unstructured data" which MySQL lacks in its technical features. Improving databases in terms of storage and their schemas can help to build more efficient and easier to understand technologies. MongoDB runs better than many other database systems used today. It is a high-quality technology which has features that are desirable in having a secure database. Known for its scalability, memory processing and concurrency MongoDB makes for a desired database. All these features make it easy to develop a fast, reliable database which makes it work alongside evolving applications and is sought after by businesses. It is flexible and can be adapted to work with many platforms and can be integrated into software much easier than other database management systems. The most common frameworks that are used in conjunction with MongoDB are NodeJS and a newer one Mongo Flask. It is also supported by cloud services such as Amazon Web services and Google cloud platform.

2.4.2 MongoDB vs MySQL

There are ultimately two types of databases. These include SQL database and NoSQL databases. MySQL is an example of a SQL database while MongoDB is an example if a NoSQL database. MongoDB is stored in a document like JSON. It is flexible and dynamic because the structure can be changed to meet customers' needs and can be scaled horizontally. SQL is written in SQL query language and can be scaled vertically. MongoDB is a much newer server and was released in 2009 vs SQL server which has been published since 1989. MySQL had document orientated structure model while SQL is a rela-

tional database management system (RDMS) model. Joins, concurrency and foreign keys are not supported by MongoDB - only SQL databases. MongoDB's are created and maintained in an agile development practise while SQL databases are supported mainly by waterfall life cycles practises. This makes MongoDB's more popular with modern companies who use agile development and many older companies are now switching to use MongoDB and agile also. Data Schemes are Dynamic in MongoDB but static/fixed in SQL databases. MongoDB's are more usable and flexible as they can be run on Windows, Linux and os X operating systems in comparison to MySQL which can only be run on Windows operating systems. [15]

2.4.3 MongoDB vs Cassandra

Both MongoDB and Cassandra are NoSQL databases. Large companies such as Facebook use both databases. Cassandra can handle large amounts of unstructured data e.g. Instagram has about 80 million photos uploaded daily to its Cassandra database. MongoDB, because it is schema-free, documents can be created without creating their structures first, in comparison to structures having to be predefined in Cassandra databases. Cassandra databases are queries with CQL querying language. This is similar to SQL querying language. MongoDB currently had no support for any querying languages. MongoDB has queries which are structured as JSON fragments. Cassandra was released in 2008 by Facebook but currently being maintained by Apache software foundation. This is a newer NoSQL database than MongoDB and already has become more advanced and popular among large companies. [16]

2.4.4 MongoDB vs Microsoft SQL Server

Our final option was a relational database management system developed by Microsoft, called Microsoft SQL Server, and is a database management and analysis system for e-commerce, line-of-business, and data warehousing solutions. Whereas MongoDB is the database for giant ideas. MongoDB stores data in JSON-like documents that can vary in structure, offering a dynamic, flexible schema. MongoDB was also designed for high availability and scalability, with built-in replication and auto-sharding. Microsoft SQL Server and MongoDB can be primarily classified as "Databases" tools. "Reliable and easy to use", "High performance" and "Great with .net" are the key factors why developers consider Microsoft SQL Server; whereas "Document-oriented storage", "No SQL" and "Ease of use" are the primary reasons why MongoDB is favoured. MongoDB is an open source tool with 16.3K GitHub stars and 4.1K GitHub forks. Here's a link to MongoDB's open source repos-

itory on GitHub. Microsoft SQL Server also had expensive licensing which there were cheaper alternatives for. [17]

Chapter 3

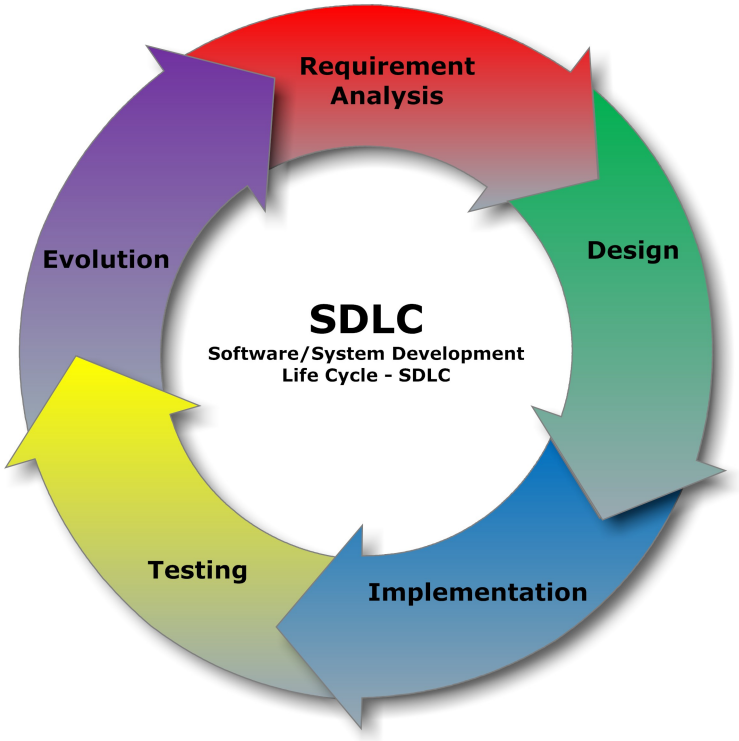
Methodology

3.1 Project Management

Project Management is one of the key elements to ensuring that the project is laid out correctly and that each component part is complete at a given date. Because this project was completed by a team it was important for both members to complete certain aspects of the project. In order to manage this project, GitHub was used as it was very useful due to it being a team project. It enables both members of the team to work on separate individual branches. After each iteration, these branches were merged into the master branch. After each integration, the project was at a working state of the project. This could be used for industry standard as continuous delivery. For this project it meant that after each Agile sprint there was a working version to present to customer and supervisor. Although it is a working version, it may still need changes in the next sprint. At the beginning of each sprint all branches pull from the master so as every team member is working off the same latest release of code.

3.1.1 Agile

The project used Agile project management methodologies. An iterative approach was taken. This meant that each week certain tasks are completed. The project team met once a week to discuss what has been completed since the last meeting, what must be completed before the next and discuss any difficulties experienced. The meetings were short and concise. Once every week to two weeks, the team met their mentor and explained what was completed since the last meeting and is to be done before the next. These weekly meetings were the iterative approach. Each week the process would



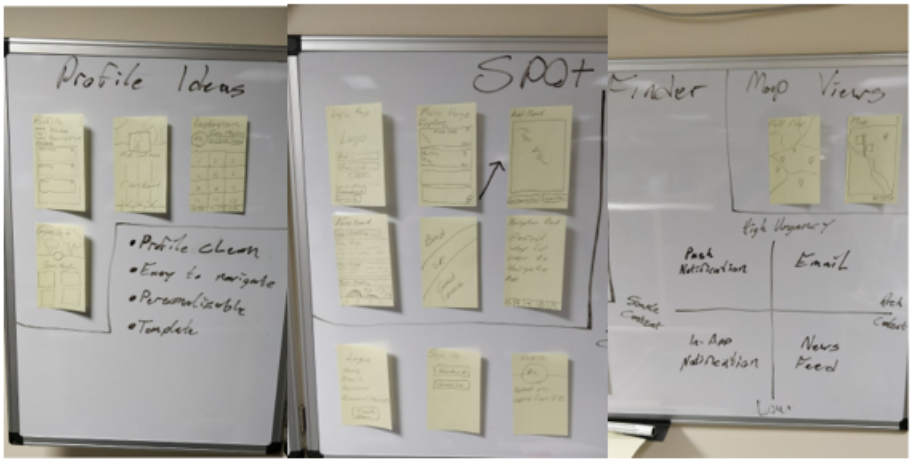
be repeated to meet, plan, design, develop, test, and evaluate.

3.2 Planning

During the meeting and planning phase, the milestones for the project were identified and broken down into simpler, more manageable tasks. These are the tasks which must be complete before each meeting and can be found in the issues section of the GitHub repository. Each sprint started after the weekly meeting with the supervisor and the aim is for the work to be complete before the meeting the following week.

3.3 Design

Throughout the agile development life cycle design is a step in every iteration. The design is gradually built on and is never defined at the beginning of the project. The gradual evolution of the design enables taking advantage of new technologies that come on stream as well as meeting any new requirements brought forward by users. The user experience is key when designing an



Project Design

application and to make sure of that, it was to be made user friendly. The type of end user of the item must be considered when designing.

3.4 Testing

Testing was carried out continuously. When tasks were complete that were stated in the test plan the tests were carried out. If possible, user tests were carried out when meeting with the supervisor. The supervisor used the application and commented on any changes that should or could be made for the next meeting.

3.5 Evaluating and Managing

After each sprint before the meeting phase a review was carried out on the tasks completed in the last sprint. If there were changes to make these were completed in the next sprint. In the evaluation section of the agile development cycle the code by both developers was mergers into the master branch of the GitHub repository. The evaluation process is also used to look back, see how much of the previous iteration that was not complete and change the workload applied for a single iteration. This could also apply at an industrial level where continuous delivery is needed. After each commit, a working version was present for our supervisor. But this might mean that future changes are still forthcoming.

Chapter 4

System Design

All static text is stored in the MongoDB hosted at http://localhost:27017 . This information is accessed by the Android application though the Python flask server which hosts the Mongo database. Each row of the database is called individually to display the appropriate information, update, and delete a different spot. When creating a spot in the database, a new row is added to the end of the database for each entry. The web application both sends and receives information to and from the database while the Android application only receives information from the database.

4.1 Types of Technologies

4.1.1 Server

Mongo and Flask are used to develop the back-end server for the project. It is coded in Python and connected to the MongoDB database. The database was setup in MongoDB using atlas. The MongoDB database contains two collections, user and location. The collections are then made up of documents that contain the information for each user and location. There is a name for every location and piece of informative text about the location in the app. The description contains the long pieces of text in the database. Each user is intended to have a username, email and password. Flask was used as a web framework in this case to allow access to the database and display and edit its contents. This was achieved by connecting MongoDB and Flask to implement CRUD operations. The flask server is run from the 'FlaskServer.py' file. This file contains the route to show the running server. The Flask server was then connected to MongoDB. The PyMongo api allowed the connection between these two frameworks. PyMongo is imported

in Flask. In the FlaskServer.py file, it calls the Mongo URI. This allows it to recognise the database. The data types stored in the database are converted to JSON string also in this file. This allows readable interpretation of the data. The CRUD implementation in the Flask allows specific routes to access the database. The collection 'user' or 'location' are referenced to ensure the correct data is being accessed. On the android side, the information in the database is accessed by using retrofit to send requests to the server, the server receives the request and points it the the path specified in the request e.g /user/login. From here the data can then be manipulated in the database to either create a user, create a location, login a user, get all users, get all locations etc. The server will then give back a response which will change depending on the request it receives. If it were requesting all the locations then the server would retrieve them from the database and pass them back to the app in Json format.

Connecting Android with the Database

The information displayed in the Android Application is being stored in the database and not the app itself. The text information is being read into the app from the database, so it is more secure. This was done so the application does not contain large amounts of text. This allowed testing to be done to see how Android works with reading from a database as it is a modern technology that has been used in the app development sector. Due to androids' vast libraries and amount of open source code, little research and testing many different URL readers to fetch the data and be able to display it in the correct way. The database was changed many times. In the most recent database, it contains a id, and a description.

Each user or location is in its own document and has its own unique id in the database. This allowed for each user or location to be read in separately so the exact text could be specified in the application. This database connection is done in Location.java, Login.java, Tools.java and User.java. The same string readings are done in the classes. The URL is read in as a string. The URL is of the Data that is displayed in a JSON format when you search the URL. A method is then used to read any data that is in JSON format. This then will use a set state in the application that reads the data contained in the 'multi' document that is in the database. The set state is used to rebuild what the user calls inside Android. This means all the data is read in the app to the console. This also decodes the JSON to be able to display it as plain text. To display the text in the app it is completed in the widget. Android uses a list view builder. For the data that has been read in to be transferred to the list display an item count is used. This reads the data variable from

the JSON data and add the length of the database. It has been changed to one in this case, so it only reads one piece of information when it is called. A container in the app is what displays the text. The variable is called and has an int and the name of the fields in the database being used. The int is referencing the number each document in the database is given. When you open the app, there is an option of "MY LOCATIONS" or to "VIEW LOCATIONS" .

For example, when you click view my locations, it can be seen in a recycler view that recycles XML documents that are populated with location data. To use the database in an efficient way with Android the way the text was being called was changed multiple times. With the research involved, the only way it seemed possible to display the text was in a recycler view.

This meant there had to be a string used in calling the specific pieces of information. This was as the item builder contains the Build context and a string. Without the list view builder an item builder was not allowed. When the data variable was used with a field from the database it could only print one description as all the data in the collection uses description fields. The data variable had to be used so the text was not being displayed as JSON text. The data variable could not be used with two fields being called as only one is used in the set state method. This meant this was the only way the data could be displayed in a correct way that suited the database. The containers mean the text is displayed in card like paragraphs that allow for an easy to read display in the application. The text documents and pictures will fit nicely beside each other and the text can be aligned nicely.

Web App

Initially, we had planned on making a web application which should at least support basic CRUD operations which has been developed for the administrator to edit the data. But, that plan never came to fruition, as we didn't get to implement a web app.

4.1.2 Client

Mobile App

Android studio was used to develop the mobile application intended for Android devices. Research was carried out to learn java programming language and learn the syntax of Android along with understanding how widgets work correctly. When creating the application, initially all pages were split up into two classes. This made the code very long and caused a lot of confusion. At

this stage the developers decided to break up the code into a class for each location or spot. The main page holds the home page of the application along with the splashscreen of the application. The spashscreen has a "startActivity" method. This method calls the timer function which is an async method. The maps page contains all elements for the Google maps used within the application. The map view library is imported at the beginning of the file. This enables the creating of a Google maps image in the application. This Google maps image however is places on top of the widget. This is a key feature in the development of the project. The map view is a separate widget that in linked to directly from Google maps api. This is called from within a stateless widget. When the back button is selected the used is returned to a empty page. For user friendliness a message to go back one more page is presented to the user. Therefore, a specific location stands out on the map when clicked, and a label appears over the location if visited; if not, they have the option add a spot. Each page containing the locations of the spots are laid out in similar format. One of which page contains multiple photos in a scrolling slideshow. This user must select previous/next button to navigate through the images. The images are stored in a list and list is controlled by a photo Index to see what number of the list is currently being displayed. The text and URL location for each page is taken directly from the MongoDB Flask database.

4.2 System Integration

4.2.1 Sublime Text

This popular cross-platform text editor with a Python interface was used to code the server which can found in the FlaskServer.py file and is started via the terminal using export $FLASK_APP = FlaskServer.py$.

4.2.2 Postman

Postman then tests the requests from the server. Before testing, an interface of requests must be created. Then, we can call them from whatever activity the information is needed. [18]

4.2.3 Retrofit

Firstly, what is Retrofit? Retrofit is a REST Client for Java and Android. It makes it relatively easy to retrieve and upload JSON (or other structured

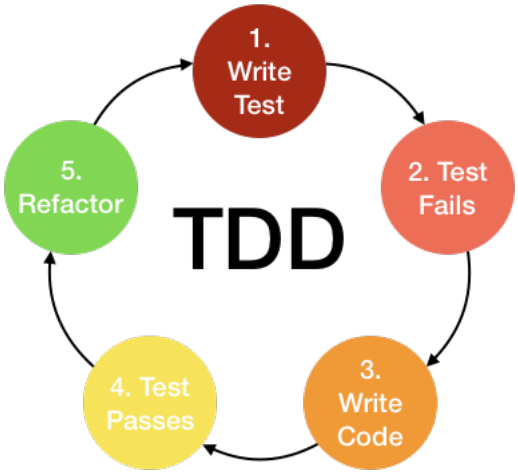


Figure 4.1: Caption

data) via a REST based webservice. In Retrofit you configure which converter is used for the data serialization. [19] Why it was used? Once the requests worked, I used to retrofit to make calls to the server from the app. [20]

4.2.4 Types of Testing

Test Driven Development

Test-driven development is an evolutionary approach which combines test-first development where you write a test before you write just enough production code to fulfil that test and refactoring. What is the primary goal of TDD? One view is the goal of TDD is specification and not validation. In other words, it is one way to think through your requirements or design before your write your functional code. Implies that it is both an important agile requirement and design technique. Another view is that TDD is a programming technique. [21]

Before writing implementation code, the developer writes automated unit test cases for the new functionality they are about to implement. After writing test cases that generally will not even compile, the developers write implementation code to pass these test cases. The developer writes a few test cases, implements the code, writes a few test cases, implements the code, and so on. The work is kept within the developer's intellectual control because he or she is continuously making small design and implementation decisions and increasing functionality at a relatively consistent rate. New functionality is not considered properly implemented unless these new unit test cases and

every other unit test case written for the code base run properly.

Some professed benefits to TDD are:

• In any process, there exists a gap between decision (design developed) and feedback (functionality and performance obtained by implementing that design). The success of TDD can be attributed to reducing that gap, as the fine granular test-then code cycle gives constant feedback to the developer. An often-cited tenet of software engineering, in concert with the "Cost of Change" [22] is that the longer a defect remains in a software system the more difficult and costly it is to remove. With TDD, defects are identified very quickly, and the source of the defect is more easily determined.

• TDD tempts programmers to write code that is automatically testable, such as having functions/methods returning a value, which can be checked against expected results. Benefits of automated testing include the following:

1. Production of a reliable system
2. Improvement of the quality of the test effort
3. Reduction of the test effort
4. Minimization of the schedule.

• Test cases create a thorough regression test bed. By continuously running these automated test cases, one can easily identify if a new change breaks anything in the existing system. This test bed should also allow smooth integration of new functionality into the code base. With XP, developers do little or no up-front design before embarking in tight TDD cycles consisting of test case generation followed by code implementation. However, many of the benefits listed above can be realized in essentially any development process simply by shifting from unit test after implementing to unit test before implementing. [23]

Behaviour Driven Development

Behaviour Driven Development is an agile software development approach that encourages collaboration between all project participants. It is an evolution of Test Driven Development and Acceptance test-driven planning. The most important core principle of Behaviour Driven Development is that "business and technology people should refer to the same system in the same way" [24]

In order to achieve this objective, a common Language is needed for specifying system behaviours, allowing customers to specify requirements from a business perspective, businesses analysts to attach concrete examples clarifying the system behaviour, and developers to implement the required system behaviour in a Test Driven Development manner. The second core principle of BDD says that "any system should have an identified, verifiable value

to the business". Organizing the development effort around the system behaviour could achieve the first objective, but how could we know when we have delivered a behaviour? If the behaviour is described using executable scenarios (executable acceptance tests) then the software can be automatically verified through successful passing of the tests. When Dan North introduced this concept, he proposed an ubiquitous language for analysis, so that the requirements can be captured into the codebase. This language represents the requirements as user stories, and the acceptance criteria as scenarios attached to user stories. He proposed the following standard form for writing scenarios: given some initial context, when an event occurs, then ensure some outcomes.

Comparison

TDD is a test of component in isolation and is an example of Unit testing, whereas BDD is a system test. It is also a behavioural test. While BDD focuses on the value, TDD cares about the quality. In TDD, tests are written at the same time as the code; so, the developer writes and reads the tests, the testers also read the code. Whereas in BDD, the person that understands the customer best writes the tests and almost anyone can read the tests, such as developers, testers, stakeholders, business and product owners. With Unit tests, you know specifically what did not work unlike BDD where you don't. Unit tests tend to be quicker, due to behaviour tests being done as a whole. Although, Behavioural Tests tend to be high, but 100 per cent of code coverage is not uncommon. Regarding maintenance, Unit tests change first; whereas with BDD, the high-level nature of Behavioural Tests mean that they change infrequently. BDD testing is portable unlike Unit tests as it not coupled to the code. [25]

Acceptance Test Driven Development

Acceptance Test Driven Development involves team members with different perspectives collaborating to write acceptance tests in advance of implementing the corresponding functionality. The collaborative discussions that occur to generate the acceptance test is often referred to as the three amigos, representing the three perspectives of customer (the problem needed to be solved), development (how we solve the problem), and testing. These acceptance tests represent the user's point of view and act as a form of requirements to describe how the system will function, as well as serve as a way of verifying that the system functions as intended. In some cases, the team automates the acceptance tests.

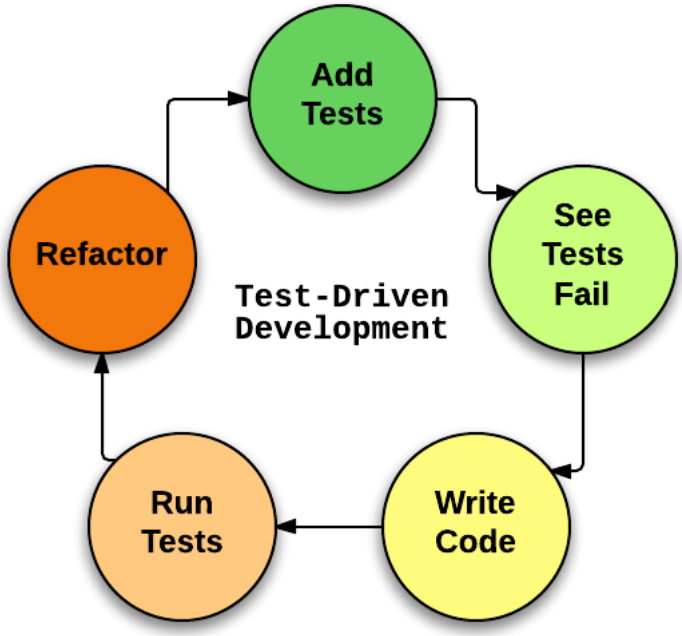


Figure 4.2: Acceptance Test Driven Development

Benefits

Just as TDD results in applications designed to be easier to unit test, ATDD favors the creation of interfaces specific to functional testing. (Testing through an application's actual UI is considered less effective.)

Pitfalls

Even more than the use of automated acceptance tests, this practice is strongly associated with the use of specific tools such as Fit/FitNess, Cucumber, or others.

One major risk, therefore, is that the tool chosen will hinder rather than advance the main purpose of this practice: facilitating conversation between developers and product owners about product requirements. Tools should be adapted to meet product owners' needs rather than the other way around. [26]

Chapter 5

System Evaluation

Since the first meeting with our supervisor back in October, we can say that we have set met the objectives set out for this project; some of which can be seen above in the objectives

5.1 Testing

Firstly, we chose our technologies that would be used; Therefore, we chose the appropriate three-tier architecture. We needed a back-end server, a database, and an IDE or framework to develop the app.

Then, we had to decide over which parts of the project each individual would be responsible and what technologies would be implemented. Initially, regarding the server and database, we had alternatives we were think of using, but opted not to. For example, we thought about using MySQL as our database and Django as our server.

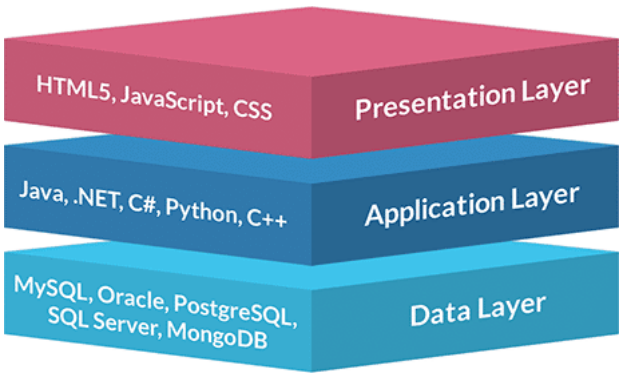


Figure 5.1: 3-Tier Architecture

5.2 Problems during Testing process

5.2.1 Back-End Problems

The server and database were first set up using firebase and then opted for Flask and MongoDB. The database would contain a user and location. The application was run locally via Flask. By default flask exposes a port for you to communicate with; the port that was exposed was 5000 by default. The page displayed just some plain text in a header. The MongoDB was then connected to Flask and it used Mongo's URL and default port (27017) to connect and run it from the browser. When run from the command line it was displayed that the JS and Index were connected to the database.

Learning Flask

Before the beginning of the project, we were not familiar with this server and had little experience coding in Python. Hence, learning Flask from the ground upwards was the one of the first steps and challenges we faced, but, being written in python and having lots of online resources available to get started made it that much easier. Flask is used as the basis to implement the server and connect with the database to store the user's information. [27]

Learning MongoDB

Having studied databases such as MySQL and SQL, we knew that connecting the Flask Server to MongoDB using a RESTful api would be no easy task at first. First off, we had to set up an atlas account and get started by learning the basics, what type of database we would working with and would it suit our app. The extensive MongoDB documentation was easy to follow when any issues were encountered. In the end, we felt that it was a great choice. [28] [29]

5.3 Screenshots

Figure 5.2: Logo

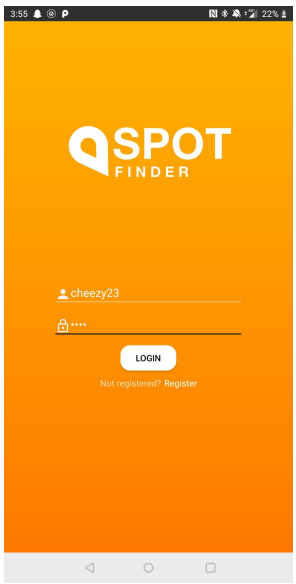


Figure 5.3: Login

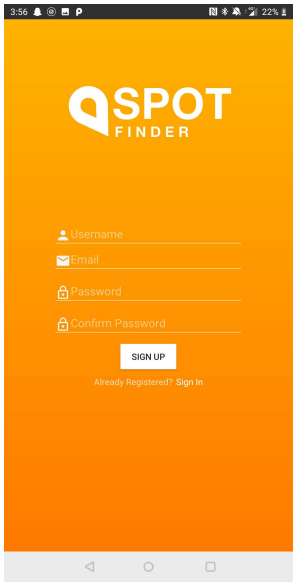


Figure 5.4: Sign Up

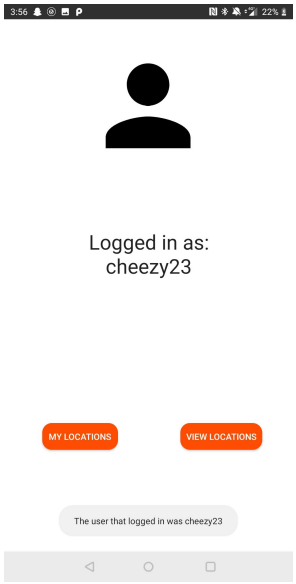


Figure 5.5: Home Page

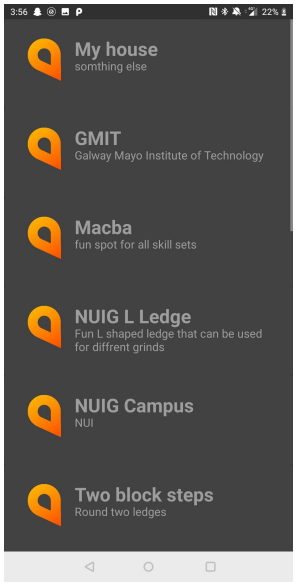


Figure 5.6: My Locations

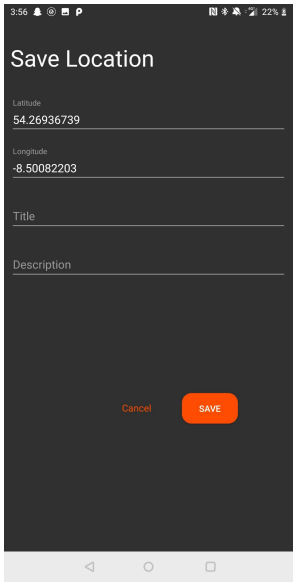


Figure 5.7: Save Locations

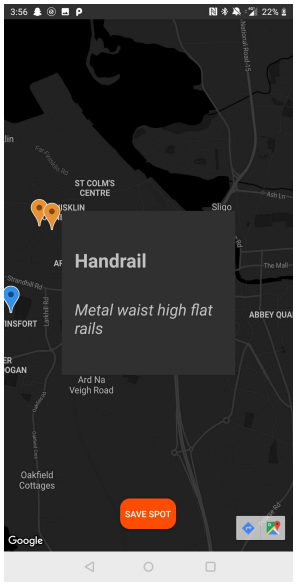


Figure 5.8: Spots Descriptions

Chapter 6

Appendices

6.1 GitHub

Click for GitHub Repo

6.2 Video

Click for Video

Chapter 7

Conclusion

After developing the app and writing the dissertation, there were definitely some parts of the project that were easier than others. Firstly, research played a major factor in what framework, platforms and language was used to make a functional application for users and servers. We outlined how the project would be done throughout this document and what we needed to have achieved as objectives which can be seen in the Introduction. A Flask server was set up which then accessed the MongoDB. Using Google Maps, users' locations were shown once locations services were working. A orange and white colour scheme was added to give a fun and complimentary looking app. The most common colour scheme is of course blue, as it is a sign of peace and order; A login class was created with activities working correctly. This would then ask for a name and password or if not signed up, the option to do so would be implemented. As progression with the app was made, meetings with our supervisors would happen and feedback between both collaborators would be regular. The Meetings we felt were successful and helpful in some stages when developing the app. The background section outlined the technologies used and why we came to choose each of them. This clears up how different technologies did or did not suit our project. Testing during the development stage is seen as it gives an overview of the steps involved. Android development was a good choice in the end when it came to developing the application. There was not many issues using this technology as it is written in Java, a programming language which is universally known and has plenty of resources when it comes to development. The mobile application was easy to deploy on phones for testing which was a feature that was enjoyable. MongoDB and Flask were relatively new technologies used for this project. This was difficult as there were not many ways of finding information on any issues that emerged. Regarding the meetings, they were successful, and the application was changed if there were any issues

and completed in the way that it was intended for the user. This kind of Application has not been completed very successfully before, in this way, from researching and testing other spot finder applications. Time Management was done efficiently in the later stages of the project, whereas towards the beginning, there was a little bit of wasted time on the back-end. But, as we already had a plan on how this project would look, the research and development portion were not taking up as much time as anticipated.

Bibliography

[1] Github issues.

[2] Github readme.

[3] Android studios pros and cons.

[4] Ionic. Ionic widgets.

[5] Priyanka Chaudhary. Ionic framework. *International Research Journal of Engineering and Technology (IRJET)*, 2018.

[6] Android sdk vs react native, 2020.

[7] Dino Trnka. Mobile app development: React native vs native (ios, android), 2018.

[8] Android sdk vs flutter, 2020.

[9] Maximilian Schwarzmüller. React native vs flutter vs ionic vs nativescript vs pwa, 2019.

[10] Rishabh Software. How benefits of android app can help businesses, 2019.

[11] Introduction to flask.

[12] Flask.

[13] Nuruldelmia Idris, Cik Feresa Mohd Foozy, and Palaniappan Shamala. A generic review of web technology: Django and flask. *International Journal of Engineering Information Computing and Application*, 1(1), 2019.

[14] Python flask vs php, 2020.

[15] Mongodb vs sql server, 2019.

[16] Matan Sarig. Cassandra vs mongodb in 2018, (2019).

[17] Microsoft sql server vs mongodb, 2020.

[18] Guide to postman.

[19] Lars Vogel. Using retrofit 2.x as rest client - tutorial. *Vogella*, 2012,2019.

[20] Goran Kovač. Rest api on android made simple or: How i learned to stop worrying and love the rxjava. *AndroidPub*, 2018.

[21] Kent Beck. Test-driven development by example. *Kent Beck, Three Rivers Institute*, 2003.

[22] Barry W. Boehm. "software engineering economics". *pauldee.org*, 1983.

[23] L Williams EM Maximilien. "assessing test-driven development at ibm". *25th International Conference on Software Engineering*, 2003.

[24] Steve Fox. What is behaviour-driven development?, 2016.

[25] Development That Pays. Test driven development vs behaviour driven development cheat sheet.

[26] Agile Alliance. Acceptance test driven development (atdd). (2017).

[27] Pallets. Flask's documentation, 2010.

[28] MongoDB Inc. Introduction to mongodb, 2008.

[29] MongoDB Inc. Install mongodb community edition on macos, 2008.