Department of Informatics

CO7201 MSc Individual Project

BIRDING WITH GOOGLE MAPS

-AARZIN TODIWALA (at485)

FINAL REPORT

SUPERVISOR: Dr Fer-Jan De Vries

SECOND MARKER: Prof Thomas Erlebach

Date: 06/09/2019

Word Count: 10041

DECLARATION

All sentences or passages quoted in this report, or computer code of any form whatsoever used and/or submitted at any stages, which are taken from other people's work have been specifically acknowledged by a clear citation of the source, specifying author, work, date and page(s). Any part of my own written work, or software coding, which is substantially based upon other people's work, is duly accompanied by clear citation of the source, specifying author, work, date and page(s). I understand that failure to do this amount to plagiarism and will be considered grounds for failure in this module and the degree examination as a whole.

Name: Aarzin Todiwala

Date: 06-09-2019

Abstract

The aim of the project was to design and implement a database that would allow a group of birdwatchers like the Leicestershire and Rutland Ornithological Society, to put data of their bird observations in the application in any smartphone. The application will allow a simple and quick way of inputting of data using Google Maps to enter location data. The application should provide an easy interface to query the database and provide tools for analyzing and displaying the data using Google Maps with perfect accuracy. The mechanism of the dynamic map is that the accuracy of the representation of the data will only depend on the zooming level. Here in this android application, log in, registration, information of the birds, putting the location of the sighted bird and viewing the data of the UK sighted birds are the main activities. There are three databases in this application in which one is for user login, second is for information of the birds and third is for the registration of the sighted bird. Here, the data is stored in Android Studio while when an app is installed on the smartphone the data will be kept in the phone as in temporary data. But after putting the data in server database the data will be automatically be stored in server database.

Table of Contents:

Table of Figures:

1 Introduction

Tracking little flying animals as they over continental-scale distances could be a tough provision and engineering challenge. Though no trailing system works well with all species, enhancements to ancient technologies, like satellite trailing, together with innovations associated with international positioning systems, cellular networks, star geolocation, radar, and knowledge technology are up our understanding of once and wherever birds go throughout their annual cycles and informing varied scientific disciplines, as well as organic process biology, population ecology, and international modification.

A new tool permits people to examine this knowledge in the time period. With Bird chase with google maps application, people will verify weekly google maps, that show wherever birds are sharp-sighted in a time period. The maps are the results of an associate automatic knowledge-flow system that collects data from the user through our mechanical man application, resulting in a colossal quantity of knowledge that lets users see wherever birds are at any given moment. Uniting knowledge from the portals could be a nice step for conservation, as antecedently the info gathered by the portals are terribly scattered. With this info, each scientist and people will gain a larger understanding of bird distribution and movements. This, in turn, contributes to a larger understanding of conservation management problems like wherever to position (or where to not place) wind farms.

1.1 Overview

To design and implement an application, with the information of the birds and the exact location, that would allow a group of birdwatchers like in the Leicestershire and Rutland Ornithological Society, to put data of their bird observations in the web application in any desktop or laptop or smartphone and then they can even search for other birds around them, if any other user had entered that. The application will allow a simple and quick way of inputting of data using Google Maps to enter location data, which will create a database and at last it will be displayed on the map.

1.2 Objective

The application should provide an easy interface to query the database and provide tools for analyzing and displaying the data using Google Maps with the proper accuracy. The mechanism of the dynamic map is that the accuracy of the representation of the data will only depend on the zooming level.

1.3 Functional and Non-Functional Requirements

Essential

- A system with JAVA SDK and IDE such as Android Studio.
- Getting Familiar with Google API.
- Creating a user-friendly Android Application.
- A smartphone which can give an accurate location for the user.
- Database which provides information about the located birds.

Recommended

- Accuracy of locating birds in around the area should be accurate as per the location is given.

- Smartphone, Laptop and Desktop.
- Yearly and monthly monitoring of the birds on google map.

Optional

- As per the season, the appearance of birds in a certain area can be monitored.
- Extra information about the birds.
- Locating migrating birds.
- Allow the user to rate the application and overall feedback.

1.4 Why Kotlin?

Java is Android's most widely used language, but that doesn't mean it's always the best choice. Java is old, verbose, mistaken, and slow to upgrade. Kotlin is a valuable option. Open JDK developers are beginning to bridge the gap with Java 8, but not all Java 8 characteristics are being used by Android. Developers are still stuck in the old Java 7 and 6 worlds, and in the foreseeable future, this will not improve much. That's where Kotlin comes in: With the Android software Engineering, this comparatively fresh open-source language, based on the Java Virtual Machine (JVM), gains traction. There are other JVM languages you can attempt to use on Android, but Kotlin provides Android Studio integration, Google's main Android IDE, which has no language, other than Java. Here's why now is the time for your Android development projects to begin using this contemporary, advanced, pragmatic language.

- What's wrong with Java?

You likely have been using Java for years (maybe for centuries), so you know it very well. You understand the language from corner to corner, as well as several undocumented stuff that has only been experienced by veterans with years of experience.So when a new language comes into town and someone tells you to switch to it, you're likely to be sceptical. I was too. But I have also used Java for a long time, and have developed a love-hate relationship with it.

At first, I wasn't interested in turning away from Java, but I changed my mind when I began looking at Kolin and noticed the larger image. Here are some of the reasons for this:

- Java is old... very old

Java was, back in its heyday, one of the most accessible languages. But today, as well as the Java that I use on Android, there is no support for lambdas, method references, streams, try-with-resources (minSdk 19). I still need to use the old Java 6/7 worlds javax.time APIs.

There are some methods for third parties to backport some of these characteristics using instruments like RetroLambda, Streams backport, and ThreeTenABP, but that's a problem.

Android Nougat also made a bold attempt to use the Jack compiler to support some Java 8 features, but most of them can only be used if you target minSdkVersion 24 — something you shouldn't do, given how slow Android version updates have become.

Check out the platform version distribution chart here.

- Java is error prone

One of the biggest flaws in Java is the way it handles “null,” leading to the dreaded NulPointerException (NPE), popularly known as The Billion Dollar Mistake) [1].

I call it my billion-dollar mistake. It was the invention of the null reference in 1965…This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years. —Sir Charles Antony Richard Hoare

Today, the NPE is one of the most common reasons for crashes in Android apps. In fact, it's almost impossible to have an app in production without a single NullPointerException (If you do, please tell me about it, I would be interested to hear how you avoided it) [1].

And "nullability" is an even bigger problem for Android. Null is a very efficient and simple way of representing the absence of a value, and Android uses it in its framework and APIs. It's not good that Java makes it more difficult for developers to handle them [1].

A third problem has to do with the fact that programmers often ignore the way Java implements non-static inner classes and anonymous inner classes, which always keep an implicit reference to the outer class. In so doing, developers end up making their apps susceptible to memory leaks [1].

There's a whole wiki on Java's design flaws that I won't go into here, but I'll end on what might be the biggest gripe of all about Java:

- Verbosity and ceremony

Developers love clean, concise code. Less code takes less time to write, less time to read, and is less susceptible to bugs. But with Java, you must write a lot of code to get even the simplest things done. You've probably already experienced this if you're an Android developer [1].

There's considerable “ceremony” involved in Java APIs, and Android makes that even worse by forcing developers to go through many steps, in a specific order, to get things done, such as accessing a database, handling fragment transactions, and so on [1].

If you could simplify a lot of these processes, it would improve your experience and productivity as an Android developer considerably [1].

- Kotlin to the rescue

Java isn't the only language you can use to build Android apps. The most strongly supported JVM language in the Android ecosystem—aside from Java—is Kotlin, an open-source, statically-typed language developed by JetBrains [1].

JetBrains created one of the most popular IDEs, IntelliJ IDEA, as well as Android Studio, which Google crowned as the standard IDE for Android development. It understood the pain developers face in day-to-day development workflow, and with Kotlin it has attempted to address those. JetBrains uses Kotlin in production to develop its own products, so it is unlikely that the language will suddenly be abandoned [1].

Kotlin takes a pragmatic approach by not including features such as having its own build system or package manager because open source tools such as Gradle and Maven already handle this well. Having its own build system would have broken projects that already use Gradle and Maven [1].

Another pragmatic approach for Kotlin was to not re-implement the entire Java collections framework. That would have been easy, but the creators also wanted Kotlin to be compatible with the JDK collection interfaces without breaking any existing project implementations [1].

One other huge benefit of Kotlin is that most of its language design decisions focused on maintaining backward compatibility with many Java and Android projects. For example, Kotlin still supports Java 6 bytecode because more than half Android devices still run on it [1].

- Kotlin is 100% interoperable with Java

This is the first thing I loved about Kotlin. You can call Java code from Kotlin and vice-versa seamlessly. Both Kotlin and Java generate the same bytecode, so there's no fear that you're shipping something completely different with Kotlin [1].

That means you can start using Kotlin in your existing project, with all your old Java code, right away. Start by writing some simple and small parts of your app in Kotlin as you start getting familiar with its constructs and syntax (which, by the way, is super-simple) [1].

I started using Kotlin for small parts of a large project, including a few UI components and simple business logic. Only four-to-five per cent of the entire codebase was written in Kotlin; the rest is still in Java (which I plan to convert eventually) [1].

This mix of Java and Kotlin code is working well in my project. Its interoperability is truly a blessing [1].

1.5 Advantages of Kotlin

- Lambda expressions + Inline functions = performant custom control structures

  Lambda expressions and anonymous functions are 'function literals', i.e. functions that are not declared but passed immediately as an expression. [2]

  Using higher-order functions imposes certain runtime penalties: each function is an object, and it captures a closure, i.e. those variables that are accessed in the body of the

function. Memory allocations (both for function objects and classes) and virtual calls introduce runtime overhead [3].

But it appears that in many cases this kind of overhead can be eliminated by inlining the lambda expressions. The functions shown below are good examples of this situation. I.e., the lock() function could be easily inlined at call-sites. Consider the following case [3]:

```
lock(l) { foo() }
```

Instead of creating a function object for the parameter and generating a call, the compiler could emit the following code [3]:

```
l.lock()
try {
    foo()
}
finally {
    l.unlock()
}
```

- Extension functions

  Kotlin provides the ability to extend a class with new functionality without having to inherit from the class or use design patterns such as Decorator. This is done via special declarations called *extensions*. [2]

- Null-safety

  Kotlin's type system is aimed at eliminating the danger of null references from code. [2]

  To allow nulls, we can declare a variable as a nullable string, written String? [4]:

```
var b: String? = "abc"
b = null // ok
print(b)
```

- Smart casts

  In many cases, one does not need to use explicit cast operators in Kotlin, because the compiler tracks the is-checks and explicit casts for immutable values and inserts (safe) casts automatically when needed [5]:

```
fun demo(x: Any) {
```

```
    if (x is String) {
        print(x.length) // x is automatically cast to String
    }
}
```

The compiler is smart enough to know a cast to be safe if a negative check leads to a return [5]:

```
if (x !is String) return
print(x.length) // x is automatically cast to String
```

- String templates

  Strings are represented by the type String. Strings are immutable. Elements of a string are characters that can be accessed by the indexing operation: s[i]. [2]

- Properties

  Properties in Kotlin classes can be declared either as mutable using the var keyword or as read-only using the val keyword. [2]

- Primary constructors

  The class declaration consists of the class name, the class header (specifying its type parameters, the primary constructor etc.) and the class body, surrounded by curly braces. Both the header and the body are optional; if the class has no body, curly braces can be omitted. [2]

- First-class delegation

  The Delegation pattern has proven to be a good alternative to implementation inheritance, and Kotlin supports it natively requiring zero boilerplate code. A class Derived can implement an interface Base by delegating all of its public members to a specified object. [2]

- Singletons

  If a supertype has a constructor, appropriate constructor parameters must be passed to it. Many supertypes may be specified as a comma-separated list after the colon. [2]

- Declaration-site variance & Type projections

  In general, to create an instance of such a class, we need to provide the type arguments [6]:

  ```
  val box: Box<Int> = Box<Int>(1)
  ```

  But if the parameters may be inferred, e.g. from the constructor arguments or by some other means, one is allowed to omit the type arguments [6]:

```
val box = Box(1) // 1 has type Int, so the compiler figures out that we are talking about Box<Int>
```

- Range expressions

  Kotlin lets you easily create ranges of values using the rangeTo() function from the kotlin.ranges package and its operator form ... Usually, rangeTo() is complemented by in or !in functions. [2]

- Operator overloading

  Kotlin allows us to provide implementations for a predefined set of operators on our types. These operators have fixed symbolic representation (like + or *) and fixed precedence. [2]

- Companion objects

  If you need to write a function that can be called without having a class instance but needs access to the internals of a class (for example, a factory method), you can write it as a member of an object declaration inside that class. [2]

- Data classes

  We frequently create classes whose main purpose is to hold data. In such a class some standard functionality and utility functions are often mechanically derivable from the data. [2]

- Coroutines

  Asynchronous or non-blocking programming is a new reality. Whether we're creating server-side, desktop or mobile applications, it's important that we provide an experience that is not only fluid from the user's perspective but scalable when needed. [2]

2 Background Material

- **https://www.bto.org/**

  They harness the skills and passion of birdwatchers to advance our understanding of ornithology and produce impartial science - communicated clearly for the benefit of birds and people [7].

- **http://www.lros.org.uk/birdrecording.htm**

  This Society welcomes all records of birds in Leicestershire and Rutland, including breeding records, counts, first and last dates of migrants, visible migration records, interesting behavioural notes and occurrences of unusual species. By sending in records of your sightings, you are taking the first vital step towards helping to protect birds. Accurate information is crucial to conservationists for drawing up strategies on how to conserve birds and their environment. LROS records and surveys help to identify key changes in population, preferred sites or habitats [8].

- **https://waarneming.nl/**

  Waarneming.nl is the largest nature platform in the Netherlands and part of Stichting Natuurinformatie. Waarneming.nl wants to enable everyone to save and share nature observations via the internet, in order to capture the natural wealth of the Netherlands and the world for now and for the future [9].

- **https://observation.org/info.php**

  Observation International focuses specifically on observers who attach importance to registering their observations in a politically independent manner. Observation International is a nonprofit organization without a lucrative purpose.

  The main objective of International Observation is the optimum facilitation of observers in order to make their nature experience even more valuable. As a foundation, Observation International wishes to involve as many local communities as possible in its activities. Doing so, we respect the rights and achievements of these communities, observers and researchers. [10]

- **"The Birdlife of Britain"- Book by Peter Hayman and Philip Burton** [11]**.**

  Its a book on identifying and understanding the birds of Britain and Europe. It also gives an introduction to the bird watching which tells us about the techniques of bird watching and recording what we see and develop interest. It tells us about the structure of the bird and also about the migrating birds which come in different season as per the climate change or the seasonal change. It also tells the difference between the male bird and the female bird and also the young birds after they are born.

- **Google Maps API**

  The Maps API returns helpful data about places and locations. It is called by JavaScript. It does two major things:

- It can cause maps to appear for the user.
- It can return data about a latitude/longitude location or return data about an address.

The Maps JavaScript API lets you customize maps with your own content and imagery for display on web pages and mobile devices. The Maps JavaScript API features four basic map types (roadmap, satellite, hybrid, and terrain) which you can modify using layers and styles, controls and events, and various services and libraries.

- **Google Maps**

  - The Google Maps API lets us request maps from Google and displays them. Example.

  - It also lets us request an address from Google (we give them a physical location (latitude and longitude), they turn it into an address); We can do the reverse, by giving Google an address and requesting a physical location (latitude and longitude). Try searching from the homepage to see this in action.

  - We can also get information about businesses, landmarks, restaurants, etc. from the Google Maps "Places" library. This info includes when they open/close each day, for example.

  - Google Maps helps autocomplete our users' typing, in the search box. This makes it easier and more intuitive to type a real location that will have good results.

- **Basic Knowledge About Android Kotlin**

  The Kotlin Standard Library provides a comprehensive set of tools for managing collections – groups of a variable number of items (possibly zero) that share significance to the problem being solved and are operated upon commonly [12].

  Collections are a common concept for most programming languages, so if you're familiar with, for example, Java or Python collections, you can skip this introduction and proceed to the detailed sections [12].

  A collection usually contains a number of objects (this number may also be zero) of the same type. Objects in a collection are called elements or items. For example, all the students in a department form a collection that can be used to calculate their average age. The following collection types are relevant for Kotlin [12]:

  - The list is an ordered collection with access to elements by indices – integer numbers that reflect their position. Elements can occur more than once in a list. An example of a list is a sentence: it's a group of words, their order is important, and they can repeat.

- Set is a collection of unique elements. It reflects the mathematical abstraction of the set: a group of objects without repetitions. Generally, the order of set elements has no significance. For example, an alphabet is a set of letters.
- Map (or dictionary) is a set of key-value pairs. Keys are unique, and each of them maps to exactly one value. The values can be duplicates. Maps are useful for storing logical connections between objects, for example, an employee's ID and their position.

Kotlin lets you manipulate collections independently of the exact type of objects stored in them. In other words, you add a String to a list of Strings the same way as you would do with Ints or a user-defined class. So, the Kotlin Standard Library offers generic interfaces, classes, and functions for creating, populating, and managing collections of any type. The collection interfaces and related functions are located in the kotlin.collections package [12].

- **"Android Programming with Kotlin for Beginners" Book by John Horton** [13]**.**

  Android is the world's most common mobile operating system, and Google has proclaimed Kotlin to be a first-class programming language for Android apps. With the imminent arrival of Android's most expected update, Android 10 (Q), this book will start building applications that are consistent with Android's recent version.

  It adopts a project-style strategy, whereby constructing three real-world applications and more than a dozen mini-apps we concentrate on learning the fundamentals of Android app growth and Kotlin's essentials. The book starts by providing you a powerful understanding of how Kotlin and Android work together before gradually moving on to explore the multiple Android APIs to easily build amazing Android apps. Using various layouts, you will learn how to create your applications more presentable. You will immerse yourself in Kotlin programming ideas like variables, features, data structures, object-oriented code, and how to link your Kotlin code to the UI. You'll learn to add multilingual text so that millions of prospective customers can access your app. You will learn and implement animation, graphics, and sound effects in your Android app.

- **"Android studio 3.3 Development Essentials" book by eBookFrenzy [14].**

  Starting with the basics, this book outlines the measures needed to set up an Android environment for growth and testing. An overview of Android Studio includes fields such as windows for tools, code editor and tool for the layout editor. An introduction to Android's architecture is accompanied by a thorough look at designing Android apps and user interfaces using the Android Studio environment. Chapters are also included covering the Android Architecture Components including view models, lifecycle management, Room database access, app navigation, live data and data binding.

  There are also more sophisticated subjects like intents, such as touch screen processing, gesture recognition, camera access, and video and audio playback and recording. This book edition also includes printing, transitions and file storage based on the cloud.

Also covering the ideas of material design is the use of floating action buttons, snack bars, tabbed interfaces, card views, navigation drawers and collapsing toolbars.

The book also involves particular subjects such as applying maps using the Google Maps Android API and submitting applications to the Google Play Developer Console as well as covering overall Android development methods.

3 System Planning

3.1 Project Development Approach

Each project needs to be developed with the software model which makes the project with high quality, reliable and cost-effective.

In our project, we are using Iterative Model.

Iterative Model :

- The iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented. At each iteration, design modifications are made and new functional capabilities are added. The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental) [15].
- Following is the representation of Iterative and Incremental model [15]:

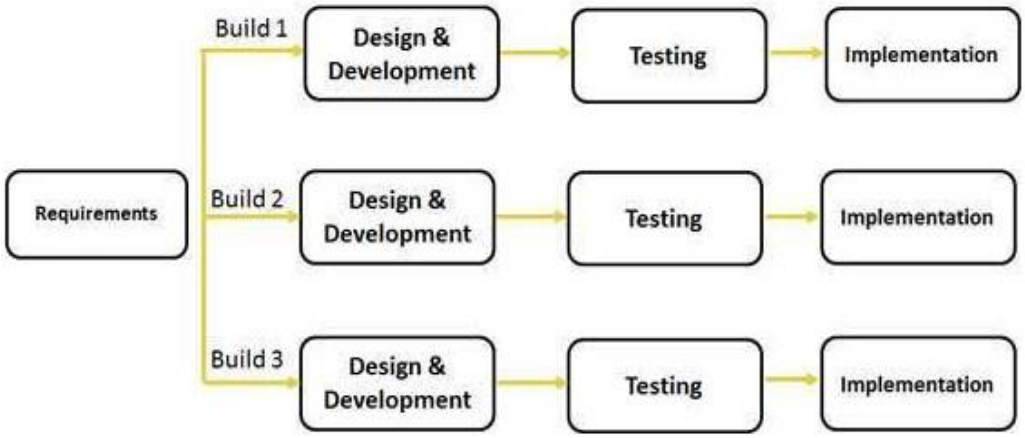


Figure 3-1: Phases of iterative model

- During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues until the complete system is ready as per the requirement. The key to successful use of an iterative software development lifecycle is rigorous validation of requirements, and verification & testing of each version of the software against those requirements within each cycle of the model. As the software evolves through successive cycles, tests must be repeated and extended to verify each version of the software [15].

- In our system, we will implement a small set of software requirements and then iteratively enhance the application versions until the complete system implemented. So, we use iterative model in our project [15].

- Advantages of the iterative model [15]:

  The advantage of this model is that there is a working model of the system at a very early stage of development which makes it easier to find functional or design flaws. Finding issues at an early stage of development enables to take corrective measures in a limited budget.

  - Some working functionality can be developed quickly and early in the life cycle.
  - Results are obtained early and periodically.
  - Parallel development can be planned.
  - Progress can be measured.
  - Less costly to change the scope or requirements.
  - Testing and debugging during smaller iteration is easy.
  - Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
  - Easier to manage risk - High risk part is done first.
  - With every increment, the operational product is delivered.
  - Issues, challenges & risks identified from each increment can be applied to the next increment.
  - It supports changing requirements.
  - Risk analysis is better.

3.2 System Modules

- Login module

  The customer enters their registered user id and password. If verified, they may start saving the bird's location i.e. getting the location of the bird. Otherwise, they will have to re-enter the user id and password.

- Registration module

  New customers first must register themselves with the Application. They have to enter basic details like name, email, password, birthdate, address and postcode.

- Information module

  After a successful login, the user can start exploring the information of the bird which contains the basic knowledge about the database.

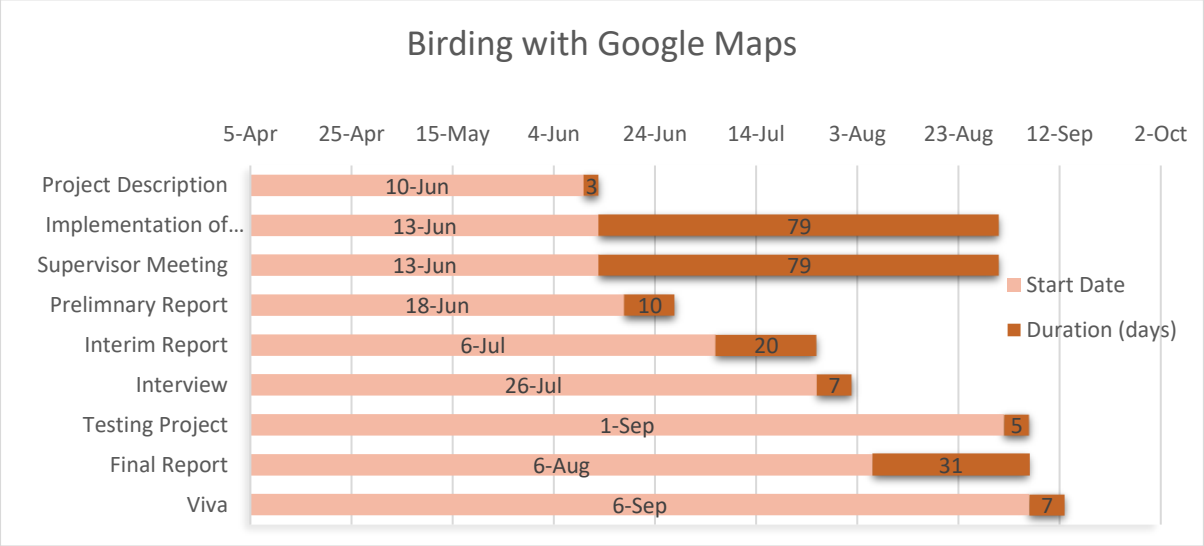
- Recording Bird module

  If user saw a bird and want to register the birds' location into the application then, user can choose the location, then the name of the bird, for them whether it is a rare species or unknown or known, at last whether they want to upload an image and finally submission of that page.

- Display on map module

  The data which will be entered by the users will be stored in a database and will be displayed in this module, after searching for the proper bird user can see that where the birds are been located.

3.3 Timeline Chart



4 Contribution of the Project

4.1 Hardware Requirements

- Processor (CPU) with 2 gigahertz (GHz) frequency or above.
- A minimum of 2 GB of RAM.
- A minimum of 20 GB of available space on the hard disk.
- Internet Connection Broadband (high-speed) Internet connection with a speed of 4 Mbps or higher.
- Keyboard and a Microsoft Mouse or some other compatible pointing device.

4.2 Software Requirement

- Windows 10 Operating System.
- Android Studio Version 3.5
- Android Platform Version: API 29: Android 10.0

4.3 Implementation

Kotlin Code – Login (Validation)

```
fun validation() {
    val pattern = Pattern.compile("^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-
z0-9]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$")
    val matcher = pattern.matcher(edit_email_login.text.toString().trim())
    if (!TextUtils.isEmpty(edit_email_login.text.toString()) &&
matcher.matches()) {
        if (!TextUtils.isEmpty(edit_password_login.text.toString())) {
            if (CSup.iC(this)) {
                button_login.visibility = View.GONE
                login_progress.visibility = View.VISIBLE
                cnf = CSup.gN().create(CNF::class.java)
                var success = cnf.doLogin(edit_email_login.text.toString(),
edit_password_login.text.toString())
                //success.enqueue(this@LoginScreen);
                val status_login =
allTables.login_verification(edit_email_login.text.toString(),
edit_password_login.text.toString())
                if (status_login) {
                    val SharedPreferences = getSharedPreferences("MainPref",
0);
                    val edit = SharedPreferences.edit();
                    val min = 20000
                    val max = 80000
                    val random = Random().nextInt(max - min + 1) + min
                    edit.putBoolean("login", true);
                    edit.putString("username", "DFBT" + random);
                    edit.apply()
                    edit.commit()
                    startActivity(Intent(this@LoginScreen,
HomeScreen::class.java))

                } else {
                    button_login.visibility = View.VISIBLE
                    login_progress.visibility = View.GONE
                    Toast.makeText(this, "Something went wrong please try
again later!!", Toast.LENGTH_LONG).show()

                }
```

```
            } else {
                Toast.makeText(this, "No Internet connection!!",
Toast.LENGTH_LONG).show()

            }

        } else {
            edit_password_login.error = "Invalid User Email"
        }
    } else {
        edit_email_login.error = "Invalid User Email"
    }
}
```

Description

Login Page has two types of validation such as email validation and password validation. This code explains that the email should contain alphabets or numbers or special characters with accompanying @ with some alphabet a dot and again an alphabet (eg., abc123@g.co), So here this is first checked. Then it has to check whether the email exists in the database or else it has to ask user to register. Function validation is used for validation of login page.

XML Code:

```
<LinearLayout
    android:id="@+id/login_li"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:orientation="vertical">

    <android.support.design.widget.TextInputLayout
        android:id="@+id/emailTextInputLayout"
        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email">

        <android.support.design.widget.TextInputEditText
            android:id="@+id/edit_email_login"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textWebEmailAddress" />

    </android.support.design.widget.TextInputLayout>


    <android.support.design.widget.TextInputLayout
        android:id="@+id/passwordTextInputLayout"
        style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="@dimen/_16dp"
        android:hint="Password">

        <android.support.design.widget.TextInputEditText
            android:id="@+id/edit_password_login"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPassword" />
```

```
    </android.support.design.widget.TextInputLayout>
```

Description:

To display the text bar for Email and to display text bar for Password this XML File is used.

Kotlin Code – Registration (Validation)

```
   fun validation() {


        val pattern = Pattern.compile("^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-
Za-z0-9]+(\\.[A-Za-z0-9]+)*(\\.[A-Za-z]{2,})$")

        if (!TextUtils.isEmpty(edit_name.text.toString())) {

            val matcher = pattern.matcher(edit_email.text.toString().trim())
            if (!TextUtils.isEmpty(edit_email.text.toString()) &&
matcher.matches()) {
                if (!TextUtils.isEmpty(edit_password.text.toString())) {
                    if
(edit_password.text.toString().equals(edit_password_c.text.toString())) {
                        if
(!TextUtils.isEmpty(edit_birthdate.text.toString())) {
                            if
(!TextUtils.isEmpty(edit_address.text.toString())) {
                                if
(!TextUtils.isEmpty(edit_postcode.text.toString())) {

                                    if (iC(this)) {
                                        register_button.visibility = View.GONE
                                        registration_progress.visibility =
View.VISIBLE
                                        cnf =
CSup.gN().create(CNF::class.java)
                                        var success = cnf.doRegister(
                                                edit_name.text.toString(),
                                                edit_email.text.toString(),

edit_password_c.text.toString(),
                                                edit_address.text.toString(),
                                                edit_postcode.text.toString(),

edit_birthdate.text.toString())

                                        val status =
allTables.UserInsert(edit_name.text.toString()
                                                , edit_email.text.toString(),
edit_password_c.text.toString(),
                                                edit_address.text.toString(),
edit_postcode.text.toString(),

edit_birthdate.text.toString())

                                        if (status > 0) {
                                            Toast.makeText(this,
"Congratulations,Registration successfully", Toast.LENGTH_LONG).show()
```

```
startActivity(Intent(this@RegistrationActivity, LoginScreen::class.java))
                                        } else {
                                            register_button.visibility =
View.VISIBLE
                                            registration_progress.visibility =
View.GONE
                                            Toast.makeText(this, "Something
went wrong please try again later!!", Toast.LENGTH_LONG).show()

                                        }
                                        // success.enqueue(this)


                                    } else {
                                        Toast.makeText(this, "No Internet
connection!!", Toast.LENGTH_LONG).show()
                                    }
                                    // API CALLLLLL
                                } else {
                                    edit_address.error = "Invalid postcode"
                                }
                            } else {
                                edit_address.error = "Invalid Address"
                            }
                        } else {
                            edit_birthdate.error = "Invalid Birth Date"
                        }
                    } else {
                        edit_password.error = "both password should be match"
                        edit_password.setText("")
                        edit_password_c.setText("")
                    }
                } else {
                    edit_password.error = "Invalid Password"
                }
            } else {
                edit_email.error = "Invalid User Email"
            }
        } else {
            edit_name.error = "Invalid User Name"
        }
    }
}
```

Description

Registration Page has many types of validation such as email validation, password validation and many more related to it. This code explains that the email should contain alphabets or numbers or special characters with accompanying @ with some alphabet a dot and again an alphabet (eg., abc123@g.co), So here this is first checked. It also checks whether the password and the re-entered password are same. All the fields are required and cannot be kept empty. The code also defines to store the data into database. Thus function validation is used to validate Registration page.

XML Code

```
<android.support.design.widget.TextInputLayout
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Name">

    <android.support.design.widget.TextInputEditText
```

```
        android:id="@+id/edit_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textWebEmailAddress"
        android:singleLine="true" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/_16dp"
    android:hint="Email">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/edit_email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textWebEmailAddress"
        android:singleLine="true" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/_16dp"
    android:hint="Password">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/edit_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:singleLine="true" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/_16dp"
    android:hint="Confirm Password">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/edit_password_c"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:singleLine="true" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/_16dp"
```

```
    android:hint="Birth Date">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/edit_birthdate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:clickable="true"
        android:cursorVisible="false"
        android:inputType="textWebEmailAddress"
        android:singleLine="true" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/_16dp"
    android:hint="Address">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/edit_address"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:inputType="textMultiLine"
        android:singleLine="false" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/_16dp"
    android:hint="Postcode">

    <android.support.design.widget.TextInputEditText
        android:id="@+id/edit_postcode"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text"
        android:singleLine="true" />

</android.support.design.widget.TextInputLayout>
```

Description

To display the text bar for Name, Email, Password, Confirm Password, Birthdate, Address and Postcode, for this XML File is used.

Kotlin Code – HomeScreen

```
private fun onClicks() {
    home_progressbar.visibility = View.GONE
    li_main.visibility = View.VISIBLE
    option1.setOnClickListener(this)
    option2.setOnClickListener(this)
    option3.setOnClickListener(this)
}
```

Description
setOnClickListener is used to direct one page to another, where here there three options.

XML Code

```
<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1">


    <ImageView
        android:id="@+id/option1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:alpha="0.5"
        android:scaleType="fitXY"
        android:src="@drawable/home1" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_marginLeft="22dp"
        android:layout_marginRight="22dp"
        android:text="WANT TO EXPLORE ABOUT BIRDS?"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary"
        android:textSize="18dp"
        android:textStyle="bold" />


</RelativeLayout>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1">


    <ImageView
        android:id="@+id/option2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:alpha="0.5"
        android:scaleType="fitXY"
        android:src="@drawable/home2" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_marginLeft="22dp"
        android:layout_marginRight="22dp"
        android:text="SAW A BIRD? \n WANT TO TELL US?"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary"
        android:textSize="18dp"
        android:textStyle="bold" />


</RelativeLayout>

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="1">
```

```
    <ImageView
        android:id="@+id/option3"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:alpha="0.5"
        android:scaleType="fitXY"
        android:src="@drawable/home3" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:layout_marginLeft="22dp"
        android:layout_marginRight="22dp"
        android:text="WHERE YOU CAN LOCATE BIRDS?"
        android:textAlignment="center"
        android:textColor="@color/colorPrimary"
        android:textSize="18dp"
        android:textStyle="bold" />


</RelativeLayout>
```

Description

To display the image and the text above it. This XML File is for displaying data on Home – screen.And after clicking on the text we can get redirected to the next page.

Kotlin Code – Information about Birds

```
private fun insertDummyData() {
    allTables = AllTables(this, null)
    val SharedPreferences = getSharedPreferences("MainPref", 0);
    val userName = SharedPreferences.getString("username", "DFBT20989");

    val bitmapOrg_1 = BitmapFactory.decodeResource(resources,
R.drawable.bird_1)
    val bao_1 = ByteArrayOutputStream()
    bitmapOrg_1.compress(Bitmap.CompressFormat.JPEG, 100, bao_1)
    val ba_1 = bao_1.toByteArray()
    val bit_1 = Base64.encodeToString(ba_1, Base64.DEFAULT)


    val bitmapOrg_2 = BitmapFactory.decodeResource(resources,
R.drawable.bird_2)
    val bao_2 = ByteArrayOutputStream()
    bitmapOrg_2.compress(Bitmap.CompressFormat.JPEG, 100, bao_2)
    val ba_2 = bao_2.toByteArray()
    val bit_2 = Base64.encodeToString(ba_2, Base64.DEFAULT)


    val bitmapOrg_3 = BitmapFactory.decodeResource(resources,
R.drawable.bird_3)
    val bao_3 = ByteArrayOutputStream()
    bitmapOrg_3.compress(Bitmap.CompressFormat.JPEG, 100, bao_3)
    val ba_3 = bao_3.toByteArray()
    val bit_3 = Base64.encodeToString(ba_3, Base64.DEFAULT)
```

```
    val bitmapOrg_4 = BitmapFactory.decodeResource(resources,
R.drawable.bird_4)
    val bao_4 = ByteArrayOutputStream()
    bitmapOrg_4.compress(Bitmap.CompressFormat.JPEG, 100, bao_4)
    val ba_4 = bao_4.toByteArray()
    val bit_4 = Base64.encodeToString(ba_4, Base64.DEFAULT)

    val bitmapOrg_5 = BitmapFactory.decodeResource(resources,
R.drawable.bird_5)
    val bao_5 = ByteArrayOutputStream()
    bitmapOrg_5.compress(Bitmap.CompressFormat.JPEG, 100, bao_5)
    val ba_5 = bao_5.toByteArray()
    val bit_5 = Base64.encodeToString(ba_5, Base64.DEFAULT)


    val success_1 = allTables.birdInsert(getString(R.string.date_1),
            userName,
            "" + getString(R.string.latitude_1),
            "" + getString(R.string.longitude_1),
            getString(R.string.bird_name_1),
            getString(R.string.category_1),
            bit_1,
            getString(R.string.desc_1),
            "")

    val success_2 = allTables.birdInsert(getString(R.string.date_2),
            userName,
            "" + getString(R.string.latitude_2),
            "" + getString(R.string.longitude_2),
            getString(R.string.bird_name_2),
            getString(R.string.category_2),
            bit_2,
            getString(R.string.desc_2),
            "")

    val success_3 = allTables.birdInsert(getString(R.string.date_3),
            userName,
            "" + getString(R.string.latitude_3),
            "" + getString(R.string.longitude_3),
            getString(R.string.bird_name_3),
            getString(R.string.category_3),
            bit_3,
            getString(R.string.desc_3),
            "")

    val success_4 = allTables.birdInsert(getString(R.string.date_4),
            userName,
            "" + getString(R.string.latitude_4),
            "" + getString(R.string.longitude_4),
            getString(R.string.bird_name_4),
            getString(R.string.category_4),
            bit_2,
            getString(R.string.desc_2),
            "")

    val success_5 = allTables.birdInsert(getString(R.string.date_5),
            userName,
            "" + getString(R.string.latitude_5),
            "" + getString(R.string.longitude_5),
            getString(R.string.bird_name_5),
            getString(R.string.category_5),
```

```
            bit_1,
            getString(R.string.desc_1),
            "")

    val success_6 = allTables.birdInsert(getString(R.string.date_6),
            userName,
            "" + getString(R.string.latitude_6),
            "" + getString(R.string.longitude_6),
            getString(R.string.bird_name_6),
            getString(R.string.category_6),
            bit_3,
            getString(R.string.desc_3),
            "")

    val success_7 = allTables.birdInsert(getString(R.string.date_7),
            userName,
            "" + getString(R.string.latitude_7),
            "" + getString(R.string.longitude_7),
            getString(R.string.bird_name_7),
            getString(R.string.category_7),
            bit_4,
            getString(R.string.desc_4),
            "")

    val success_8 = allTables.birdInsert(getString(R.string.date_8),
            userName,
            "" + getString(R.string.latitude_8),
            "" + getString(R.string.longitude_8),
            getString(R.string.bird_name_8),
            getString(R.string.category_8),
            bit_5,
            getString(R.string.desc_5),
            "")


    val success_9 = allTables.birdInsert(getString(R.string.date_9),
            userName,
            "" + getString(R.string.latitude_9),
            "" + getString(R.string.longitude_9),
            getString(R.string.bird_name_9),
            getString(R.string.category_9),
            bit_4,
            getString(R.string.desc_4),
            "")

    val success_10 = allTables.birdInsert(getString(R.string.date_10),
            userName,
            "" + getString(R.string.latitude_10),
            "" + getString(R.string.longitude_10),
            getString(R.string.bird_name_10),
            getString(R.string.category_10),
            bit_5,
            getString(R.string.desc_5),
            "")
    if (success_1 > 0 && success_2 > 0 && success_3 > 0 && success_4 > 0 &&
success_5 > 0 && success_6 > 0 && success_7 > 0) {
        isInsert = true
        onClicks()
    } else {
        isInsert = false
    }
```

```
    val sp = getSharedPreferences("InPref", 0);
    val edit = sp.edit();
    edit.putBoolean("is_insert", isInsert!!);
    edit.apply()
    edit.commit()
}
```

Description

In this Kotlin File, we can observe that the data has been called from the database and been asked to display. Basically, the information of the bird has to be displayed as per the sequence of name. Here the sequence is from Image of the bird, Date, Species and Information of the bird.

XML Code

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_centerVertical="true"
    android:orientation="vertical"
    android:padding="@dimen/_16dp">

    <TextView
        android:id="@+id/bird_name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Reena"
        android:textAllCaps="true"
        android:textColor="@color/colorPrimary"
        android:textSize="16sp"
        android:textStyle="bold" />

    <ImageView
        android:id="@+id/bird_image"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:layout_margin="@dimen/_16dp"
        android:adjustViewBounds="true"
        android:scaleType="fitCenter"
        android:src="@drawable/empty_image" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="@dimen/_6dp"
        android:weightSum="1">

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.3"
            android:text="Date"
            android:textColor="#ff000000"
            android:textSize="14sp" />

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.1"
            android:gravity="center"
```

```
            android:text=":"
            android:textColor="#ff000000"
            android:textSize="14sp" />

        <TextView
            android:id="@+id/date"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="0.6"
            android:text="Reena"
            android:textColor="#ff000000"
            android:textSize="14sp" />
    </LinearLayout>
```

Description

To display the information of the birds, like image, date, species and information of the bird.

Kotlin Code – Enable Location

```
@SuppressLint("RestrictedApi")
fun locationEnable() {
    mFusedLocationClient =
LocationServices.getFusedLocationProviderClient(this);
    mSettingsClient = LocationServices.getSettingsClient(this);

    mLocationCallback = object : LocationCallback() {
        override fun onLocationResult(locationResult: LocationResult?) {
            mCurrentLocation = locationResult!!.getLastLocation();
            updateUI()
        }
    }
    mRequestingLocationUpdates = false;

    mLocationRequest = LocationRequest();
    mLocationRequest.setInterval(UPDATE_INTERVAL_IN_MILLISECONDS);

mLocationRequest.setFastestInterval(FASTEST_UPDATE_INTERVAL_IN_MILLISECONDS);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

    var builder = LocationSettingsRequest.Builder();
    builder.addLocationRequest(mLocationRequest);
    mLocationSettingsRequest = builder.build();
}

fun PermissionCheckLocation() {
    Dexter.withActivity(this@FormActivity)
            .withPermissions(
                    Manifest.permission.ACCESS_FINE_LOCATION,
                    Manifest.permission.ACCESS_COARSE_LOCATION

            )
            .withListener(object : MultiplePermissionsListener {
                override fun onPermissionsChecked(report:
MultiplePermissionsReport) {
                    if (report.areAllPermissionsGranted()) {
                        locationEnable()
                        startLocationUpdates()
```

```
                    }

                    if (report.isAnyPermissionPermanentlyDenied) {
                    }
                }
                override fun onPermissionRationaleShouldBeShown(permissions:
List<PermissionRequest>, token: PermissionToken) {
                    token.continuePermissionRequest()
                }
            })
            .onSameThread()
            .check()
}
```

Description

This code is especially for accessing the location in your smartphone, as we all know in each and every smartphone we have to on location or give permission to the application to access the location.

XML Code

```
<fragment
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="250dp"
    android:layout_below="@+id/rl1"
    android:layout_marginTop="@dimen/_16dp" />


<TextView
    android:id="@+id/map_latlong"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/map"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="@dimen/_16dp"
    android:text="Loading..."
    android:textAlignment="center"
    android:textSize="@dimen/_16dp" />
```

Description

To access Location in the application this XML File is used.

Kotlin Code – Camera Functions

```
fun PermissionCheck(i: Int) {
    Dexter.withActivity(this@FormActivity)
            .withPermissions(
                    Manifest.permission.CAMERA,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE,
                    Manifest.permission.READ_EXTERNAL_STORAGE
            )
            .withListener(object : MultiplePermissionsListener {
                override fun onPermissionsChecked(report:
MultiplePermissionsReport) {
                    if (report.areAllPermissionsGranted()) {
```

```
                        if (i == 0) {
                            openCameraIntent()
                        } else if (i == 1) {
                            openGallery()
                        }
                    }
                    if (report.isAnyPermissionPermanentlyDenied) {
                    }
                }
                override fun onPermissionRationaleShouldBeShown(permissions:
List<PermissionRequest>, token: PermissionToken) {
                    token.continuePermissionRequest()
                }
            })
            .onSameThread()
            .check()
}


private fun openGallery() {
    val i = Intent(
            Intent.ACTION_PICK,
            android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
    startActivityForResult(i, PICK_FROM_GALLERY)
}


override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    if (requestCode == REQUEST_CAPTURE_IMAGE &&
            resultCode == RESULT_OK) {
        if (data != null) {
            val imageBitmap = data?.getExtras()?.get("data") as Bitmap
            convertToBase64(this, imageBitmap).execute()
        } else {
            tD("null data camera")
        }
    } else if (requestCode == PICK_FROM_GALLERY &&
            resultCode == RESULT_OK) {
        if (data != null) {
            val uri = data.data
            if (uri != null) {
                var bitmap: Bitmap;
                bitmap =
MediaStore.Images.Media.getBitmap(this.contentResolver, uri)
                bitmap = Bitmap.createScaledBitmap(bitmap, (bitmap.getWidth()
* 0.8).toInt(), (bitmap.getHeight() * 0.8).toInt(), true)

                convertToBase64(this, bitmap).execute()

            } else {
                tD("null uri")
            }

        } else {
            tD("null data")
        }

    } else if (requestCode == REQUEST_CHECK_SETTINGS) {
```

```
        locationEnable()
    }
}


inner class convertToBase64(val c: Activity, val b: Bitmap) : AsyncTask<Void,
Void, Bitmap>() {

    lateinit var progressDial: ProgressDialog
    override fun onPreExecute() {
        super.onPreExecute()
        progressDial = ProgressDialog(c)
        progressDial.setMessage("Preparing Image...")
        progressDial.show()

    }

    override fun onPostExecute(result: Bitmap) {
        super.onPostExecute(result)
        progressDial.dismiss()
        Image_image.visibility = View.VISIBLE
        Image_image.setImageBitmap(result)
    }

    override fun doInBackground(vararg params: Void?): Bitmap {
        encodeTobase64(b)
        return b

    }
    fun encodeTobase64(image: Bitmap): String {
        val baos = ByteArrayOutputStream()
        image.compress(Bitmap.CompressFormat.JPEG, 100, baos)
        val b = baos.toByteArray()
        baseImage = Base64.encodeToString(b, Base64.DEFAULT)

        return baseImage!!
    }
}

fun tD(m: String) {
    Toast.makeText(this@FormActivity, m, Toast.LENGTH_LONG).show()
}
```

Description

This code is especially for accessing the Camera in your smartphone, as well as the existing images if you want to share, this camera function is not mandatory if the user wants to upload picture hey can or else leave that option. If the image is taken by the smartphone camera or taken from the existing file then it can be displayed on the device.

XML Code

```
<Button
    android:id="@+id/capture_image"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/radio_grp"
    android:layout_marginTop="@dimen/_16dp"
    android:background="@color/colorPrimary"
    android:text="Select Image"
```

```
    android:textColor="@color/white"
    android:textSize="@dimen/_16dp" />

<ImageView
    android:id="@+id/Image_image"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:layout_below="@+id/capture_image"
    android:layout_marginTop="@dimen/_16dp"
    android:visibility="gone" />
```

Description

To access image option in the application this XML File is used, his option is optional in the Form Activity.

Kotlin Code – Bird Search for existing locating data

```
private fun searchData() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        val manager = getSystemService(Context.SEARCH_SERVICE) as
SearchManager
        searchView.setSearchableInfo(manager.getSearchableInfo(componentName))
        searchView.setOnQueryTextListener(object :
SearchView.OnQueryTextListener {
            override fun onQueryTextSubmit(s: String?): Boolean {
                cursor = allTables.getBirdListByKeyword(s!!)
                if (cursor == null) {
                    Toast.makeText(this@BirdSerch, "No records found!",
Toast.LENGTH_LONG).show()
                } else {
                    Toast.makeText(this@BirdSerch,
cursor!!.getCount().toString() + " records found!", Toast.LENGTH_LONG).show()
                }
                customAdapter!!.swapCursor(cursor)

                return false
            }

            override fun onQueryTextChange(s: String?): Boolean {
                if(s!!.length!=0) {
                    lay_map.visibility = View.GONE
                    listView_search.visibility = View.VISIBLE
                    cursor = allTables.getBirdListByKeyword(s!!)
                    if (cursor != null) {
                        customAdapter!!.swapCursor(cursor)
                    }
                }else
                {
                    lay_map.visibility = View.GONE
                    listView_search.visibility = View.GONE
                }
                return false
            }
        })

    }
}
```

```
@SuppressLint("RestrictedApi")
fun locationEnable(lat: Double, lon: Double) {
    mFusedLocationClient =
LocationServices.getFusedLocationProviderClient(this);
    mSettingsClient = LocationServices.getSettingsClient(this);

    mLocationCallback = object : LocationCallback() {
        override fun onLocationResult(locationResult: LocationResult?) {
            mCurrentLocation = locationResult!!.getLastLocation();
            updateUI(lat, lon)
        }
    }
    mRequestingLocationUpdates = false
    mLocationRequest = LocationRequest()
    mLocationRequest.setInterval(UPDATE_INTERVAL_IN_MILLISECONDS);

mLocationRequest.setFastestInterval(FASTEST_UPDATE_INTERVAL_IN_MILLISECONDS);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

    var builder = LocationSettingsRequest.Builder();
    builder.addLocationRequest(mLocationRequest);
    mLocationSettingsRequest = builder.build();
}

fun updateUI(lat: Double, lon: Double) {
    if (mapReady) {
        mMap.clear()
        var latlong = LatLng(lat, lon)
        mMap.mapType = com.google.android.gms.maps.GoogleMap.MAP_TYPE_HYBRID
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latlong))
        mMap.animateCamera(CameraUpdateFactory.zoomTo(7F))
        if (cursor != null && cursor!!.moveToFirst()) {
            do {
                val lat1 =
cursor!!.getString(cursor!!.getColumnIndexOrThrow(AllTables.LATITUDE)).toDoubl
e()
                val lon1 =
cursor!!.getString(cursor!!.getColumnIndexOrThrow(AllTables.LONGITUDE)).toDoub
le()
                var latlong = LatLng(lat1, lon1)
                mMap.addMarker(MarkerOptions().position(latlong).title(""))

            } while (cursor!!.moveToNext())
        }
    } else {
        tD("map is not ready")
    }
}
fun PermissionCheckLocation(lat: Double, lon: Double) {
    Dexter.withActivity(this@BirdSerch)
            .withPermissions(
                    Manifest.permission.ACCESS_FINE_LOCATION,
                    Manifest.permission.ACCESS_COARSE_LOCATION

            )
            .withListener(object : MultiplePermissionsListener {
                override fun onPermissionsChecked(report:
MultiplePermissionsReport) {
                    if (report.areAllPermissionsGranted()) {
                        locationEnable(lat, lon)
```

```
                        startLocationUpdates()
                    }
                    if (report.isAnyPermissionPermanentlyDenied) {
                    }
                }
                override fun onPermissionRationaleShouldBeShown(permissions:
List<PermissionRequest>, token: PermissionToken) {
                    token.continuePermissionRequest()
                }
            })
            .onSameThread()
            .check()
}

override fun onMapReady(googleMap: GoogleMap?) {
    mMap = googleMap!!;
    mapReady = true
}
fun startLocationUpdates() {
    mSettingsClient
            .checkLocationSettings(mLocationSettingsRequest)
            .addOnSuccessListener(this, object :
OnSuccessListener<LocationSettingsResponse> {
                @SuppressLint("MissingPermission")
                override fun onSuccess(p0: LocationSettingsResponse?) {

mFusedLocationClient.requestLocationUpdates(mLocationRequest,
                            mLocationCallback, Looper.myLooper());
                }
            })
            .addOnFailureListener(this, object : OnFailureListener {
                override fun onFailure(p0: Exception) {

                    if (p0 is ResolvableApiException) {
                        try {
                            p0.startResolutionForResult(this@BirdSerch,
                                    REQUEST_CHECK_SETTINGS)
                        } catch (sendEx: IntentSender.SendIntentException) {
                        }
                    }
                }
            })

}
fun tD(m: String) {
    Toast.makeText(this@BirdSerch, m, Toast.LENGTH_LONG).show()
}
inner class someTask(val context: Context) : AsyncTask<Cursor, Cursor,
Cursor>() {
    override fun onPreExecute() {
        super.onPreExecute()
        search_progress.visibility = View.VISIBLE
        searchView.visibility = View.GONE
        allTables = AllTables(context, null)

    }

    override fun doInBackground(vararg params: Cursor?): Cursor {
        cursor = allTables.getBirdCursor()
        runOnUiThread {
            customAdapter = SearchAdapter(context, cursor!!, 0)
```

```
            listView_search.adapter = customAdapter
        }
        return cursor!!
    }
    override fun onPostExecute(cursor: Cursor?) {
        super.onPostExecute(cursor)
        if (cursor != null) {
            search_progress.visibility = View.GONE
            searchView.visibility = View.VISIBLE
            searchData()

            listView_search.onItemClickListener = object :
AdapterView.OnItemClickListener {
                override fun onItemClick(parent: AdapterView<*>?, view: View?,
position: Int, id: Long) {
                    if (listView_search.visibility == View.VISIBLE) {
                        lay_map.visibility = View.VISIBLE
                        lat =
cursor!!.getString(cursor!!.getColumnIndexOrThrow(AllTables.LATITUDE)).toDoubl
e()
                        lon =
cursor!!.getString(cursor!!.getColumnIndexOrThrow(AllTables.LONGITUDE)).toDoub
le()

                        locationEnable(lat, lon)
                        PermissionCheckLocation(lat, lon)
                        var map: SupportMapFragment? =
supportFragmentManager.findFragmentById(R.id.map) as SupportMapFragment
                        map?.getMapAsync(this@BirdSerch)

                        hideSoftKeyBoard(context, view!!)
                    }
                }
            }
        } else {
            Toast.makeText(this@BirdSerch, "Please insert bird data
first..!!", Toast.LENGTH_LONG).show()
        }

    }
}

fun hideSoftKeyBoard(context: Context, view: View) {
    try {
        val imm = context.getSystemService(Context.INPUT_METHOD_SERVICE) as
InputMethodManager
        imm?.hideSoftInputFromWindow(view.windowToken,
InputMethodManager.HIDE_NOT_ALWAYS)
    } catch (e: Exception) {
        // TODO: handle exception
        e.printStackTrace()
    }

}
```

Description

In search view function the data from the existing table will search the name entered by the user. Then as per the selection, the actual data will be displayed which includes the map (which needs latitude and longitude from the table which is already there in the existing table), name of the bird

and date. The smartphone should have given access to the application or else the map won't be visible.

XML Code

```
<SearchView
    android:id="@+id/searchView"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:iconifiedByDefault="false"
    android:queryHint="@string/search_hint"
    android:searchIcon="@drawable/search" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/searchView"
    android:orientation="vertical">

    <LinearLayout
        android:id="@+id/lay_map"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/searchView"
        android:orientation="vertical"
        android:visibility="gone">

        <fragment
            android:id="@+id/map"
            android:name="com.google.android.gms.maps.SupportMapFragment"
            android:layout_width="match_parent"
            android:layout_height="230dp"
            android:layout_below="@+id/rl1"
            android:layout_marginTop="@dimen/_16dp" />
    </LinearLayout>

    <ListView
        android:id="@+id/listView_search"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:divider="@null"
        android:dividerHeight="0dp"
        android:visibility="invisible"
        android:paddingTop="@dimen/_8dp"
        android:paddingBottom="@dimen/_8dp"/>
</LinearLayout>
```

Description

To get search option and after that selecting a bird the view which generates includes, Map, Bird name, species and date.

Kotlin Code – Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.leaf.birdtracking">
```

```
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
/>
    <uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />

    <uses-feature android:name="android.hardware.camera" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/bird_tracking_icon"
        android:label="@string/app_name"
        android:roundIcon="@drawable/bird_tracking_icon"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">


        <uses-library
            android:name="org.apache.http.legacy"
            android:required="false" />

        <activity android:name=".HomeScreen" />
        <activity
            android:name=".OnSup.FormActivity"
            android:parentActivityName=".HomeScreen">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".HomeScreen" />
        </activity>
        <activity android:name=".OnSup.Birds_Detail"
            android:parentActivityName=".HomeScreen">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".HomeScreen" />
        </activity>

        <activity android:name=".OnSup.BirdSerch"
            android:parentActivityName=".HomeScreen">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value=".HomeScreen" />
        </activity>
        <activity
            android:name=".RegistrationActivity"
            android:windowSoftInputMode="adjustResize" />
        <activity
            android:name=".LoginScreen"
            android:windowSoftInputMode="adjustResize" />
        <activity
            android:name=".SplashActivity"

android:theme="@style/Theme.AppCompat.Light.NoActionBar.FullScreen">
            <intent-filter>
```

```
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".MainActivity" />

        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />
    </application>

</manifest>
```

Description

Manifest file includes all access which includes access for location, access for the camera, and access for existing gallery. All the cards (Activities) are connected in Manifest file, from loading page to the Bird searching page, All pages are connected. The Google API key is linked in this file but the actual API key is stored in string file.

Kotlin Code – Database File

```
class AllTables(context: Context, factory: SQLiteDatabase.CursorFactory?)
    : SQLiteOpenHelper(context, DATABASE_NAME, factory, DATABASE_VERSION) {


    companion object {
        private val DATABASE_VERSION = 1
        private val DATABASE_NAME = "Bird_Tracking_Database"
        val TABLE_NAME_USER = "UserInfo"
        val TABLE_NAME_BIRD_INSERT = "InsertBird"

        val KEY_ROWID = "_id"

        val USERID = "user_id"
        val USERNAME = "username"
        val USEREMAIL = "useremail"
        val USEREPASS = "userpassword"
        val USEREADD = "useraddress"
        val USEREPOSTCODE = "userpostcode"
        val USEREBIRTHDATE = "userbithdate"

        val BIRDINSERTID = "bird_insert_id"
        val DATE = "date"
        val L_USERNAME = "l_username"
        val LATITUDE = "lattitude"
        val LONGITUDE = "longitude"
        val BIRDNAME = "Birdname"
        val BIRDCATEGORY = "Birdcategory"
        val IMAGE = "IMAGE"
        val IMAGE_NAME = "image_name"
        val BIRD_DESC = "bird_desc"

    }

    override fun onCreate(db: SQLiteDatabase) {
```

```
        val CREATE_USER_TABLE = ("CREATE TABLE " +
                TABLE_NAME_USER + "("
                + USERID + " INTEGER PRIMARY KEY," +
                USERNAME + " TEXT," +
                USEREMAIL + " TEXT," +
                USEREPASS + " TEXT," +
                USEREADD + " TEXT," +
                USEREPOSTCODE + " TEXT," +
                USEREBIRTHDATE + " TEXT" + ")")


        val CREATE_BIRDINSERT_TABLE = ("CREATE TABLE " +
                TABLE_NAME_BIRD_INSERT + "("
                + BIRDINSERTID + " INTEGER PRIMARY KEY AUTOINCREMENT," +
                DATE + " TEXT," +
                L_USERNAME + " TEXT," +
                LATITUDE + " TEXT," +
                LONGITUDE + " TEXT," +
                BIRDNAME + " TEXT," +
                BIRDCATEGORY + " TEXT," +
                IMAGE + " TEXT," +
                BIRD_DESC + " TEXT," +
                IMAGE_NAME + " TEXT" + ")")

        db.execSQL(CREATE_USER_TABLE)
        db.execSQL(CREATE_BIRDINSERT_TABLE)

    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion:
Int) {
        db!!.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME_USER);
        db!!.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME_BIRD_INSERT);
    }

    fun UserInsert(username: String,
                   useremail: String,
                   userpassword: String,
                   useraddress: String,
                   userpostcode: String,
                   userbithdate: String): Long {

        val values = ContentValues()
        values.put(USERNAME, username)
        values.put(USEREMAIL, useremail)
        values.put(USEREPASS, userpassword)
        values.put(USEREADD, useraddress)
        values.put(USEREPOSTCODE, userpostcode)
        values.put(USEREBIRTHDATE, userbithdate)
        val db = this.writableDatabase
        val i = db.insert(TABLE_NAME_USER, null, values)
        db.close()
        return i

    }


    fun birdInsert(date: String,
                   username: String,
                   latitude: String,
                   longitude: String,
```

```
                    birdname: String,
                    bird_category: String,
                    image: String,
                    bird_desc: String,
                    image_name: String): Long {

        val values = ContentValues()
        values.put(DATE, date)
        values.put(L_USERNAME, username)
        values.put(LATITUDE, latitude)
        values.put(LONGITUDE, longitude)
        values.put(BIRDNAME, birdname)
        values.put(BIRDCATEGORY, bird_category)
        values.put(IMAGE, image)
        values.put(BIRD_DESC, bird_desc)
        values.put(IMAGE_NAME, image_name)
        val db = this.writableDatabase
        val i = db.insert(TABLE_NAME_BIRD_INSERT, null, values)
        db.close()
        return i

    }


    fun login_verification(useremail: String, userpassword: String): Boolean {

        var login = false
        val selectQuery = "SELECT  * FROM " + TABLE_NAME_USER + " WHERE " +
USEREMAIL + " = '" + useremail + "' AND " +
                USEREPASS + " = '" + userpassword + "' ; "

        val db = this.writableDatabase
        val cursor = db.rawQuery(selectQuery, null)

        if (cursor.moveToFirst()) {
            login = true
        } else {
            login = false
        }
        return login


    }


    fun getBirdData(): ArrayList<Bird> {
        val bird_list: ArrayList<Bird> = ArrayList()
        bird_list.clear()
        // Select All Query
        val selectQuery = "SELECT  * FROM $TABLE_NAME_BIRD_INSERT"

        val db = this.writableDatabase
        val cursor = db.rawQuery(selectQuery, null)

        // looping through all rows and adding to list
        if (cursor.moveToFirst()) {
            do {
                val bird = Bird()
                bird.date =
cursor.getString(cursor.getColumnIndexOrThrow(DATE))
                bird.userName =
```

```
cursor.getString(cursor.getColumnIndexOrThrow(L_USERNAME))
                bird.bird_name =
cursor.getString(cursor.getColumnIndexOrThrow(BIRDNAME))
                bird.latitude =
cursor.getString(cursor.getColumnIndexOrThrow(LATITUDE))
                bird.longitude =
cursor.getString(cursor.getColumnIndexOrThrow(LONGITUDE))
                bird.category =
cursor.getString(cursor.getColumnIndexOrThrow(BIRDCATEGORY))
                bird.bird_image =
cursor.getString(cursor.getColumnIndexOrThrow(IMAGE))
                bird.bird_desc =
cursor.getString(cursor.getColumnIndexOrThrow(BIRD_DESC))
                bird_list.add(bird)
            } while (cursor.moveToNext())
        }

        // close db connection
        db.close()
        // return notes list
        return bird_list
    }


    fun getBirdCursor(): Cursor? {
        val db = this.writableDatabase
        val selectQuery = "SELECT  rowid as " +
                KEY_ROWID + "," +
                DATE + "," +
                BIRDNAME + "," +
                LATITUDE + "," +
                LONGITUDE + "," +
                BIRDCATEGORY + "," +
                IMAGE +
                " FROM " + TABLE_NAME_BIRD_INSERT

        val cursor = db.rawQuery(selectQuery, null)

        if (cursor == null) {
            return null;
        } else if (!cursor.moveToFirst()) {
            cursor.close();
            return null;
        }
        // return notes list
        return cursor
    }


    fun getBirdListByKeyword(search: String): Cursor? {
        //Open connection to read only
        val db = this.writableDatabase
        val selectQuery = "SELECT  rowid as " +
                KEY_ROWID + "," +
                DATE + "," +
                BIRDNAME + "," +
                LATITUDE + "," +
                LONGITUDE + "," +
                BIRDCATEGORY + "," +
                IMAGE +
                " FROM " + TABLE_NAME_BIRD_INSERT +
```

```
                " WHERE " + BIRDNAME + "  LIKE  '%" + search + "%' "


        val cursor = db.rawQuery(selectQuery, null)
        // looping through all rows and adding to list

        if (cursor != null) {
            cursor.moveToFirst()
        }
        return cursor
    }

}
```

Description

As we have discussed earlier that there are two databases one is for login and other is for bird recording. For both, the insertion of the data is done from this kotlin file. Also for getting the data this Kotlin file is used, example for login data this file checks the data and verifies the data. For another example the search for the bird in the Records of the Bird, this Kotlin file is used.

Kotlin Code – First Page (Splash Window)

```
  override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)


        Handler().postDelayed(Runnable {

            val SharedPreferences = getSharedPreferences("MainPref", 0);

            if (SharedPreferences.getBoolean("login", false)) {
                startActivity(Intent(this@SplashActivity,
HomeScreen::class.java))
            } else {
                startActivity(Intent(this@SplashActivity,
LoginScreen::class.java))
            }
        }, 3000);
    }
}
```

Description

This kotlin file delays for 3000 seconds and displays the home – screen or else if you are not logged In than it will display the login page. That's the reason it is known as SplashActivity.

XML Code

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:background="#ffffff"
    android:layout_height="match_parent"
    tools:context=".SplashActivity">


    <ImageView
        android:src="@drawable/bird_tracking_icon"
        android:layout_centerInParent="true"
        android:id="@+id/splash_image"
        android:layout_width="200dp"
        android:layout_height="200dp" />


    <TextView
        android:textStyle="bold"
        android:text="@string/app_name"
        android:textSize="32dp"
        android:textColor="@color/colorPrimary"
        android:layout_marginTop="16dp"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/splash_image"
        android:id="@+id/splah_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />


</RelativeLayout>
```

Description

This XML file represents the Image and the Application name.

4.4 System Design

- Use Case Diagram

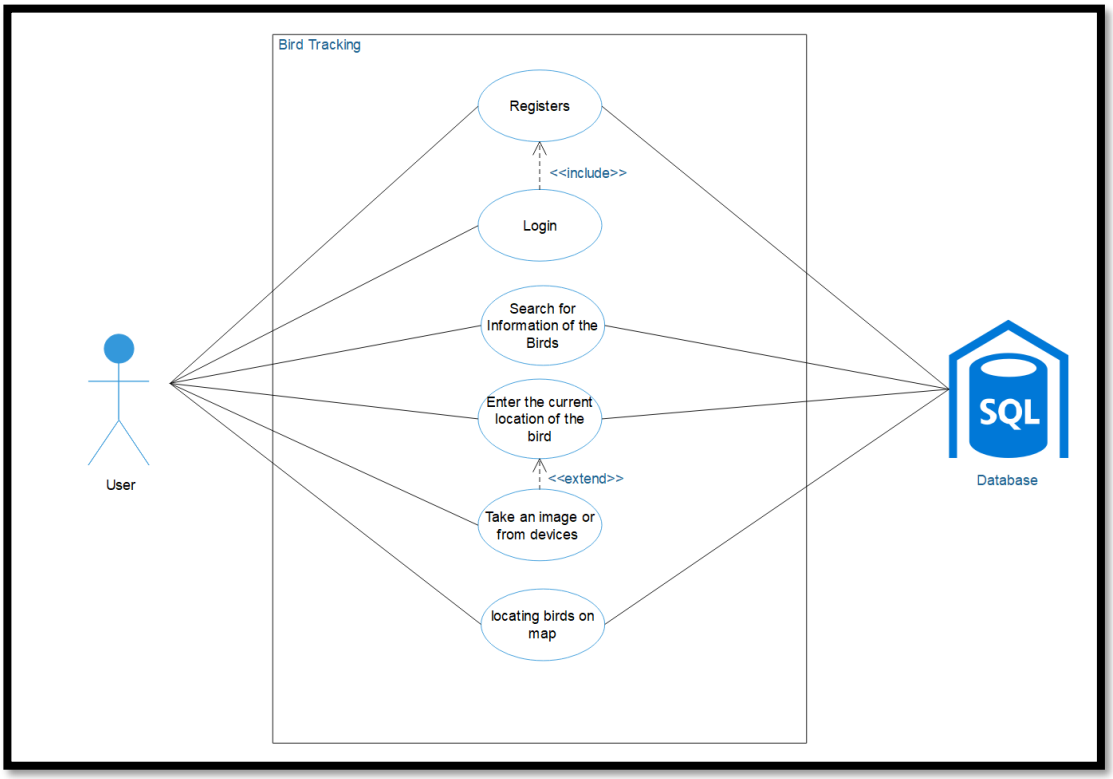


Figure 4-1 : Use-Case Diagram

A UML use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behaviour (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (i.e. use case diagram). A key concept of use case modelling is that it helps us design a system from the end user's perspective. It is an effective technique for communicating system behaviour in the user's terms by specifying all externally visible system behaviour [16].

This Use Case diagram shows that User Registers his/her data then user has to log in as per the registration. After Login he can search for bird and get information about the bird. If the bird is sighted and user wants to record the birds location than enter the location of the bird with that it can also take a picture and upload it but that’s optional. At the end the user can search for bird and see on the map the location of the bird.

- Sequence Diagram

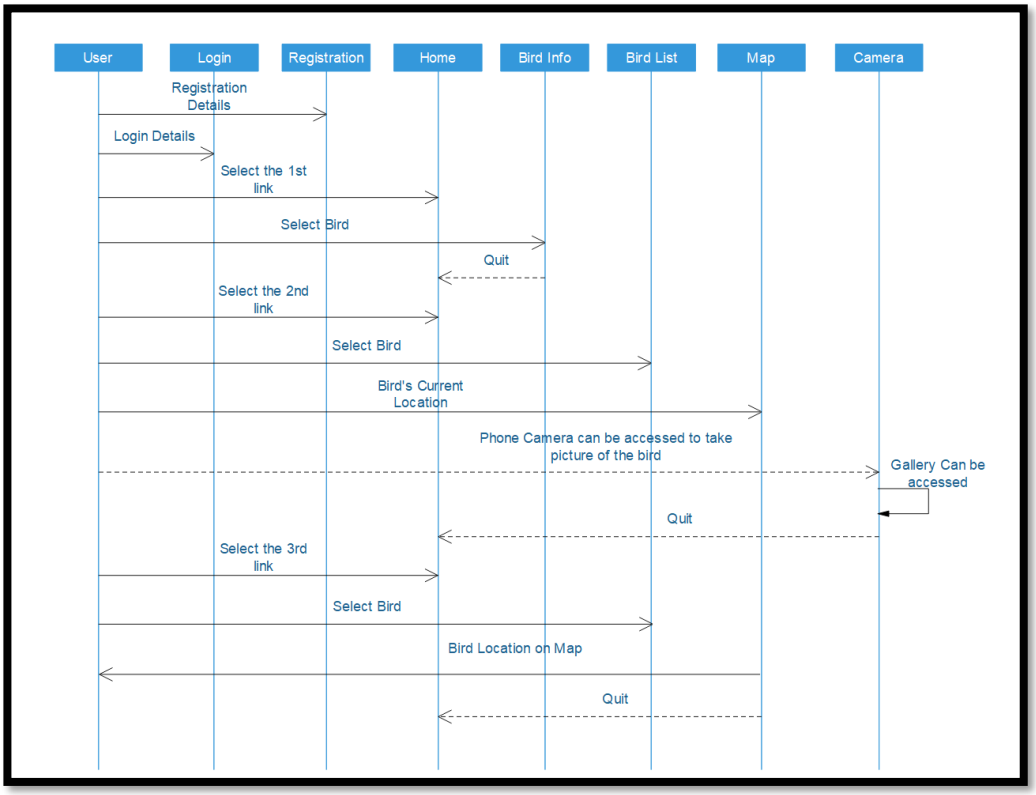


Figure 4-2 : Sequence Diagram

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when [17].

This Sequence diagram shows that User Registers his/her data then user has to log in as per the registration. After Login he can search for bird and get information about the bird. If the bird is sighted, first search for the bird in the list, then the user wants to record the birds location than enter the location of the bird with that it can also take a picture and upload it but that's optional. At the end the user can search for bird and see on the map the location of the bird.

- Class Diagram

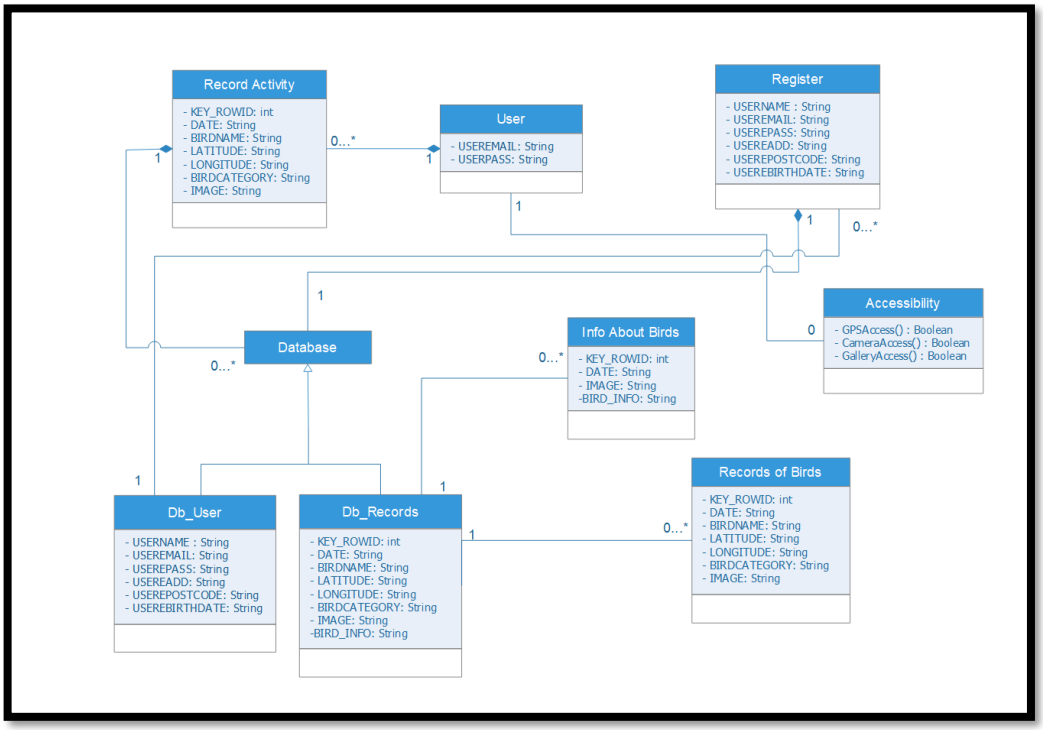


Figure 4-3 : Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects [18]a.

In this class diagram, the relationship of one class to another is shown, as how the databases are linked from one class to another.

- Database Schema

The term "database schema" can refer to a visual representation of a database, a set of rules that govern a database, or to the entire set of objects belonging to a particular user. Read on to find out more about database schemas and how they are used [19].

Database Schema for User Registration

Column Name	Data Type	Size	Constraint	Description
BIRD_ID	INT	50	Primary Key (Auto Increment)	ID No.
USEREMAIL	STRING	50	---	First Name
USERPASS	STRING	50	---	Last Name
USERADD	STRING	50	---	Phone No.
USER-POSTCODE	STRING	50	---	Password
USER-BIRTHDATE	STRING	50	---	Bird Category

Database Schema for Bird Information

Column Name	Data Type	Size	Constraint	Description
KEY_ROWID	INT	50	Primary Key (Auto Increment)	ID No.
DATE	STRING	50	- - -	First Name
BIRDNAME	STRING	50	- - -	Last Name
LATITUDE	STRING	50	- - -	Phone No.
LONGITUDE	STRING	50	- - -	Password
BIRD-CATEGORY	STRING	50	- - -	Bird Category
IMAGE	STRING	50	- - -	Image
BIRD-INFORMATION	STRING	100	- - -	Birds Information

4.5 Snapshots

Figure 4-4 : Snapshot 1

This is the Splash Activity.

Figure 4-5 : Snapshot 2

This is the home screen with three options.

Figure 4-6 : Snapshot 3

The Information of the Birds

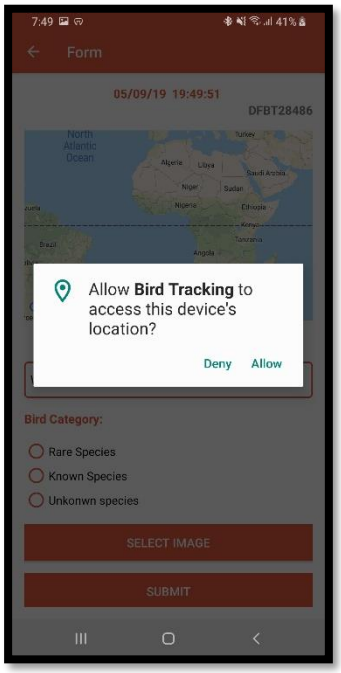


Figure 4-7 : Snapshot 4

Accessing Device location.

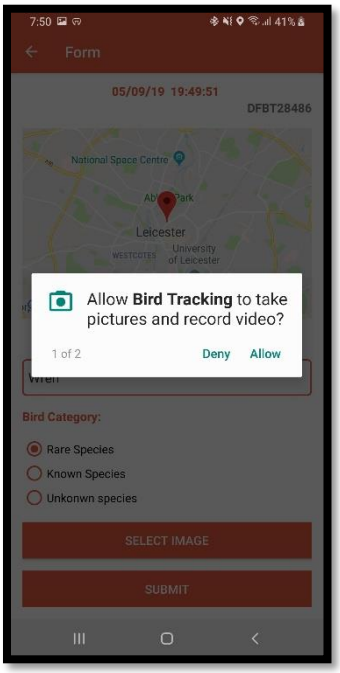


Figure 4-8 : Snapshot 5

Accessing Device gallery.

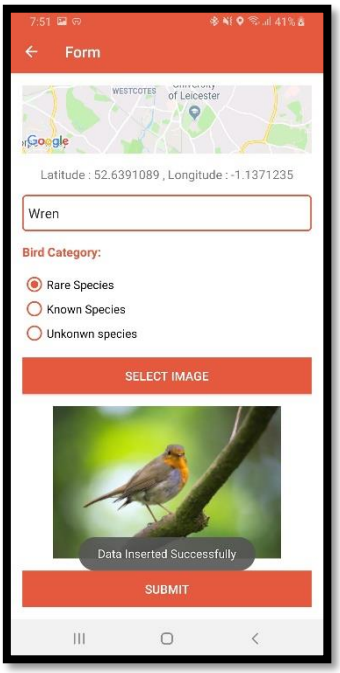


Figure 4-9 : Snapshot 6

The Final form Activity

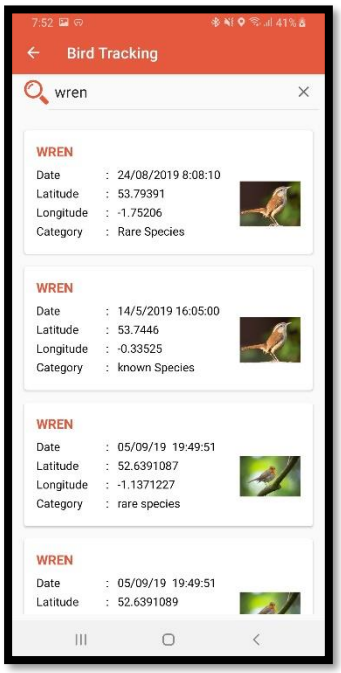


Figure 4-10 : Snapshot 7

Search for Recorded Bird.

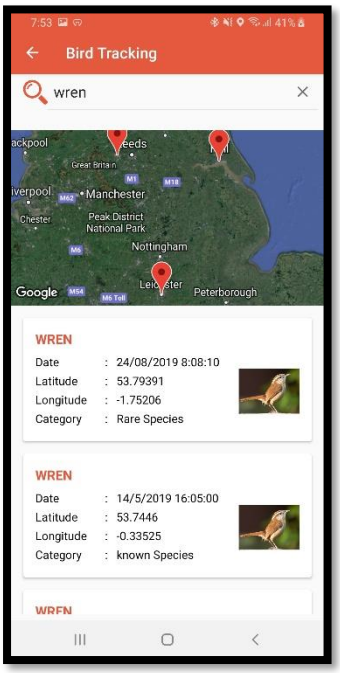


Figure 4-11 : Snapshot 8

Getting location on Google Map.

4.6 Test Cases

Test ID	Case	Test Data	Expected Result	Actual Result	Pass/Fail
1	Login	- User ID - Password	If User ID & password is empty or invalid then display error message otherwise login successfully	If User ID & password is empty or invalid then display an error message.	Pass
2	Register	All fields	If any field is empty, the display message “Blank field”	If any field is empty, display message “Blank field”	Pass
3	Getting Location	Maps	Getting the current location from the User	Getting exact current location from the User	Pass
4	Dropdown list for bird names	List	Selection should be made from the drop down list.	Selection should be made from the drop down list.	Pass

5	Form Page for recoding birds location	All field	If any field is empty, the display message “Blank field”	If any field is empty, the display message “Blank field”	Pass
6	Search for birds location	Search	Birds name should be displayed when the user types the birds' name.	Birds name will be displayed when the user types the birds' name.	Pass
7	Display Recorded Data	Data	The last registered data will be displayed at the end of the list.	Data is been displayed at the end of the list.	Pass
8	Display on Map	Map	The location of the bird stored earlier should be seen on the map as a pointer.	The location of the bird stored earlier will be seen on the map as a pointer.	Pass

- Unit Testing

In unit testing, various modules have been tested individually. This has been done manually to test if the expected result is seen on the screen. Table 1 listed the various unit testing results used for the checking for an application accuracy and efficiency.

- Compatibility Testing

This application was mainly designed for android Smartphone version KitKat (4.4–4.4.4) as it helps users to find the different location of the birds and to also give the location of the bird if it is seen. Different Android phones have different screen sizes and resolution. This application has been made compatible with android devices regardless of their screen sizes and android KitKat and older versions.

- Location Testing

Testing was carried out on different centralized locations of the city as mentioned in form of coordinates. This shows the location of the birds in the application on a point.

5 Conclusion and Future Scope

5.1 Conclusion

This application was made in consideration of the user who is keen to locate the birds nearby. By knowing the name of the bird and by uploading the location, the location of the bird last sighted can be confirmed. Even you can upload a picture of the bird for giving more accurate information of the bird. The accuracy of the birds' location is accurate, the zooming on the google map gives the exact location of the bird. In this application for viewing the birds location the google street map is used. This Application is beneficial for users who are fanatic of birds.

5.2 Future Scope

This application can show weekly and monthly data of the bird located nearby. While uploading this application on google map the database has to be stored on the server through which the application will be monitored by all and the data entered will be seen by all users. The Application can also add the feature like change password, logout and change profile. The data entered by the user itself can also be added as one in the menu.

5.3 Difficulties Faced

Kotlin is an upcoming language and so it changes with every update of android studio.

- The redirecting page was also difficult as none of the code worked, but after applying some extra functions and constructors it started working.
- After adding google API key and after adding code for current location the map didn’t come but after adding some constraints it started appearing.
- Putting data into the database was also a big deal but it changed by changing the code style eventually.

6 References

[1] A. Roy, "Why you should use Kotlin for Android development," Tech Beacon, [Online]. Available: https://techbeacon.com/app-dev-testing/why-you-should-use-kotlin-android-development. [Accessed 04 09 2019].

[2] "Comparison to Java Programming Language," Kotlin, [Online]. Available: https://kotlinlang.org/docs/reference/comparison-to-java.html.

[3] "Inline Functions," Kotlin, [Online]. Available: https://kotlinlang.org/docs/reference/inline-functions.html. [Accessed 04 09 2019].

[4] "Null Safety," Kotlin, [Online]. Available: https://kotlinlang.org/docs/reference/null-safety.html. [Accessed 06 09 2019].

[5] "Type Checks and Casts: 'is' and 'as'," Kotlin, [Online]. Available: https://kotlinlang.org/docs/reference/typecasts.html. [Accessed 04 09 2019].

[6] "Generic," Kotlin, [Online]. Available: https://kotlinlang.org/docs/reference/generics.html. [Accessed 04 09 2019].

[7] "BirdTrack," March 2016. [Online]. Available: https://www.bto.org/our-science/projects/birdtrack/taking-part/birdtrack-apps.

[8] "Leicestershire & Rutland Ornithological Society," June 2003. [Online]. Available: http://www.lros.org.uk/birdrecording.htm.

[9] "Waarneming.nl," [Online]. Available: https://waarneming.nl/.

[10] Observation.org, [Online]. Available: https://observation.org/. [Accessed 04 09 2019].

[11] P. H. a. P. Burton, The Birdlife of Britain, London: Bruce Marshall, 1976.

[12] "Kotlin Collections Overview," Kotlin, [Online]. Available: https://kotlinlang.org/docs/reference/collections-overview.html. [Accessed 02 09 2019].

[13] J. Horton, Android Programming with Kotlin for Beginners, Bimingham - Mumbai, 2019.

[14] eBookFrenzy, Android studio 3.3 Development Essentials, Amazon, 2019.

[15] "Iterative Model - Design," Tutorialspoint, [Online]. Available: https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm. [Accessed 30 08 2019].

[16] "What is Use Case Diagram?," Visual Paradigm, [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/. [Accessed 05 09 2019].

[17] "What is Sequence Diagram?," Visual Paradigm, [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/. [Accessed 05 09 2019].

[18] "What is Class Diagram?," Visual Paradigm, [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/. [Accessed 05 09 2019].

[19] "What is a Database Schema," LucidChart, [Online]. Available: https://www.lucidchart.com/pages/database-diagram/database-schema. [Accessed 06 09 2019].