

---

# ANDROID BLUETOOTH APP

---

## Contents

Project Introduction .....	2
Required Screens for Trail/Demo .....	0
Splash Screen .....	0
Register Screen .....	0
Login Screen .....	1
Chip Connection Screen .....	2
Stats Screen .....	2
Workflows .....	3
Beacon Detection and Registration .....	4
Ping System Flow .....	5
Recovery from failure .....	6
Platform Dependencies & Requirements .....	7
AWS .....	7
Cognito – User Management .....	7
Lambda – Serverless functions .....	7
Admin Access to AWS .....	7
Estimote .....	8
Beacon Connectivity .....	8
Github .....	8
Technical Requirements .....	9
Lifecycle and Background Services .....	9
App Start .....	9
App Active in Foreground .....	9
App in Background .....	9
App Closed .....	9
Assumptions .....	10
Expectations & Deliverables .....	11
Phase 1 Deliverable .....	11
Phase 2 Deliverable .....	11
Phase 3 Deliverable .....	11
Phase 4 Deliverables .....	11

### Project Introduction

The objective of this project is to develop an android app, with Bluetooth connectivity to an Estimote beacon, backedend by AWS Cloud Services

The goal of the app is to record the GPS coordinates of a device within the proximity of an Estimote beacon at a regular time frequency.

The logic of the app requires pinging the BT beacon every 60 seconds, and once a successful ping is received, accessing location services of the device and writing the current GPS coordinates to an AWS backend

This data (once collected and stored) will undergo a transform on the backend. The transformed data will be then be available through an API call for display on the app.

The best analogy to think about this system is the operation of a fitness tracker and a phone app; i.e. fitness trackers track pulse rate and are connected to a phone at all times. The app will then display the statistics collected by the tracker.

The app also requires a user management system and we have selected AWS Cognito to manage user profiles, authorisation etc. There are extensive AWS Android libraries to support this and the app will have to leverage pre-existing code to manage the session handling throughout the logic flow.

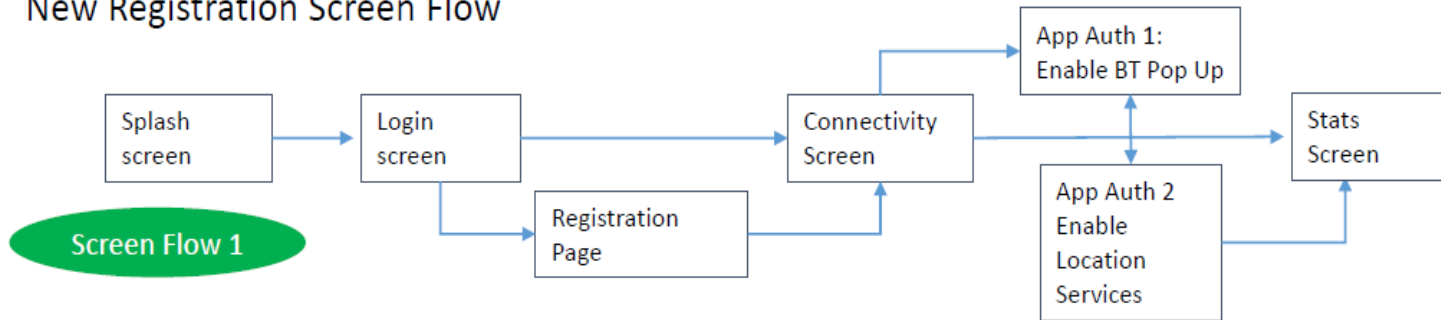
This project requires that the app is constantly communicating with the BT beacon and writing the GPS coordinates to the AWS backend. To achieve this, it will be necessary to develop the app to persist as a background service in android. The app must have the appropriate logic and service capabilities to achieve this while minimising impact on battery.

Please find the required development and deliverable milestones in the following section - [Expectations & Deliverables](#)

## Required Screens for Trail/Demo

Screen Flow – Once logged in, user app must always be recording GPS coordinates and writing to cloud

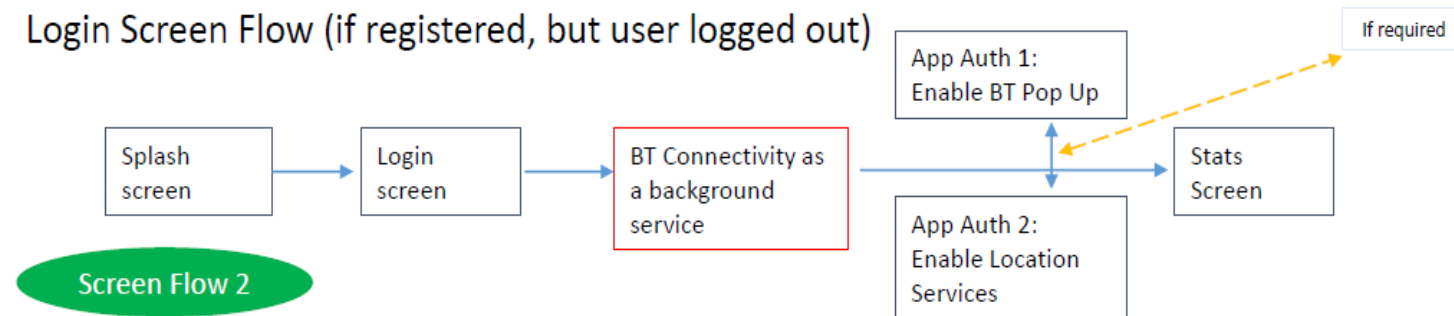
### New Registration Screen Flow



#### Notes

- Most important screen
- Renders the DB stats feed from AWS
- Single AWS RQ to access stats for the device user (logged on)
- Graphics to be defined in phase 2
- Current objective is to display json results information.

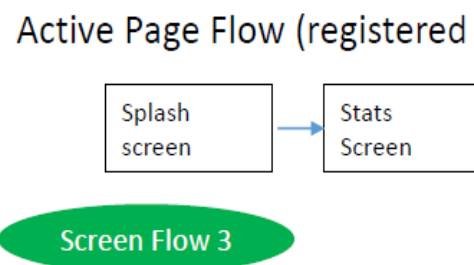
### Login Screen Flow (if registered, but user logged out)



#### Notes


- In this scenario, once the user has signed in and the chip id has been received from Cognito, the app must search for and connect to the BT chip if in range as a background service; monitors for the chip and connects if detected.

### Active Page Flow (registered and logged in)



#### Stats Screen Notes

- **Most important screen** - Renders the DB stats feed from AWS
- Single AWS RQ to access stats for the device user (logged on)
- Current objective is to display json results information.
- Ping and write GPS service continues to execute in background during display
- Graphics to be defined in phase 2

<p>Splash Screen</p> 	<p><b>Screen :</b> Splash Screens; single logo against white background</p>
--	---

<p>Register Screen</p>	<p><b>Screen : Simple form for user registration</b></p> <p><b>Required fields: ----</b></p> <ol style="list-style-type: none"> <li>1. First Name</li> <li>2. Second Name</li> <li>3. Email</li> <li>4. Address</li> <li>5. Password</li> </ol> <p><b><u>User Story</u></b></p> <p>The user will only ever have to launch/visit this screen once in their app engagement story. <b>Once the user has been registered and logged in, in the optimal scenario, they will start permanently logged in.</b> However, if the user does log out, the user story is described in the login screen.</p>
<p><i>Notes</i></p>	<p>AWS Cognito will manage User Profiles and Authorisation etc. There are some sample Android code examples available which detail the process for app connectivity to Cognito User Pools - <a href="https://github.com/aws-labs/aws-sdk-android-samples/tree/master/AmazonCognitoYourUserPoolsDemo">https://github.com/aws-labs/aws-sdk-android-samples/tree/master/AmazonCognitoYourUserPoolsDemo</a></p> <p>This sample code also contains sample UIs for the registration page.</p> <p>The AWS Cognito User pools already exist and the test credentials will be provided to 1) Facilitate the Cognito connection, 2) Test that the app is writing correctly to Cognito.</p>

Login Screen	<p><b>Screen</b> : Simple Login for User.</p> <p><b>Required fields:</b> ----</p> <ol style="list-style-type: none"> <li>1. Email</li> <li>2. Password</li> </ol> <p><b>User Story</b></p> <p>The goal of this app is to continually record GPS coordinates while in the proximity of the beacon; this requires that the user is continually logged in for the background services to operate (i.e. connect to AWS etc)</p> <p>Upon registering, the user will automatically be signed in. From that point on, it will be necessary to keep an updated credentials handshake with AWS services.</p> <p>If for some reason the connection to AWS drops; no internet etc, the appropriate fail-safes will have to be in place to facilitate reconnection. In this circumstance, it may be necessary for the user to log back in and refresh the credentials token with AWS. This scenario will need to be scoped in further detail.</p> <p>Please note: The sign-out functionality is a phase two development; it is not included of the scope of the initial phase</p>
Notes	<p>The user login will be authenticated against the AWS Cognito profile saved for this user. This is a token-based engagement with AWS. Once the user has been signed in, the app will receive credentials to access AWS services and load the next page in the flow (See screen flow chart).</p> <p>Please note, that the beacon UUID will be saved with the user profile in the Conito record upon initial beacon connection. This is returned to the user/app upon login to facilitate</p>

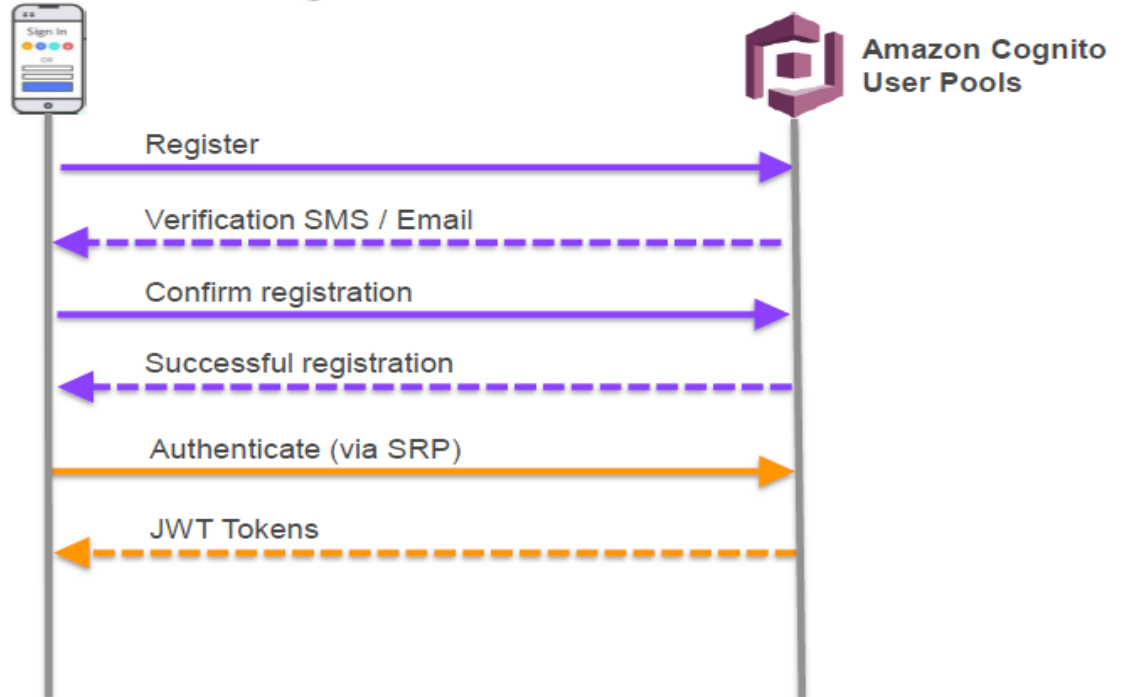
Chip Connection Screen	<p><b>Screen :</b> 1) Scan and find beacon 2) Connect to beacon</p> <p><b>Required UI widgets: ----</b></p> <ul style="list-style-type: none"> <li>• Auto ask user to turn on Bluetooth (if not already enabled)</li> <li>• Auto ask user to turn on location services (If not already enabled)</li> <li>• Scan for beacon button. <ul style="list-style-type: none"> <li>○ Scan progress bar</li> </ul> </li> <li>• Once Estimote beacon has been found, provide pop up to user to confirm.</li> </ul> <p><b>User Story</b> First time the user scans for the BT beacon to register the beacon with their user profile.</p>
Notes	<p>This process will be built on the Estimote API for connectivity. <a href="#">See Estimote section</a></p> <p>This is the setup screen so the appropriate checks should be built in here; BT on, GPS on etc. Once GPS and BT have been activated, the app should automatically scan for the Estimote beacon. Once detected, a message should indicate that the beacon has been found while in the background, the app will write the UUID of the beacon to the user-profile on Cognito.</p> <p>The app should pair to the beacon at this point. Once paired, the ping system logic should commence.</p>

Stats Screen	<p><b>Objective:</b> Display user stats.</p> <p><b>Required UI widgets:</b></p> <ul style="list-style-type: none"> <li>• Display JSON information provided from the backend API call.</li> </ul> <p><b>User Story</b> This is the primary screen for the user. Once the app has been configured and is recording the GPS markers of the device; this screen will act as the interface screen for the user stats.</p> <p>This is the “engagement” screen and will ultimately be a rich graphic screen. The UI for this screen is required in a later phase for this project. The user will interact with components built on their recorded stats; i.e. animations, graphs, etc.</p>
Notes	<p>Please refer the app screen flows to establish when this screen is active. Once the user profile has been created and the BT chip is associated with the user, the stats screen is primary screen that will load when the user opens the ap.</p> <p>This screen is built on the data retrieved from a API call to a Lambda fxn on the AWS backend. The request is sent on screen load and is loaded into the page when received. In phase 1, the requirement is to display the info in a panel on the screen.</p> <p>The AWS Lambda sdk for android and AWS credentials sdk are required for this screen</p>

## Workflows

Registration flow.

## Sign-up and Sign-in

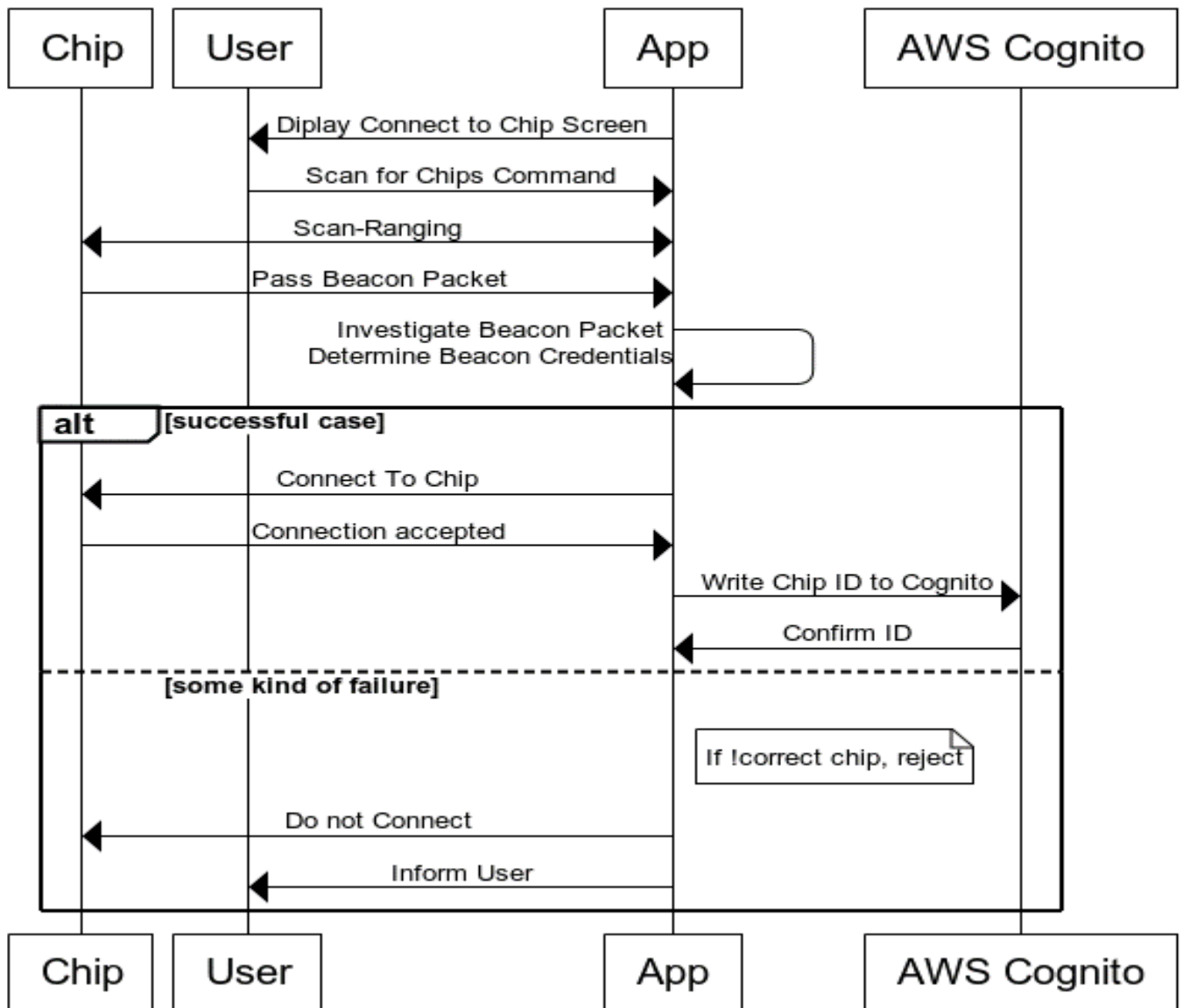
Authentication Workflow:

<https://aws.amazon.com/blogs/mobile/customizing-your-user-pool-authentication-flow/>



Beacon Detection and Registration

### Beacon Registration Flow - Ranging



## Ping System Flow

This is the key activity flow of the app. For the majority of the workflow lifecycle, this activity will run as a background service. Please see [background services](#) for more information on this. This workflow should be implemented as a Scheduled Job on the Android system.

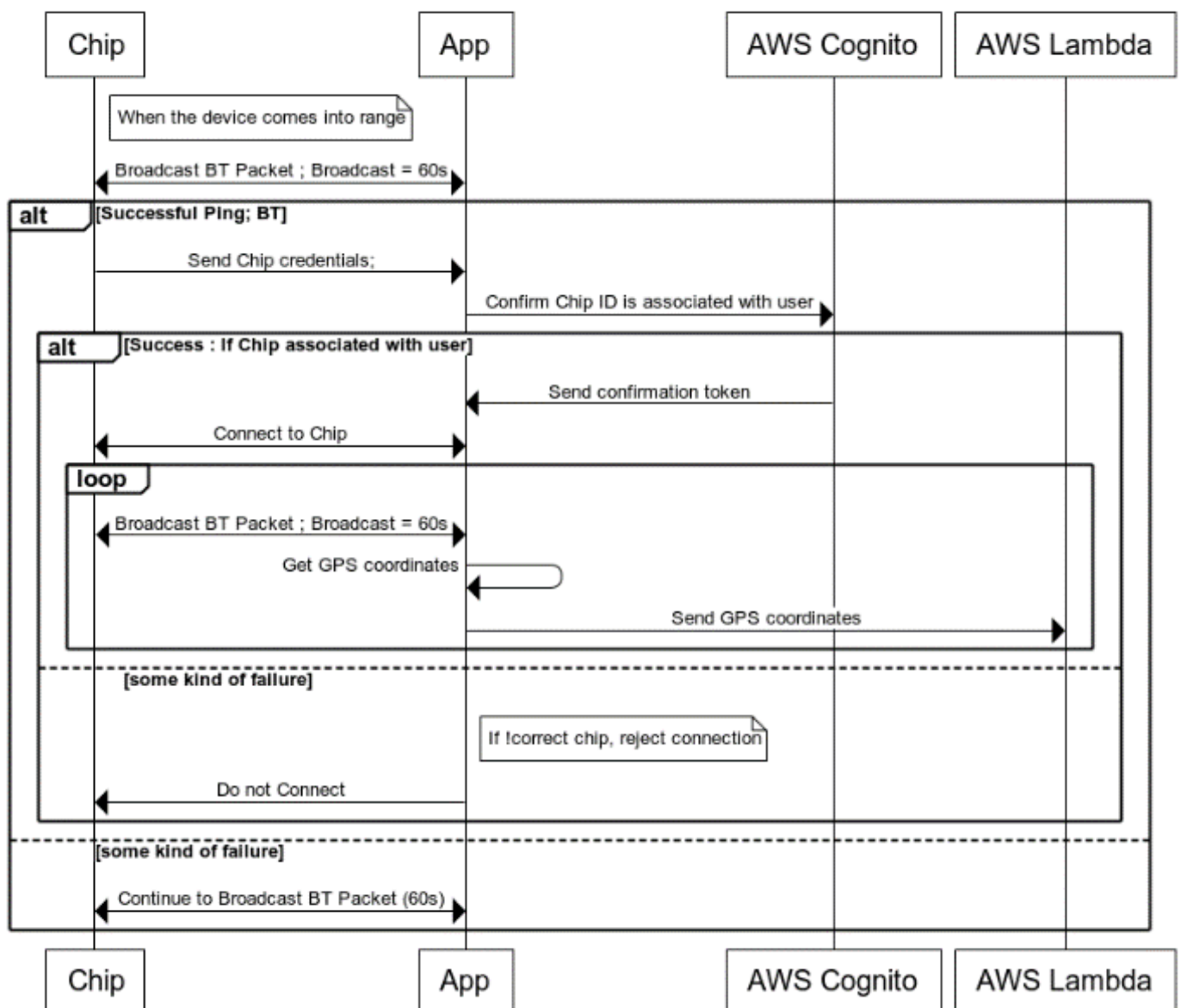
This chart documents the interaction between the beacon, the app, and AWS within the ping system. **Every 60 seconds**, the BT Beacon will pulse to the app/phone. Each device must be configured to receive/send this ping accordingly. Once a successful ping has been received, the app will request the current GPS coordinates from the phone's location services. Once retrieved, the GPS coordinates are written to AWS through an API call (Lambda). The data should encapsulate the following...

- UUID
- Latitude
- Longitude

The AWS API details will be provided for testing purposes.

The process should only execute if the chip is within range.

## Connected Flow - Monitoring



### Recovery from failure

This flow must be able to recover gracefully from a number of failure scenarios i.e. network availability, error response from AWS etc

These scenarios will be detailed in v2 of this document.

## Platform Dependencies & Requirements

### AWS

#### Cognito – User Management

Amazon Cognito lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0. Cognito handles authentication.

Amazon Cognito User Pools provide a secure user directory that scales to hundreds of millions of users. As a fully managed service, User Pools are easy to set up without any worries about standing up server infrastructure.

Amazon Cognito provides solutions to control access to backend resources from your app. You can define roles and map users to different roles so your app can access only the resources that are authorized for each user.

Link to AWS Cognito overview - <https://aws.amazon.com/cognito/>

Link to AWS Cognito on Github - <https://github.com/aws-labs/aws-sdk-android-samples>

Link to AWS Docs for Android setup to interact with Cognito User Pools - <https://docs.aws.amazon.com/cognito/latest/developerguide/setting-up-android-sdk.html>

Link to github for above - <https://github.com/aws-labs/aws-sdk-android-samples/tree/master/AmazonCognitoYourUserPoolsDemo>

#### Lambda – Serverless functions

Overview - <https://aws.amazon.com/lambda/>

Dev Guide - <https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>

Android SDK for Lambda - <https://github.com/aws/aws-sdk-android>

An AWS Lambda function will be configured to receive the post data sent by the phone every 60 seconds. The key information fields to be sent/captured are...

- UUID
- Latitude
- Longitude

Please note you will not have to develop the Lambda fxn. An API call will be provided to you for this piece.

#### Admin Access to AWS

For test purposes, it may be necessary to access AWS Console to confirm code execution against AWS Cognito. If required, a user profile will be set up to facilitate access to the required services. To be discussed further

## Estimote

This project will use Estimote stickers as the BT beacon for the POC. Stickers can broadcast either iBeacon or Eddystone protocols but for the purposes of this app, the Bluetooth Stickers will just facilitate a ping communication with the app. This will be a very basic packet, UUID and some other basic information for capture and recording.

As mentioned earlier, the objective of the app is to constantly record GPS coordinates, when in range of the BT chip. To facilitate this, the app will be “conscious” of the BT UUID; it is associated with the device/app user. The APP

Estimote Stickers, leverage the Estimote Nearables protocol and API.

### [What are region Monitoring and Ranging?](#)

<https://github.com/Estimote/Android-Proximity-SDK>

Estimote allows for API connectivity up to 20 stickers for POC and testing purposes (i.e. no API transaction costs). It also has a beacon developer/management tool online and provides a number of app code templates to speed up development. It would be advantageous if the selected developer has experience of the Estimote online interface for beacon management. Estimote account credentials are available however; ideally, the developer would have an Estimote account.

## Beacon Connectivity

The app-beacon communication must transmit the following information on each ping (every 60 seconds)

- UUID of the beacon
- Major
- Minor

The received UUID requires authentication against AWS Credentials to ensure the UUID correct; please see [workflows](#). To be discussed.

## Github

A GitHub account will be provided to facilitate transfer of completed code. It is anticipated that there will be local testing required once the initial transfer has been completed. To be discussed

## Technical Requirements

### Lifecycle and Background Services

This section details the process for each app state and the activity expected of the app during this state. This is not an exhaustive list and will require review to ensure all states (and logic) is captured.

Please refer to the [overview screen flow graph](#), which visualises the following.

#### App Start

Flow	Notes
Screen Flow 1	App initiation process, load to registration....
Screen Flow 2	App login process – sign user in.
Screen Flow 3	User is logged in and app loads straight to stats engagement screen.

#### App Active in Foreground

Flow	Notes
Screen Flow 1	As per workflow chart
Screen Flow 2	As per workflow chart
Screen Flow 3	Stats screen will be the primary view in this state.

#### App in Background

Flow	Notes
Screen Flow 1	As per workflow chart
Screen Flow 2	As per workflow chart
Screen Flow 3	Stats screen will be the primary view in this state.  The <a href="#">ping system flow</a> should execute at all times when the app is open and in the background.

#### App Closed

Flow	Notes
Background Services	The <a href="#">ping system flow</a> should continue to execute at all times when the app is not active and the chip is in range. This should execute as a scheduled Job.

## Assumptions

To be discussed and agreed..

1. Project scope, timelines, milestones, technology and deliverables.
2. Rules of engagement;
  - a. Developer access and availability.
  - b. Recruiter access and availability
3. Access to dependency credentials.
4. Access to BT Beacon hardware
5. Project costings and job quote.
  - a. Phased deliverable and quote per deliverable.
6. AOB

## Expectations & Deliverables

The entire app is to be completed and alpha tested by 31<sup>st</sup> May.

The developed app code to be delivered to a Github repository.

### Phase 1 Deliverable

The first deliverable must consist of an Android app project (APK) with the following:

#### Screen & Logic

1. Splash screen (optional)
2. [Screen with BT Scan and Connect to Estimote Beacon](#)
  - a. Beacon ranging, discovery and writing chip to AWS Cognito profile
3. Screen that displays a message after the GPS coordinates have been successfully written to the AWS Backend
  - a. [Beacon ping system](#) ( every 60 seconds)
  - b. Retrieving GPS coordinates (on successful ping)
  - c. Writing GPS coordinates (and required metadata) to backend.
  - d. This above logic must be developed as a background service

**NB – please note;** fixed AWS credentials will be provided to support the development of phase 1.

This is the core logic of the app and must be delivered in the first milestone to support testing the backend with live data.

### Phase 2 Deliverable

The first deliverable must build upon the release of phase 1 with the following:

#### Screen

1. Stats Screen with retrieve stats button.

#### Logic

- Connection to AWS backend API and retrieval of data
- Text render of JSON results on screen

### Phase 3 Deliverable

- User management system & screens: To be detailed further in v2 of this document.

### Phase 4 Deliverables

- UI and Graphical components : To be detailed further in v2 of this document