Android Beacon Platform used to detect beacons and send users push notification by using Proximity Beacon API and Nearby Messaging API. Due to unintended misuse Google has since stopped the service. However, instead this is a project an alternative to the same model using other workaround. It is possible to detect nearby Beacons using Android Beacon Library and send automated push notification to mobile devices using Firebase (Realtime database, Cloud Functions and Messaging Services etc).

Detecting Beacons using Android Beacon Library

To use the library we need to get a reference to the beaconManager class and define the type of Beacon we want to detect and bind this. There are AltBeacon, iBeacon, EddyStone etc. This project was done for EddyStone beacons and hence the beacon layout are provided. BeaconLayout are generally some string format defined like “m:2-3=beac,i:4-19,i:20-21,i:22-23,p:24-24,d:25-25”. These are easily found in a google search.

```
beaconManager = BeaconManager.getInstanceForApplication(this);

beaconManager.getBeaconParsers().add(new BeaconParser().
setBeaconLayout("m:2-3=beac,i:4-19,i:20-21,i:22-23,p:24-24,d:25-25"));

// Detect the main identifier (UID) frame:
beaconManager.getBeaconParsers().add(new BeaconParser().
        setBeaconLayout(BeaconParser.EDDYSTONE_UID_LAYOUT));

// Detect the telemetry (TLM) frame:
beaconManager.getBeaconParsers().add(new BeaconParser().
        setBeaconLayout(BeaconParser.EDDYSTONE_TLM_LAYOUT));

// Detect the URL frame:
beaconManager.getBeaconParsers().add(new BeaconParser().
        setBeaconLayout(BeaconParser.EDDYSTONE_URL_LAYOUT));

beaconManager.bind(this);
```

Next we need to implement the BeaconConsumer class. All the beacon related work happens in the onBeaconServiceConnect() method. So the library detects beacon by monitoring and ranging beacons. First, we need to define a Region for which we want to

detect beacon. Then, we need to monitor that Region. By adding a monitor notifier class we can monitor beacons in that Region by calling methods like didEnterRegion() or didExitRegion().

```
beaconManager.startMonitoringBeaconsInRegion(firstBeaconRegion);
beaconManager.startMonitoringBeaconsInRegion(secondBeaconRegion);

beaconManager.removeAllMonitorNotifiers();

beaconManager.addMonitorNotifier(new MonitorNotifier() {
@Override
public void didEnterRegion(Region region) {

try {
beaconManager.startRangingBeaconsInRegion(region);

//Update Firebase Realtime Database …

} catch (RemoteException e) {
e.printStackTrace();
}
}

@Override
public void didExitRegion(Region region) {
try {
beaconManager.stopRangingBeaconsInRegion(region);
} catch (RemoteException e) {
e.printStackTrace();
}
}
});

beaconManager.addRangeNotifier(new RangeNotifier() {

@Override
```

```
public void didRangeBeaconsInRegion(Collection beacons, Region region)
{
//further beacon processing
}
});
```

Once a beacon is detected in a Region a BeconModel is formed using the BeaconModel class for each beacon. BeaconModel class is formed using properties like beaconID, distance and message etc.

We use FirebaseDatabase and FirebaseMessaging service in this project to use backend database and push messaging system in the App using following dependencies in:

```
//firebase messaging
implementation 'com.google.firebase:firebase-messaging:20.2.0'

//firebase database
implementation 'com.google.firebase:firebase-database:19.3.1'
```

Updating Database and Sending Push Notification

As we have reference to the Firebase Database in the app each beacon is pushed into Firebase Realtime NoSQL Database by using a query like this :

```
//Firebase database setup database = FirebaseDatabase.getInstance();
beaconDBRef = database.getReference().child("messages"); //write to
database
beaconDBRef.child(beaconModel.getBeaconName()).setValue(beaconModel);
```

Which in turn updates values for each beacon in the Database like this:

- messages
  - _0x0e32fea91b13

        - _beaconName: “0x0e32fea91b13”
        - _distance: 0.3035367107261693
        - _message: “Welcome second beacon “
    - _0xa9937b420872
        - _beaconName: “0xa9937b420872”
        - _distance: 0.7036247189071709
        - _message: “Welcome First beacon “

Additionally the Application has an ADMIN button which allows the owner/ADMIN of the beacon to change the default message of the beacons whenever they want to. Once the message is updated through the application, the application stores the data locally using SharedPreferences so that the message is not lost during subsequent app restart.

The messages for each beacon will get updated in the Firebase only during the next beacon data update that is when the beacons go out of range or they have been restarted. This will then update the Firebase RealTime database with the message entered from the application. Any next beacon detection (that is when beacon in Range once) will be detected by Firebase Cloud Functions and will send the push notification comprising of the message to the android device.

To Implement the Firebase Cloud Function however we need to use Node.js. Using npm we install necessary plugin for firebase in local machine and initiate a FirebaseCloudFunction project. Once a function is developed and tested locally it is deployed in the FirebaseConsole. So our FirebaseCloudFunction detects Realtime database changes and sends push notification using following javascript code :

```
exports.sendBeaconNotification =
functions.database.ref('messages/{pushId}')
.onWrite( (change, context) =&gt; {

const beacon = change.after.val();
const beaconName = beacon.beaconName;
var beaconMessage = beacon.message;
console.log('beaconObject', beacon);

const payload = { notification: {
```

```
title : 'Welcome to our Store!',
body : "" + beaconMessage
}};

return admin.messaging().sendToTopic("global", payload)
.then(function(response){
console.log('Notification sent succesfully:', response);
})
.catch(function(error){
console.log('Notification sent failed:', error);
})
});
```

In Android a push notification is handled using pendingIntent and by showing a notification in the MyFirebaseMessagingService class which extends the FirebaseMessagingService Class and implements methods like onMessageReceived() or onNewToken() etc.

It’s in the onMessageReceived() a new push notification is shown to the user using pendingIntent:

```
Intent intent = new Intent(this, MainActivity.class);
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /*
Request code */, intent,
PendingIntent.FLAG_ONE_SHOT);

String channelId = "1"
;//getString(R.string.default_notification_channel_id);
Uri defaultSoundUri =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
NotificationCompat.Builder notificationBuilder =
new NotificationCompat.Builder(this, channelId)
.setSmallIcon(R.mipmap.ic_launcher)
.setContentTitle(title)
.setContentText(messageBody)
.setAutoCancel(true)
```

```
.setSound(defaultSoundUri)
.setContentIntent(pendingIntent);

NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
```

The project is so far done for statically defined beacons. However, with further work it can be converted to detect beacons dynamically and send targeted push notification to users.