CCExtractor: Beacon

Basic Information

Name	Nishtha Bodani
University	Indian Institute of Technology, Varanasi
Degree	Bachelor of Technology
Year	Sophomore
Timezone	Indian Standard Time (GMT +5:30)
Email	nbodani8@gmail.com
Resume	NISHTHA BODANI - RESUME
Github	nb9960
LinkedIn	https://www.linkedin.com/in/nishtha-bodani/
Postal Address	10/M/7 (3rd floor), Dabauli, Kanpur, Uttar Pradesh, India - 208022
Phone Number	(+91) 8840965840

Project

Abstract

This project aims to develop a flutter build native interface to ease the group travelling (or hiking). By using this, the group leader would be able to share his location with the entire crew, and in case if someone loses contact with the group, he can quickly get in the right place by following the beacon.

Motivation

Since last year, I started to build a keen interest in flutter because of it's highly growing community and easy-to-understand documentation.

Also, the idea behind this project itself motivates me, as I would be able to learn more while contributing to this project, and this ensures that I commit my maximum time towards this project.

Why CCExtractor

The wide range of topics and technologies CCExtractor consists of, intrigued me to contribute to this society. By joining the community, there are much greater chances that I would be able to learn more on not just my project but also other topics that include peer to peer, AI, cloud, and other technologies. I have seen the exciting quality discussions on community channels which thereby motivates me to explore more stuff. At last, the supportive environment that mentors provide is one of the essential factors that I would love to contribute to this organisation.

I'm only applying for CCExtractor for GSoC'21 and have no plans to contribute to any other organisation.

Proposed Deliverables

- ➔ A Flutter application that uses firebase (or other suggested Backend as a Service tool) by which the user can share his location with his group.
- ➔ A leader can pass on the beacon, i.e., can change the leader to anyone else in the group, and would have the right to remove a user from the group.
- ➔ Each beacon would have a shortcode (namely passkey), and beacons can be accessed either by sharing a URL or just the passkey.
- ➔ A leader can set the duration of the beacon and can change it. Or he can preschedule the beacon which would get activated automatically.
- ➔ A leader can add landmarks and can share the route followed from the start of the beacon till the current location (the current location may be the end location).
- ➔ Extending the application to create `N` beacons for a single group (each beacon having a different leader)
- ➔ Adding unit tests and widget tests for all the necessary views.

➔ Proper **end-to-end testing** in the **production-ready environment** to ensure that the application doesn't break in iOS as well as android.

Background Info

The application that I had developed uses firestore for storing data, and the current data model representing the beacon (which needs modification) is:

```
{
        "leader": "name_of_leader",
        "followers":["array_of_joinees"],
        "expiryDate": "timestamp_format",
        "lat": "coordinate of leader",
        "long": "coordinate of leader"
}
```

The features developed in this demo application are:

- ➔ Uses firebase_dynamic_link to share the URL along with just sharing passkey for inviting other users directly to view the location
- ➔ Uses geocoder to get the name of the current location
- ➔ A leader can pass the beacon (but the leader's location won't update in real-time for beacon followers)
- ➔ Duration can be updated

Plan of Action

Since I've already initiated the project so I'll start with resolving the bugs in the current codebase. For example, currently, the leader's panel doesn't take real-time updates of the users who joined the beacon, for this, StreamBuilder needs to be added. Also, the codebase needs refactoring (I would follow the suggested architecture, maybe stacked architecture) and adding provided backend configuration (as I have used the firebase project and my own configuration file) and Google Maps API needs to be added.

Currently, neither marker for the current location nor any route is displayed, for this, I'll be using Polyline package to display routes, and the Marker option provided by google_maps_flutter to add a set of markers (landmarks) added by the leader along with the current location marker.

Currently, only the expiry DateTime is stored after taking the duration from the leader.

Also, whenever the app restarts and the beacon is active, the user has to login again, this behaviour needs to be fixed. For pre-scheduling beacon, start and end date both need to be specified. Also, if a user is following a beacon, it needs to be stored either using shared preferences or local storage.

Once we store the data model for a beacon in local storage, it would be easier to add a feature of `N` beacons for one group of users as fetching the list of active beacons for the current passkey would not affect app performance. (though this requires a completely modified data model which I would be discussing in detailed plan and implementation).

Timeline

Following is a breakout of the time I'll spend on different aspects of the Project:

60% - Adding major features
10% - UI finishing and solving bugs that arise after implementing the said features
20% - Adding unit tests for significant classes and widget test for some components
10% - Adding automation using Github Actions for best development practice and documenting the entire codebase

Community Bonding Period — May 17 - Jun 06

- I'll discuss the final data model and backend configuration (if firebase is finalized then I'll configure the app)
- I'll discuss the final UI and state management (whether to use any plugin or not)
- Setup all required API keys such as for google map services.
- I'll discuss the expected documentation and testing standards.
- Setup other logistics i.e GSoC blog.

Week 1 — Jun 07 - Jun 13

- Redoing the current codebase keeping the best architecture and practices in mind.
- I'll start with configuring essential services (Backend service, i.e, firebase for now, as well as Google maps API).
- I'll be adding a StreamBuilder for the leader's panel to get real-time updates on who joined the beacon.
- Implementing a smooth fallback mechanism for when the user declines to share their location.

Week 2 — Jun 14 - Jun 20

- Implementing real-time location updates in google maps by adding location change listener event provided by **location package**
- Moving the Google map current location marker upon location changing and updating the location displayed in the panel accordingly.
- I'll be adding **polyline package** to display the route between initial and current coordinates.

Week 3 & 4 — Jun 21 - Jul 04

- I'll start with updating the data model which would support add-on features.
- I'll be adding a share route feature for the case when beacon is active.
- I'll be adding an option for the leader by which he can add landmarks that would be stored in a set of markers and would be displayed to the joinees.
- I'll be adding an active beacon's data model in local storage.
- I'll be adding an option to pre-schedule the beacon (URL would be generated just after creating a beacon for sharing).

Week 5 — Jul 05 - Jul 11

- Adding support to update the coordinates when beacon is transferred and the leader changes (this might require to save the coordinates of each user).
- Seamlessly updating the location in all devices that have joined the beacon.
- Activating and deactivating beacons automatically according to the set DateTime.
- Modifying UI for the deactivated beacons.

Evaluation — Jun 12 - Jul 16

- All the major expected functionalities (except N beacons for one group) would be ready for testing and the application would be in a presentable state which means it can be deployed for beta testing too.
- Since I won't be able to test for iOS, I'll ask my mentors to discuss the problems that occurred in iOS.

Week 6 — Jul 12 - Jul 18

- Modifying data model to support N beacon one group feature.
- Displaying N beacons corresponding to a passkey and adding an option to switch the beacons.
- Modifying UI to add an option to switch the beacon seamlessly.

Week 7 Jul 19 - Jul 25

- This week is kept as a buffer week to complete any leftover (if any) work.
- I'll fix the bugs which would have occurred after implementing the said features.
- Fixing bugs to support iOS.
- Implementing all other suggested features based on the reviews of mentors from evaluation.
- I'll work on the user interface and beautify the application using the amazing widgets provided by Flutter. Also, I'll add basic animations (wherever necessary), without slowing down the app.

Week 8 & 9 Jul 26 - Aug 08

- I'll go through the flutter testing docs thoroughly to implement better testing practices.
- I'll add widget tests for all essential components and enough integration tests to cover all the important use cases and better performance profiling.
- I'll take care to extract out repeated code by grouping the unit and widget tests wherever possible.

Week 10 (Ultimate Week) Aug 09 - Aug 15

- After writing all the tests, I'll write a basic script to automate running tests by the Github CI.
- I'll be documenting the code for the entire codebase based on the community standards.
- I'll clean up the documentation and code and will wrap this up to ensure that it is organized perfectly.
- I'll finalize and test the application on both the platforms (android and ios) and prepare the configuration file (if required) for successful deployment.
- For the final week, I'll focus on the reviews given by my mentors and discuss with the community what other features can be added to improvise the user experience.

Final Evaluation **Aug 16 - Aug 23**

- The production-ready environment can be tested.
- All unit and widget tests can be run and they all will pass.
- The community can review the documentation and comment whether it follows the community standards or not
- The codebase will follow the best possible flutter practices with understandable architecture.

Detailed Plan and Implementation

The leader's panel in the current application doesn't update whenever a new user joins the beacon. I'll add a StreamBuilder to display the real-time updates in the beacon model.

Proposed data model

In a final data model, we'll be having beacons field as a subcollection, for which I have defined beacon's model separately. And in beacon model we'll be having joinees and landmark field as a subcollection, for which I have defined user's model and landmark model respectively: (The thing that needs to be discussed is whether joinees list must be same for entire group or different list for each beacon)

Main collection:

```
{
 "passkey": "unique id for a group with number of beacons",
 "beacons": "itinerary model having below beacon model as a subcollection"
}
```

Beacon's subcollection:

```
{
      "beacon_id": "unique id to identify each beacon",
      # this will help when user wants to switch the beacon
      "beacon_title": "name for beacon, like heading to Delhi",
      "leader": "username_of_leader",
      #itinerary model having user model as a subcollection
      "joinees":[
            #storing coordinates of each user is necessary because when
            Leader is switched then we must have coordinates to seamlessly
            update map accordingly
            {
                #unique username for each user and by default user_id
would be stored in place of username if user wants to stay anonymous
            "username": "unique username of joinee",
            "lat": "latitude of user's location",
            "long": "longitude of user's location",
             },
      ],
      "startDateTime": "timestamp_format",
      "expiryDateTime": "timestamp_format",
```

```
      "initial_lat": "initial coordinate of leader",
      "initial_long": "initial coordinate of leader",
      "current_lat": "current coordinate of leader",
      "current_long": "current coordinate of leader",
      "landmarks": [
          {
            "title": "name_for_location",
            "lat": "latitude of location",
            "long": "longitude of location",
          },
      ],
}
```

Displaying route and markers, and updating leader's current location marker

I'll be using the flutter **polyline** package to display the route between initial location and current location. To set polyline I'll be using following sample code:

```
setPolylines() async {
 List<PointLatLng> result = await
     polylinePoints?.getRouteBetweenCoordinates(
        googleAPIKey,
        INITIAL_LOCATION.latitude,
        INITIAL_LOCATION.longitude,
        CURRENT_LOCATION.latitude,
        CURRENT_LOCATION.longitude);
      if(result.isNotEmpty){
      // loop through all PointLatLng points and convert them
     // to a list of LatLng, required by the Polyline
     result.forEach((PointLatLng point){
        polylineCoordinates.add(
           LatLng(point.latitude, point.longitude));
     });
  }
  setState(() {
     // create a Polyline instance with an id, an RGB color and the list
of LatLng pairs
     Polyline polyline = Polyline(
        polylineId: PolylineId("poly"),
        color: Color.fromARGB(255, 40, 122, 198),
```

```
        points: polylineCoordinates
      );
      // add the constructed polyline as a set of points to the polyline
set, which will eventually end up showing up on the map
      _polylines.add(polyline);
    });
}
```

For sharing the route, I'll use **firebase_dynamic_links** and **share** package and will pass initial and current coordinates of the leader along with passkey and beacon id to share the route.

I'll be using the flutter **location** package to update the location of the leader seamlessly. To update the location of leader in real-time, I'll declare the following function for **leader only**:

```
location.onLocationChanged().listen((LocationData cLoc) {
// push the currentLocation to firebase.
// this syntax might be incorrect as I'll be updating the data model and
// this is w.r.t the current data model I'm using, i.e, one beacon for one
// group. For the updated one check on id needs to be passed, i.e, if
current_beacon_id == snapshot.beacon_id
_firestore.collection('hikes').doc(passKey).update({
      'current_lat': cLoc.latitude,
      'current_long': cLoc.longitude,
    });
});
```

And then if the user is a follower (and not the leader) then I'll add a stream to get real-time updated data. Also, marker for current location would be moved seamlessly using sample code below:

```
void updatePinOnMap() async {
    // create a new CameraPosition instance every time the location
changes, so the camera follows the pin as it moves with an animation
    CameraPosition cPosition = CameraPosition(
      zoom: CAMERA_ZOOM,
      tilt: CAMERA_TILT,
      bearing: CAMERA_BEARING,
      target: LatLng(currentLocation.latitude,
      currentLocation.longitude),
    );
```

```
    final GoogleMapController controller = await _controller.future;
    controller.animateCamera(CameraUpdate.newCameraPosition(cPosition));
setState(() {       // updated position
      var pinPosition = LatLng(currentLocation.latitude,
      currentLocation.longitude);
      // the trick is to remove the marker (by id)
      // and add it again at the updated location
      _markers.removeWhere(
      (m) => m.markerId.value == 'sourcePin');       _markers.add(Marker(
          markerId: MarkerId('sourcePin'),
          position: pinPosition, // updated position
          icon: sourceIcon
      ));
    });
  }
```

(Please note this may impact the application's performance but to fix this we can tweak how frequently onLocationChanged() is called)

Adding Landmark: The current location marker would be customizable and on tapping it would display a dialog asking for title (name user would like to define this location with) and then coordinates would be stored in an array and then a set of markers would be displayed by fetching that list from the backend. (google_maps_flutter provides an option to display markers for the set of markers which I would use).

Pre-scheduling beacons and activating and deactivating them automatically & N beacons for one group

To pre-schedule events and to make sure if any joinee comes in before the start DateTime we just need to add a condition and to add a Stream to check for it so that **DateTime.now() <= Start_DateTime** (and the leader's application is active then location updates).

To deactivate the beacon on a leader's demand **locationData.cancel() or locationData.pause()** needs to be called. If a pause method is called then the leader can resume the tracking too by calling **locationData.resume()** (this might be helpful in case when the group takes rest for a few minutes and the leader doesn't want to drain his battery).

N beacons one group

For having N beacons corresponding to one group, I've defined the user model above. Other than that a drawer or maybe a switch button that would then open a panel needs to be added so that the user can switch to a beacon. Now when the user switches the beacon, ideally his name would be removed from the previous beacon and would be added to the new one. While implementing changes for N beacons one group, change in the URL that is passed corresponding to one beacon would need modifications, i.e, beacon_id, and group passkey both would have to be passed as a parameter.

Testing and production-ready environment

Testing: I believe that tests can very well explain the intention of the programmer for a piece of code, better than comments can do as writing tests ensures that developers catch the regressions at an early stage.

To **unit test** any particular class in isolation, any external dependencies (classes) needs to be mocked so no external noise affects our tests. The mocked object would simulate the behavior of a real object, but knows in advance what's supposed to happen in the test and its intended behavior. I'll use **mockito** package to add mock classes.

For **widget testing** of all files in UI, I would use WidgetTester along with pump() method for testing Stateful Widgets and would refer to this article.

At the end of the development process, I'll add the rest of the tests like **integration tests** to ensure proper **performance profiling.** Integration test mimics user behaviors. The goal would be to verify that all the widgets and services being tested work together as expected. This will also check if the ScrollView behavior is working smoothly and the application is free of 'jank', i.e, no frame is being skipped.

To maintain proper testing standards, I'll refer to the flutter cookbook.

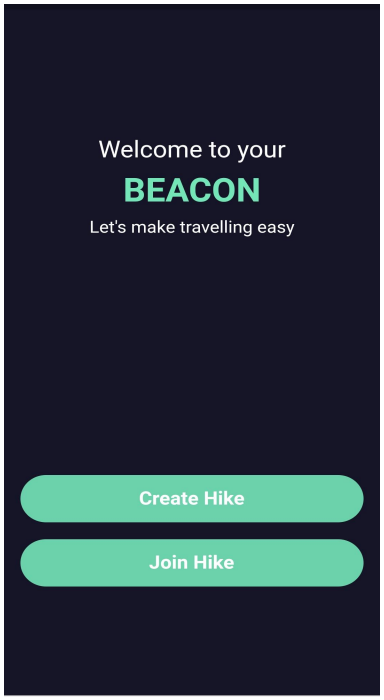
Production-ready environment: I'll be adding the following piece of code to ensure proper developing workflow, i.e, on each commit to the main branch, testing and analyzes would occur to ensure application doesn't break anywhere. Also, as I won't be able to **test for iOS** so I'll add a feature to test app in iOS after every commit to ensure nothing breaks in iOS as well. And if needed I'll add issue_template and pull_request_template.

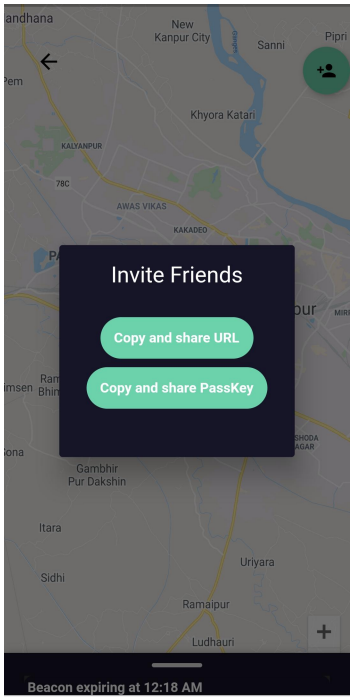
```
name: Flutter CI

on:
 push:
   branches:
     - master
 pull_request:
```

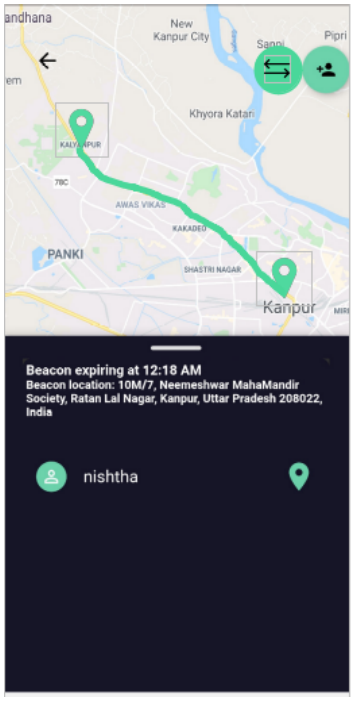
```yaml
jobs:
 build:
   strategy:
     matrix:
       platform: [ubuntu-latest, macos-latest, windows-latest]
   runs-on: ${{ matrix.platform }}
   steps:

   # Setup Java environment in order to build the Android app.
   - uses: actions/checkout@v1
   - uses: actions/setup-java@v1
     with:
       java-version: '12.x'

   # Setup the flutter environment.
   - uses: subosito/flutter-action@v1
     with:
       channel: 'stable'

   # Get flutter dependencies.
   - run: flutter pub get

   # Check for any formatting issues in the code.
   - run: flutter format --set-exit-if-changed .

   # Statically analyze the Dart code for any errors.
   - run: flutter analyze .

   # Run tests for our flutter project.
   - run: flutter test --coverage

   # Sends test coverage to coveralls
   - name: Coveralls
     uses: coverallsapp/github-action@master
     with:
       github-token: ${{ secrets.GITHUB_TOKEN }}
       flag-name: run-${{ matrix.platform }}
       parallel: true

   # Build apk.
   - run: flutter build apk

   # Build ios
   - name: Run ios build
     if: ${{ matrix.platform == 'macos-latest'}}
     run: flutter build ios --release --no-codesign

   # Upload generated apk to the artifacts.
   - uses: actions/upload-artifact@v1
     with:
       name: release-apk
       path: build/app/outputs/apk/release/app-release.apk
```

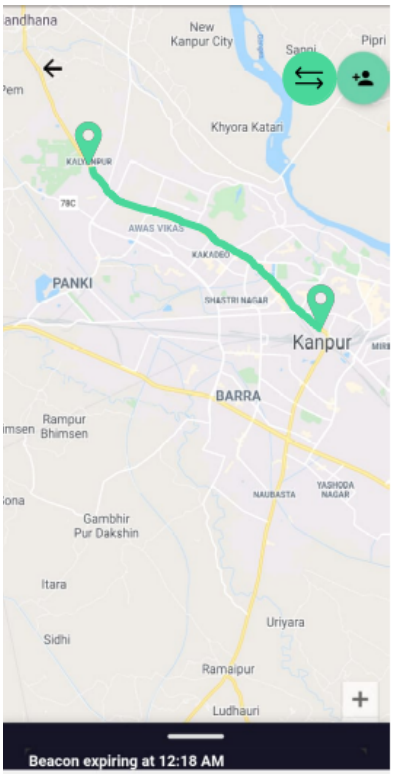
```
finish:
  needs: build
  runs-on: ubuntu-latest
  steps:
  - name: Coveralls
    uses: coverallsapp/github-action@master
    with:
      github-token: ${{ secrets.GITHUB_TOKEN }}
      parallel-finished: true
```

Design Mockups









App Icon Suggestions

(Since these are mockups, changes to the color and some other widgets can be made later)

Proof of Work

Prerequisite Task

I have completed the **Beacon in Flutter** Take-home qualification task giving precise importance to UI Design

Current Features:

- Uses firebase_dynamic_link to share the URL along with just sharing passkey for inviting other users directly to view the location
- Uses geocoder to track the name of the location
- The leader can pass the beacon
- Duration can be updated

Github link: https://github.com/nb9960/beacon
Screenshots and dummy video:
https://drive.google.com/drive/folders/1zYhwJNlAhEQRYRf8nSeksll-DOzwO40Z?usp=sharing

Contributions to ruTorrent-flutter Project

Issues

➔ #74 - [Enhancement] Handle download and add torrent notifications
➔ #103 - [Bug] Undesired behavior of labels

Pull Requests

➔ #98 - [fix] refresh state on rss feed
➔ #85 - [feat] Remove torrent from history
➔ #107 - [fix] all label oriented bugs

Personal Information

More about me

I am a sophomore at the Indian Institute of Technology (BHU), Varanasi. An enthusiastic full-stack developer at my college's premier coding club, Club of Programmers. Along with being proficient in C/C++, Python, Javascript, and Dart, I also have good knowledge of Ubuntu OS and Visual Studio Code.

I like to work, travel, play games (both indoor & outdoor), and listen to music. I enjoy competitive programming, software development, attending hackathons and working on challenging projects. I have been involved in software development and have grown to like open source from the past year, because of its vibrant and engaging community. I actively use open-source software and like to contribute back when I can.

I constantly strive to improve my skills, and find new ways of thinking & problem solving to stay on top of recent technological advancements. I prefer to give more importance to practical implementations rather than theoretical studies. I believe in solving problems by reusing the latest technological stuff, rather than reinventing the wheel. I have been working in a startup for a year now and have come to appreciate the power of teamwork.

Communication

I'm comfortable with any of the communication mediums. I can work full-time on weekdays and am usually available between **11 AM IST (5:30 AM UTC) to 2 AM IST (8:30 PM UTC)**. On weekends, I would love to spend time communicating with the team to learn from them while working on whatever issues occur at that time.

I'll also responsibly keep my mentor updated in case of any emergency that occurs with relevant details.

Post GSoC

If there are things left unimplemented (like documenting the codebase or testing or anything other feature), I would strive to complete them post-GSoC and will keep contributing to the enhancement of this project.

Thanks