GSoC 2022 Proposal

CCEXTRACTOR DEVELOPMENT

Beacon (Frontend)

Aditya Kumar Singh

BASIC INFORMATION

Name	Aditya Kumar Singh
University	Indian Institute of Technology, Varanasi
Degree	Bachelor of Technology (B.Tech)
Year	Sophomore
Timezone	Indian Standard Time (IST) [UTC +5:30]
Email	adityakksingh23@gmail.com
GitHub	ItsAdityaKSingh
Resume	Link to file
LinkedIn	https://www.linkedin.com/in/itsadityaksingh/
Postal Address	201, S-Block, Prakriti Vihar, Kadma, Jamshedpur, Jharkhand, India - 831005
Contact Number	(+91) 9507510924

PROJECT

ABSTRACT

Beacon seeks to create a native flutter interface to make group-travel (or say hiking) easier. By using this, the group leader or leaders would be able to share their location with the entire crew while leading any hike. And in case someone loses contact with the group, one can quickly get on the right path or location by following the location shared by the leader.

This proposal aims to enhance the existing project by adding on some new features like the 'N' beacon in 1 group, maintaining the state of the app, the option to change the details of the beacon, relaying beacons, and various other features for a user-friendly and seamless app experience. Along with the additional features, optimization of the app for best performance and stability is of utmost importance.

MOTIVATION

Flutter, due to its easily accessible documentation and resources, and a fast-growing community of developers, has been of keen interest to me. Since last year I started to learn and develop apps using Flutter and gained more interest in development.

Contributing to Beacon has helped me know more, and the idea behind this project motivates me to keep learning and contribute more to it. Using new packages, following MVVM architecture, and implementing real life cases into user experiences and beyond are various reasons for my interest in this project. This also ensures my total commitment of time and effort to this project.

WHY CCExtractor ?

CCExtractor is a unique organization in itself. The variety of topics it provides and the range of technologies used to implement these are at par, and any contributor or developer would be intrigued to be a part of this society. By joining this wonderful community, I will be able to expand my horizons in terms of learning and applying technologies beyond my current tech stack. Topics such as Machine Learning, Cloud and Server technologies, and other implementations in various software, could all help me get additional expertise and skills.

Furthermore, the quality discussions and support provided on CCExtractor's community channel fascinates and motivates me to explore more stuff. The highly supportive environment and advice provided by the mentors are one of the many essential factors I would like to contribute to this organization.

PROPOSED DELIVERABLES

- Implementing **'N' beacons in one group**, with a **switchable beacon** option (modifying UI to switch amongst beacons seamlessly).
- **Maintaining state** when the app runs in the background for leader, for instance, switches between apps or minimizing the app. This includes fetching updates while maintaining navigation services.
- Running **active** and **joined** hike in **notification** bar while the app is minimized or closed.
- **Syncing** the updates for all and stopping the subscriptions if the beacon has been deactivated in the backend and then changing **hikeScreenModel** to adjust to those changes.
- **Displaying markers** for each follower and leader in a hike.
- Option to **relay a beacon** and change leaders in a hike.
- Option to **preschedule**, **reschedule** or **deactivate** a beacon on the leader's demand.
- Option to **edit** present and future beacons' details for leaders, while an option to **delete** expired beacons for every user.
- A **Website** where new users not having the app installed could head to in case a beacon or link is shared.
- Login having a '**Forgot Password**' option for easy recovery of the account in case the user forgets his/her password.
- Introducing an **Onboarding Screen** for guiding first-time users with the functionalities and running of the app.
- Adding **unit** and **widget** tests for all required views, enhancing **documentation**, and ensuring a **bug-free environment**.

- Making app **Production-ready** for both **iOS** and **Android** platforms with proper end-to-end testing.

BACKGROUND INFO

While contributing to Beacon's Frontend and Backend projects, I understood the codebases and their functionality, allowing me to incorporate new features with the necessary revisions. I have implemented features like Relaying Beacon, Prior Notifications, Countdown Timer, etc. In addition, I have improved and redesigned the UI, corrected bugs, and wrote unit tests for a model in the app. In the PROOF OF WORK section, you'll find a list of all of my other contributions. I've started working on the 'N' beacons feature and the UI part for implementing it. The following parts go into greater detail regarding it.

Technical details:

- Beacon is a cross-platform app with all its screens developed using Flutter.
- It follows a MVVM architecture for implementing Business-model logic in the app.
- It uses GraphQL for authorization, passing queries, receiving APIs, and loading other data processes between server and client.
- It uses MongoDB as a database for storing different data related to the app.

PLAN OF ACTION

The required functionalities can be implemented in one of two ways, as stated on the official Idea List and as decided on Slack:

1. One developer works on both the backend and the frontend on a feature-by-feature basis,
    - If it is decided that a single person would work on both ends of a feature, then the entire model will need to be redesigned and modified for 'N' beacons 1 group feature. The new group model's schemas would have to be implemented first (as determined later in the "*Detailed Plan and Implementation*" section), then all the frontend adjustments would need to be done.
    - Considering the feature of creating a single stream for all beacon modifications, all beacon changes and updates would need to be broadcast through a single stream instead of the current implementation. Following that, the frontend would require one subscription to get all beacon modifications.
    - Preparing a basic landing WebPage for new users, not having the app installed could head to in case a beacon or link is shared.
    - There's no way to recover one's account if one forgets a password. I'll implement this option by making changes and introducing a schema in the backend and then incorporating it in the frontend.
2. Or one developer handling complete back-end stuff (setting up installer and writing tests),
    - Changing models, adding queries and mutations for 'N' beacons 1 group

feature, and other changes for successful implementation on the front-end.
    - Proper Installation script and serverless framework for the running backend.
3. Or one developer handling complete frontend stuff (release, doc, tests, UI improvements, etc.)
    - In the case of two separate developers for frontend and backend, I would begin work on the UI portion of required features such as the **'N' beacon 1 group** functionality, giving the backend developer time to implement such modifications before bringing them to the frontend.
    - Adding a local database to maintain the state while the application cannot connect or fetch data because of an unstable network needs to be done. I would be using the hive database package to store fetched data locally.
    - Moreover, working on the features for changing the details of a beacon, scheduling, and rescheduling, along with creating custom hike reminders, also need to be implemented for more options and feasibility.
    - I have already worked on the frontend implementation for relaying a beacon seamlessly to other devices (followers). It would be fully functional with the new model changes.
    - Also, I will be using the awesome_notifications package for showing an active beacon for users. All features can be easily added after the 'N' beacons 1 group feature is done, as it requires models to be modified.
    - I'd start writing tests for all the models, viewmodels, and widgets when the features were implemented, and the UI was finished to ensure a bug-free environment.
    - After successful testing and eliminating any lingering bugs, the appropriate processes for the app's official release on iOS and Android platforms can begin.

I am willing to work in any direction. Task divisions and operating procedures might be decided during the community bonding period.

TIMELINE

The following is a breakdown of how much time I'll devote to various project elements:

- 60% - Adding major features and implementation
- 15% - Finishing the UI and solving bugs, spawning post-implementation of new features. Also, looking up any issues raised on GitHub and solving them accordingly.
- 10% - Writing unit tests for various classes and widget tests for necessary widgets for making the app bug free and production-ready.
- 15% - Working on the app's formal release on Android and iOS platforms and enhancing the documentation.

COMMUNITY BONDING PERIOD (C.B.P.) MAY 20 - JUNE 12

- I'll discuss how we'll be approaching the tasks: 1 developer per feature or two different developers for backend and frontend.
- I'll discuss the final data model for the 'N' beacon 1 group feature.
- I'll discuss the features in the Ideas List, and after undertaking some suggestions, consider and crosscheck my Plan of Action.
- I'll discuss the final UI and other minor tweaks and desired features that could be additionally implemented if required.
- I'll discuss the expected documentation and testing standards.
- I'll also be discussing the app's release after final implementations and checks.

WEEK 1 JUNE 13 - JUNE 19

- Working on the backend to modify the data model for the required changes in the **'N' beacon 1 group** feature, if decided in the Community Bonding Period.
- Otherwise, I would work on the **UI** part to allow a timeframe as agreed for the back-end implementation of the 'N' beacons 1 group feature.
- Modifying UI for incorporating the new changes.

WEEK 2 JUNE 20 - JUNE 26

- Adding an **Onboarding Screen** for displaying features and introducing apps to new users.
- Changing current local notifications with **custom notifications** for the app using awesome_notifications and providing direct linkage for straightaway joining hikes.

WEEK 3 JUNE 27 - JULY 3

- Redesigning data models for 'N' beacon 1 group feature on frontend according to decided schemas for backend, and as discussed in Community Bonding Period.
- Creating new **queries** for fetching information about groups and displaying them on the home screen. Creating two new **mutations** for creating and joining groups.

WEEK 4 JULY 4 - JULY 10

- Further, connecting backend and frontend for the redesigned model of Groups to show the groups joined by the user.
- Ensuring the buttons '**Create Group**' and '**Join Group**' function properly on the home page with each new Created Group having a unique key.

WEEK 5 JULY 11 - JULY 17

- Introducing a new '**Group**' Screen for showing all particular details of a group. For each group, a unique group page to be loaded with the option to '**create a beacon**' or '**join a beacon**' from a list (scrollable and sorted according to date of creation) of 'existing beacons' in the group.
- Connecting frontend with backend, whenever the backend is ready, or I do the required changes, as decided.
- Ensuring the UI and functionality of both the Home and Group Pages work as intended, that is, the lists load without any error and the mentioned buttons create-join groups and beacons as expected, removing any errors or bugs during the process.

EVALUATION JULY 18 - JULY 24

- Following the major feature's implementation ('N' beacons 1 group, with a redesigned UI), the application would be ready for testing and reviewing.
- Review and discuss with mentors, and improve any errors faced by mentors or left by mistake for further improvements and changes.

WEEK 6 JULY 25 - JULY 31

- Creating a query for getting all created beacons and a redesigned mutation to create and join an existing beacon.
- Joining the frontend with the backend for the changed mutations and queries for a beacon.

WEEK 7 AUG 1 - AUG 7

- Making changes for **pre-scheduling**, **rescheduling,** and **deactivating** beacons on leader's demand, as changed model post-implementation of 'N' group feature would require changes for these features too for proper functioning.
- Introducing '**Pausing**' and '**Terminating**' features for leaders of beacons to choose between pausing or terminating a hike on the '**Hike Screen**'.

WEEK 8 AUG 8 - AUG 14

- Working on the feature of '**Maintaining State**' of the app. Using the Hive database package to create boxes for '**Groups**' and make changes and store useful data while fetching for further use when network connectivity is unavailable.
- This would require changes in the '**Home**' and '**Groups**' screens.

WEEK 9 AUG 15 - AUG 21

- Moreover, modifying UI for adding an option to seamlessly **switch** amongst beacons of a group on the hike screen.
- Finally, testing and eliminating bugs and errors in the implementation of these features and making the project ready for evaluation.

WEEK 10 AUG 22 - AUG 29

- Implementing the feature for '**Displaying Markers for all Users of a beacon**'. This may be done using multiple subscriptions for all the users or a combined single subscription for all (as decided in the Community Bonding Period). Using the location package, this can be achieved.

WEEK 11 AUG 30 - SEP 4

- Combining all the subscriptions into an individual **all-in-one stream subscription** for a beacon.
- Creating a Pop-Up notification for notifying users when the leader **Terminates** the hike or **Pauses** it.

WEEK 12 SEP 5 - SEP 12

- Implementing the feature of **Relaying Beacon**, a part of which I have already completed in a PR. Though this would require changes in both the backend and the frontend after the changed model.
- Ensuring the implementation of the above features and resolving any bug arising. Also, working on the backend for hanging subscriptions if needed or decided.

FINAL EVALUATION SEP 12 - Sep 19

- For the final evaluation, the project would be ready for testing the application in a presentable state for both Android and iOS platforms.
- Review and discuss with mentors, and improve any errors faced by mentors or left by mistake for further improvements and changes.

WEEK 13 SEP 12 - SEP 18

- Keeping leader's and followers' **location data transfer active** even while switching between apps or closed screen until the hike is active using the location package or background_location package.

WEEK 14 SEP 19 - SEP 25

- Implementing the feature for **showing active hikes** on the **notification bar** for all using the awesome_notifications package.
- Testing and debugging the implemented features and reviewing their functionalities.

WEEK 15 SEP 26 - OCT 2

- Adding a **Pop-up** feature for all the users when the hike ends or pauses.
- Reviewing and testing these changes and doing the necessary changes in the backend.

WEEK 16 OCT 3 - OCT 9

- Introducing '**Custom Reminders**' feature for creating reminders based on the user's choice.
- Checking for the implementations to work as expected.

WEEK 17 OCT 10 - OCT 16

- Removing the issue where when a user sends the app to the background and returns, the 'Hike Screen' goes blank.
- Other UI changes like adding '**Animation**' for screens and buttons, beautifying UI, and making it more user-friendly.

WEEK 18 OCT 17 - OCT 23

- Adding changes for '**Forgot Password**' in the backend, as decided or waiting for it to be done, then adding it as a feature on the frontend too on the Login screen.
- Ensuring the new changes work seamlessly and bug-free.

WEEK 19 OCT 24 - OCT 30

- Creating a basic **WebPage** for landing new users not having installed the app, for minimal processes, and guiding them for installing the app.
- Improving **code documentation** for the entire codebase based on the community standards.

WEEK 20 OCT 31 - NOV 6

- Examining the **Flutter Testing** documentation and putting best testing principles into practice. Adding widget tests for critical components and integration tests to cover all major use cases and enhanced performance profiling.
- Ensuring the app is **bug-free** and follows **MVVM** architecture.

WEEK 21 NOV 7 - NOV 13

- Working on the **app's official release** for iOS and Android platforms (Taking a cushion period of 2 weeks for all the processes to complete).

WEEK 22 (ULTIMATE WEEK) NOV 14 - NOV 20

- Focussing on the reviews given by my mentors and discussing with the community what other features could be added to improvise the user experience.
- **Cleaning** up the codebase by **extracting** out repeated code by grouping the unit and widget tests wherever possible.

FINAL SUBMISSION AND EVALUATIONS NOV 21 - NOV 28

- Final app ready with all the required features and changes embedded with a clean and bug-free codebase.
- All unittests, widget tests, and integration tests are passing successfully.
- App ready and tested for launching officially on both iOS and Android platforms.

DETAILED PLAN AND IMPLEMENTATION

MAINTAINING STATE

This can be done using the hive database package. Creating Hive Boxes for the ‘Groups’ model will be needed. (As it already exists for the ‘Beacon’ model). The underlying code defines the code for the purpose.

Creating a local database for storing fetched data

```
import 'package:hive/hive.dart';

Box<Group> groupBox;
Box<Beacon> beaconBox;
Box<User> userBox;

Future<void> init() async{
```

```
 userBox = await Hive.openBox<User>('user');
 beaconBox = await Hive.openBox<Beacon>('beacon');
 groupBox = await Hive.openBox<Group>('group');
}
```

Running app while in background

Current location package plugin used to fetch location also provides an option to fetch location when the app is in the background. We can thus use a stream to listen to changes in location and keep pushing it to the backend while the beacon is active.

```
     await loc.enableBackgroundMode(enable:true);
await loc.changeSettings(interval:3000,distanceFilter:0.0);
_userlocation = loc.onLocationChanged.listen((){});
```

Here, distanceFilter(in metres) can be increased to reduce load over data fetching or app performance.

Additionally, as decided, we could also use the background_location package for continuous background location updates while the app goes into the background. The underlying code gives a hint about using it.

```
     await BackgroundLocation.startLocationService(distanceFilter: 20);
BackgroundLocation.getLocationUpdates((location) {
 setState(() {
   latitude = location.latitude.toString();
   longitude = location.longitude.toString();
   accuracy = location.accuracy.toString();
   altitude = location.altitude.toString();
   bearing = location.bearing.toString();
   speed = location.speed.toString();
   time = DateTime.fromMillisecondsSinceEpoch(
           location.time!.toInt())
       .toString();
 });
```

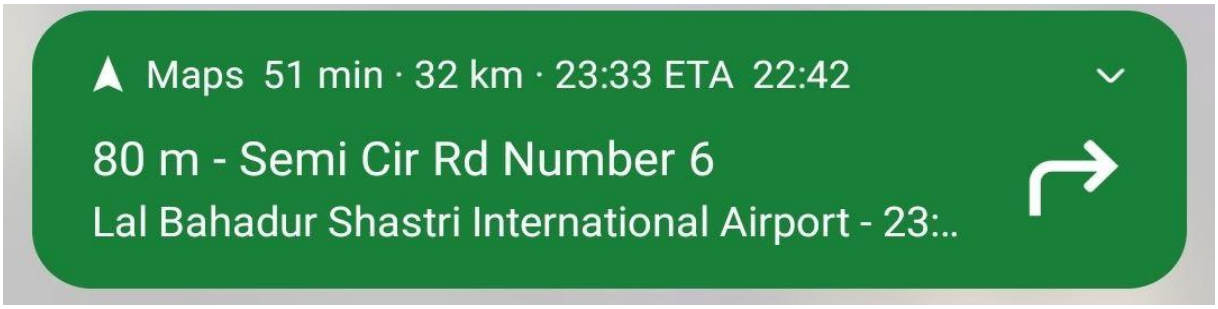
Using `BackgroundLocation.stopLocationService()` we can stop this process.

Showing Active Hikes on Notification Panel

While the app is active, we can show a customised active notification on the Notification Panel using awesome_notifications package. Either we could show a map turn-by-turn navigation using Navigation category or show a Progress Layout notification while the app is in foreground as well as in background, and as decided.

```
    NotificationChannel(
        channelGroupKey: 'progress bar example for a hike',
        icon: 'beacon official icon',
        channelKey: 'progress_bar',
        channelName: 'Progress bar notifications',
        channelDescription: 'Notifications with a progress bar layout',
        defaultColor: Colors.deepPurple,
        ledColor: Colors.deepPurple,
        vibrationPattern: lowVibrationPattern,
        onlyAlertOnce: true),
```

UI to better understand the above feature:



PROPOSED DATA MODEL

In our new model, we could have a **Beacons** list as a field inside our **Groups** model. The beacons subfield would then contain all the information needed. A different subfield for **group_joinee** list would need to be added. And for each beacon (as exists) a **beacon_joinee** list. Moreover, the User model must have two additional fields of **groups_joined_by_user** list along with **beacons_joined_by_user** list.

Proposed Data Model for Group:

```
{
 "group_name":"Name of the Group",
 "group_id": "Unique ID for each group",
 "group_passkey":"Unique key for joining each group",
 // Unique key for joining a particular group
 "beacons":["Iterable list model having beacon model as mentioned below "]
 // Contains all the beacons under the group
}
```

Proposed Data Model for Beacon:

```
{
 "group_id": "Unique ID for each group",
 //For recognizing beacon as belonging to a unique group
 "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
          {

           "username": "unique username of each joinee",
           // unique username for each user and by default user_id
       would be stored in place of username if user wants to stay anonymous
           "lat": "latitude of user's location",
           "long": "longitude of user's location",
           },
 .
 .
 .
 //All the existing fields of joinee in Beacon model
 ],
}
```

Proposed User Model for Beacon:

```
{
 "name":"name of user",
 "email":"email of user",
 "groups":["List of all the groups user is in"],
 "beacons":["List of all the beacons user is in"]
 .
 .
 .
 //All the existing fields of User model
}
```

Proposed Mutation for Creating Group:

```
String createGroup(String name){
 return '''
     mutation{
       createGroup(name: "$name"){
         group_name
         group_passkey
```

```
        }
      }
  ''';
}
```

As the new group contains no beacons or users, it can be created with a name and then function as per requirements.

Proposed Mutation for Joining Group:

```
String joinGroup(String name){
 return '''
     mutation{
       joinGroup(shortcode: "$shortcode){
         group_name
         beacons{
           //...Individual beacon data
         }
       }
     }
  ''';
}
```

Gives back the group_name and a list of all the beacons in the group.

Proposed Query for Fetching Groups of User

```
String fetchUserInfo(){
 return '''
     query{
       me{
         groups{
           group_name
           group_id
         }
         beacons{
           //...Individual beacon data
         }
       }
     }
  ''';
}
```

OTHER PROPOSED CHANGES

Displaying Markers for all Users present in a Beacon

This can be achieved by using the current location tracking process for all users. (currently only implemented for the leader)

Example:

```
for(var user in active_beacon.users){
   markers.add(Marker(
     markerId:MarkerId((markers.length)+1).toString(),
     infoWindow:InfoWindow(
       title:"${user.name}",
     ),
     position:LatLng(
       double.parse(user.location.lat),double.parse(user.location.lon)
     ),
     icon:BitmapDescriptor.defaultMarkerWithHue(
       BitmapDescriptor.hueBlue
     ),
   )
 );
}
```

Pre-Scheduling, Rescheduling and Deactivation of Beacon

Existing implementation needs some minute changes for these features to follow. For Pre-Scheduling, we need to pass a startsAt mutation with future time data. For Deactivation, we can directly pass DateTime.now( ) as a expiresAt mutation for immediate deactivation. Additionally, we could add an Edit option to give the option for rescheduling a Pre-Scheduled beacon and extend the expiresAt for an active beacon.

Creating a WebPage for landing new Users with no app

In case of new Users with no app but a link to a group or beacon, they can be guided to a WebPage having details regarding app installation and guidance for the app store. Basic features like creating an account or logging into one, showing a few details of the group or beacon could be added here.

Redesigning existing UI to incorporate new changes

This would entail renaming '**Your Beacons**' and 'Nearby Beacons' to '**Your Groups**' and '**Nearby Groups**', respectively. As a result, the user would see the joined groups and nearby groups. Also renaming '**Create Hike**' and '**Join Hike**' to '**Create Group**' and '**Join Group**' respectively. Additionally, bringing in animations for opening screens or buttons and an Onboarding Screen for new Users to get a sense of the app's working.

Tests and Official Release

Tests would be written following the official Flutter Testing Docs. Starting with unittests for models and view_models, widget tests for necessary widgets and views, and finally, integration tests for performance mapping of our app.

Further, post successful testing and a production-ready environment, working for the app's official release for both iOS and Android platforms could be initiated. I would be using the official Flutter Android and iOS release docs for reference.

PROOF OF WORK

CONTRIBUTIONS TO BEACON PROJECT

Issues:

Reported Issues

- #58 - An about page and onboarding screen
- #80 - Multiple and unsure errors by dialog boxes
- #82 - Cannot create hike immediately
- #91 - Feature for refreshing beacons
- #103 - Feature: Adding countdown for hike screen and exit pop up box upon ending
- #107 - Feature: Delete created beacons or edit start/end time and duration
- #120 - Beacon reloads each time State is updated
- #123 - Feature: Providing option to terminate or pause a hike while leader exits
- #130 - Alert box for followers when a leader pauses, exits or ends a hike

Pull Requests:

Closed

- #24 - Updated Files and Dependencies
- #45 - Updated UI, fixed bugs and refactored code
- #50 - Templates Introduced
- #61 - Exit Popbox and Rounded-Borders
- #63 - UI changes
- #64 - Added validator for Name field
- #66 - Eye Icon changed
- #68 - Changed beacon card sizing
- #72 - Multiple error messages resolved
- #74 - Fixed sliding corners and sliding to minheight
- #77 - Reversed list according to recent
- #85 - Changed card colors, added status for different beacons + feat:countdown timer for pre-scheduled beacons
- #86 - Passkey Capitalisation
- #88 - Improved hike screen UI

- #89 - Updated and sorted dependencies
- #93 - Reduced font size and changed themes
- #96 - feat: Close keyboard when text field is out of focus
- #99 - feat: nearby beacon list according to start time
- #101 - Snackbar functionality added and general UI improved
- #105 - Increased distance between buttons for smaller screen sizes
- #112 - Added reminder for a hike about to start in an hour
- #119 - Updated compilesdkVersion to support Flutter 2.10 + updated dependencies and pubspec files
- #125 - Build error when tabs are switched immediately after timer hits 0
- #146 - Fix: removed deprecated splash warnings

Open

- #83 - Create Hike immediately
- #115 - Added location_model_test
- #122 - Feature: Relay beacon to new leader
- #157 - Fixes: Multiple list reloads whenever TextField is focussed

CONTRIBUTIONS TO OTHER CCExtractor PROJECTS

beacon-backend:

- #99-Feature: Edit Start Time of beacon
- #100-Feature: Added Delete Beacon mutation

ccextractorfluttergui:

- #23-Output Settings Browse widget size fixed
- #24-CommandWidget overflowing on small screens fixed
- #26-Dashboard State fixed
- #30-Added Scroll Controller
- #31-Changed sizing of drop down

Deluge-mobile-remote-client:

- #39-Improved documentation
- #44-Added .gitignore
- #45-Added CI actions for testing PRs
- #47-Added templates
- #49-Formatted Files according to flutter norms
- #58-Removed some more unwanted files + Added more options in .gitignore file

PERSONAL INFORMATION

MORE ABOUT ME

I am Aditya Kumar Singh, a sophomore pursuing a Bachelor of Technology at the Indian Institute of Technology (BHU) in Varanasi. Since I was in high school, I've had a strong desire to learn more about different aspects of Computer Science. Data Structures and Algorithms, Software Development, Machine Learning, and Information and Security are some of the topics that pique my attention. I'm familiar with GitHub and Visual Studio Code and proficient in C/C++, Python, and Dart. I've enjoyed and contributed to Open-Source for some time now, which has allowed me to expand my knowledge and hone my skills with the support of its dynamic and active community.

Apart from Open-Source, I enjoy competitive programming, attending hackathons, and working on challenging projects. As a person, I am a self-motivated individual. I am a quick learner with solid grasping abilities. I am capable of working independently and as part of a team. To keep on top of contemporary technical breakthroughs, I continually seek to improve my skills and find new thinking and problem-solving methods. I keep my mind open to anything that needs to be learned for the organization's and my growth.

Practical implementations, rather than theoretical studies, are more important to me. Rather than reusing old conventional methods, I believe in solving problems by utilizing the most recent technological advancements.

Aside from the technical side of things, I enjoy traveling and exploring, playing games (both physical and esports), and watching football.

COMMUNICATION

Timezone: IST (UTC +5:30)

Any communication methods are acceptable to me. I am available full-time between 11 a.m. IST (5:30 a.m. UTC) and 1 a.m. IST (7:30 PM UTC) on weekdays. Time can be flexible depending on what the mentors decide and are suitable for all.

On weekends, I'd like to interact with the team to learn from them while working on whatever problems arise.

I'll also be responsible for keeping my mentor up to speed with pertinent information in an emergency.

POST GSOC

If there are any remaining tasks such as code documentation, testing, or any other feature implementation, I will work to fulfill them after the GSoC. I will continue to contribute to the

project's improvement and keep the development environment running.
Regardless of GSoC, I would love to engage in discussions with the CCExtractor community to get exposure to new technologies and ideas. I would love to be of any help even after the GSoC period.

Thank You!