GitHub Username: SJagannath

THE UPWARD THUMB

Table of contents

Description

This app primarily aims to help people who require transportation from colleagues/acquaintances who share the same office or route.

During rush hours, oftentimes cabs are not available, and yet we have several people following the same route, going to the same (or close by) destination. I have often wished that there would be some way for me to figure out which of my colleagues are in the same route so that I could get a lift from wherever I am. Usually this is when there is lack of last mile connectivity, and the distance to walk is too great, or that we are in a hurry. This app aims to solve the problem by sending out requests to people in the contacts list of the user, when the user requires a lift, and any recipient of the request can know where exactly on the route the user is waiting, and issue a response if he/she chooses to give the user a lift. Since the app uses contacts from the user's contact list, it does not expect to send requests out to untrusted people. This also reduces the user's dependence on cabs, if the user regularly requires a lift.

Intended User

- Frequent public transport user with last mile connectivity problems
- People who drive alone, and would not mind helping their colleagues/friends out

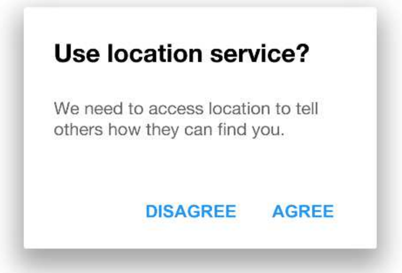
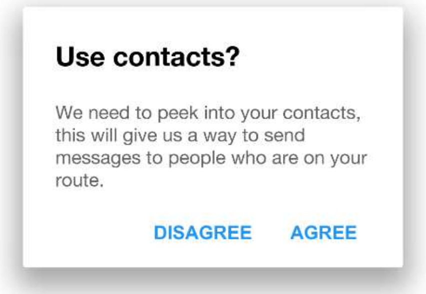
Features

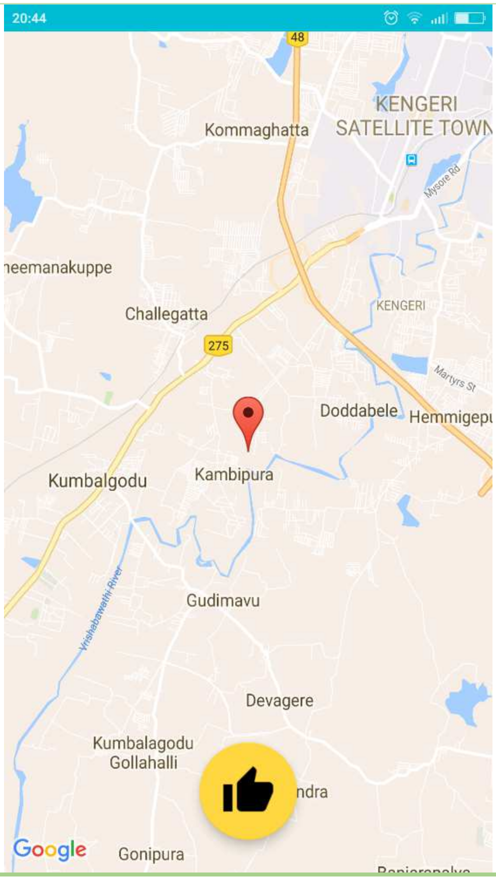
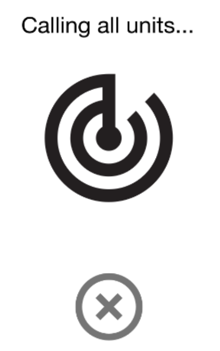
List the main features of your app. For example:

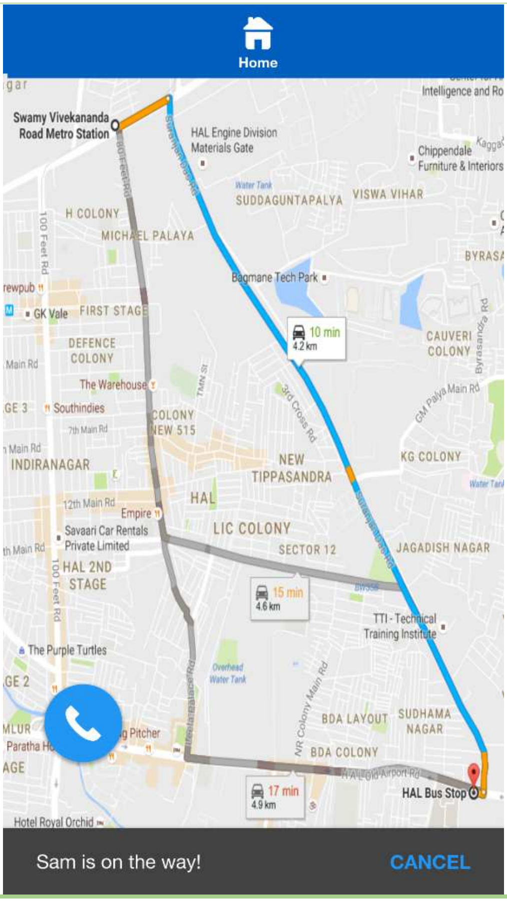
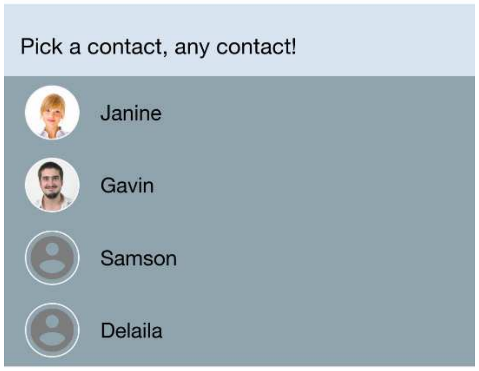
- Uses GPS location
- Uses SMS to send messages (with location) to contacts
- Maps to try and locate the sender & recipient.

User Interface Mocks

Screen name	UI	Description
Splash screen	The upward thumb Find your ride	**The user sees this first, it is the branding/logo screen to introduce the app and it's catchline "The upward thumb: find your ride"**

Permission to use Location services	 Use location service? We need to access location to tell others how they can find you. DISAGREE AGREE	**We require a way to broadcast the user's location to his contacts, in the event that he/she needs a ride, so we request user's permission to turn on GPS as required.**
Permission to use Contacts	 Use contacts? We need to peek into your contacts, this will give us a way to send messages to people who are on your route. DISAGREE AGREE	**We require a way to contact the people on the user's list via the internet or through SMS in the absence of internet.**

Main activity	 20:44 KENGERI SATELLITE TOWN Kommaghatta Challegatta Doddabele Kambipura Kumbalgodu Gudimavu Devagere Kumbalagodu Gollahalli Gonipura Google	**The main screen; tapping the thumb symbol will launch either the setup contacts activity (if there are no pre-chosen contacts to send the ride requests to), or a chooser for a specific group to broadcast the request to. The chooser also has an option to setup a new group.**
Hitch is live		**This is an intermediate screen, between when the user requests a lift, and the app is trying to find him a ride.**

Ride found	 Sam is on the way! CANCEL	**On a user accepting to give a lift to the requesting party, this is the map-based screen that shows the user where the lift-giver is, and how long he will take to arrive. It also contains a way for the user to cancel his request, and call his/her ride if required. Also contains a link to return to the main activity**
Add contacts	 Pick a contact, any contact! Janine Gavin Samson Delaila	**This is where the app accesses the user's contacts list, and provides a way for the user to add people from whom to request a lift**

Key Considerations

How will your app handle data persistence?

- App will use a content provider to store contact URI's that are to be included in the main activity screen. It will store contacts URI grouped per contact set requested by the user
- App will connect to the Contacts content provider using a Loader to fetch & display a list of contacts that the user can pick a subset from
- App will use an AsyncTask to read/write selected contact information & user's current location, and send SMS call in the background

Describe any corner cases in the UX.

App intends to use fragments (replace and add judiciously) and make use of the framework's ability to manage fragment stack to control UI flow. The app may not need to use Home navigation, but there is a provision to use it if necessary. App may need to handle the events when user denies the usage of contacts or location

Describe any libraries you'll be using and share your reasoning for including them.

- App might use Picasso/Glide to load contact images into the list view on main screen

Describe how you will implement Google Play Services.

- App will use maps to show location/direction

Next Steps: Required Tasks

Task 1: Project Setup

- Configure Picasso/Glide libraries for usage
- Configure Maps console to show on mapview.

Task 2: Implement UI for Each Activity and Fragment phase 1

- Implement UI for Location & contacts access request.
- Listview with animated thumb for main activity. Might use Controller layout to fold the toolbar with logo
- Mapview with pins, source, destination & distance/time
- Snackbar with action to cancel
- FAB for implementing phone call feature
- Home navigation (might not be required)
- Listview with images for contacts
- Animated "searching" screen with cancel option

Task 3: Required logic & UI for permissions

- Implement UI for splash and permission screens
- Verify app can access location & contacts, and handle use cases when user denies one or both permissions

Task 4: Implement UI for Each Activity and Fragment phase 2

- Implement the UI for contacts list

- Stub for: requesting a ride
    - cancellation
    - addition of contacts
    - Ride found screen
- Verify navigation between screens

Task 5: Business logic implementation

- Integrate SMS with app and verify working
- Fulfill stub implementation for ride request via SMS

Task 6: Location based logic

- Integrate Maps API with app and verify working
- Simulate ride found screen with mock locations and verify working

Task 7: Testing, review & rework

- Test for corner cases
- Test with real world users
- Review code and rework as necessary

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"