How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

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**"

GitHub Username: alexdennis

Deets

Description

Making business cards obsolete. Deets allows you to seamlessly exchange contact details electronically with the people you meet at conferences, meetups, networking events. No longer do you have to worry about running out of business cards or remember to input contact details from business cards you collect. Simply open the app and tap a button to share your contact info with people nearby and keep a history of all the people you've met with details on when and where you met them.

Intended User

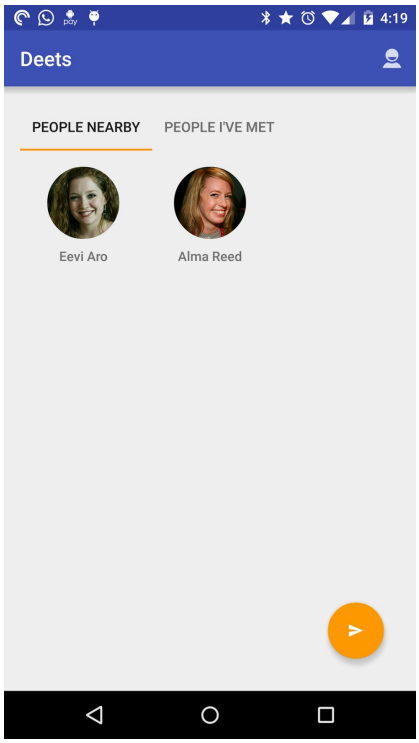
Everyone who is interested in meeting new people. This app is particularly useful for people who go to networking events and meet a lot of people at once.

Features

- Share contact details, (phone number, website, e-mail address, etc.) with people nearby
- Keep a record of all the people you've met
- Auto import contacts met into phone contact database

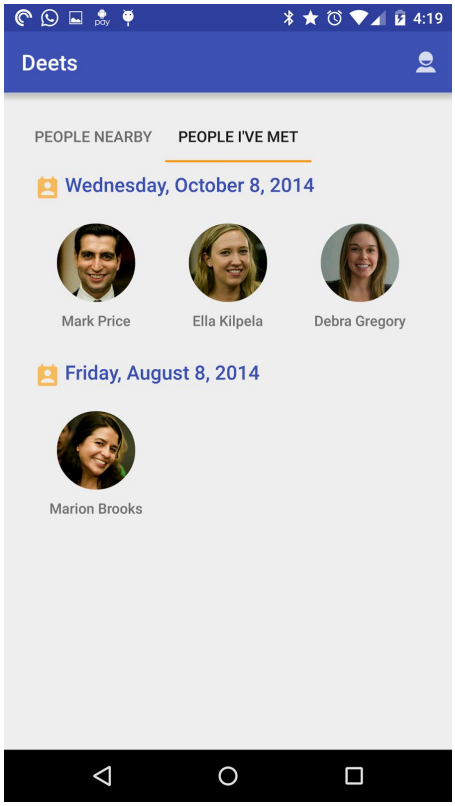
User Interface Mocks

Screen 1: People Nearby Tab



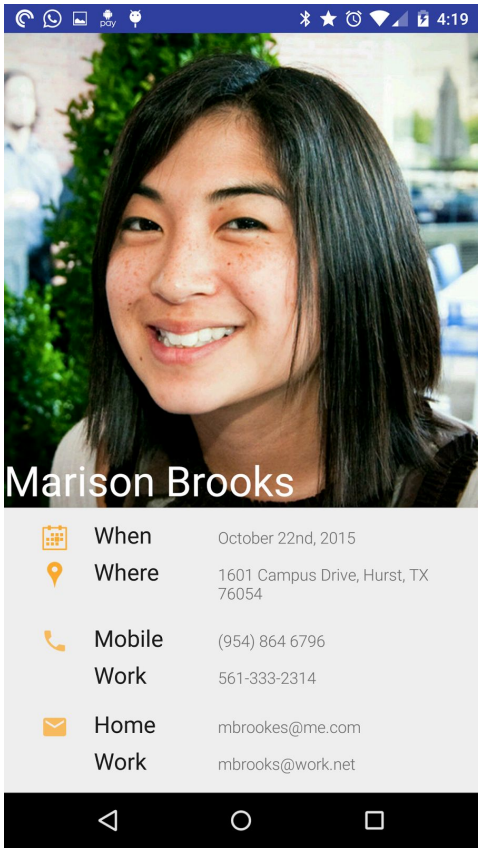
On the main screen the user will see the people nearby who are also on the same screen and then can select the person who they want to share their contact details with. When the user selects one or more persons they will be highlighted as being selected and then a quick tap on the FAB will push contact information to the persons selected.

Screen 2: History Tab



On the second tab of the main screen is a history view of all the people met in chronological order. Selecting a person will bring up their details including the date and place when they were met.

Screen 3: Contact Details Screen



1. The contact details screen will display contact information of a person that appears in the history of people met.
2. This screen will also be used to display contact information that the primary user is sharing with others which can be accessed from the profile menu bar action item on the main screens.

Screen 4: Wear companion app

From attached wearable user can start nearby search which will show searching screen and then populate with a list of people found nearby. User can swipe left through the list or tap a person and get a confirmation to tap a second time and share contact information with person without taking out their phone.

Key Considerations

How will your app handle data persistence?

1. Contact history and details will be stored in local database which will be access through a custom content provider
2. Contact details will also be optionally exported automatically to Android contacts provider. Auto export will be a settings option
3. Contact detail information that is being shared by the main user of the app can be imported from Android contact provider and then stored within the app database

Describe any corner cases in the UX.

1. If the user sends contact information to share with another person, it is possible that person does not also share contact information back. The person will show in the timeline with an photo and name, but on the contact details screen the only additional information that will be displayed is when and where the person was met. There will be a FAB button to request contact details later. This will show up as a notification on the receiver side to prompt them to send that information along with the context of photo/name/when/where the requester was encountered.

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

1. Glide for image loading
2. Google Play Services Nearby API to allow device to device close proximity communication
3. Android Design Support Library for material design widgets/controls
4. Gson for encoding and decoding messages in json
5. Google Play Services Wearable API for Wear companion app
6. Circle Image View - https://github.com/hdodenhof/CircleImageView to display avatars in a circle
7. Google Cloud Messaging

Next Steps: Required Tasks

Task 1: Project Setup

1. Create new project in Android Studio with following information, everything else can be defaults
    a. Application name: Deets
    b. Package name: com.carltondennis.deets
    c. Select form factors:
        i. "Phone and Tablet" with Minimum SDK 16
        ii. "Wear" with Minimum SDK 21
    d. Blank Activity with Fragment
    e. Blank Wear Activity
2. Add Library dependencies to app build.gradle
3. Add Wearable permissions to main app Android Manifest
4. Register package in Google Developers console and add API KEY to main app manifest as described on https://developers.google.com/nearby/messages/android/get-started
5. Register for Google Cloud Messaging and configure API information

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for NearbyFragment
- Build UI for HistoryFragment
- Build UI for ContactFragment
- Build UI for SettingsActivity

- Build UI for Wear Round MainActivity
- Build UI for Wear Square MainActivity

Task 3: Build database and content provider

- Create DB helper with database definition for contacts and history
- Create Content provider to access timeline and contact info
- Implement app initialization sequence that populates the DB with primary contact information (Primary contact is the information that gets shared with other people)

Task 4: Create App Engine application to store/register contact unique id

- Cloud database for storing unique id for each person using the app that gets assigned at startup, this could be Google Auth ID, as well as user avatar and name. This is so that we can pass URLs in Nearby Message instead of trying to send full photo info which would not fit. Per documenation, Nearby is designed for messages less than 3KB (https://developers.google.com/nearby/messages/android/pub-sub)

Task 5: Implement Nearby API to send/receive contact information

- Create animation that displays when nearby is activated
- Use Nearby API to find nearby devices and send contact messages that contain contact name and URL for contact avatar.
- Allow user to share full contact information when share button clicked via Nearby API
- Store received contact information in contact database and import into local contact provider

Task 6: Implement Contact history and Contact details

- Load history into history tab using Content provider
- Load contact information in to contact details screen

Task 7: Use GCM to send notifications

- Use GCM to allow user to request contact details from people who they have shared with but who have not shared with them.

Task 8: Wearable companion app

- Allow nearby search to be triggered from wearable. All of the work will be done on the mobile device and then sent to the wearable using the Wear Messages API

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**"