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: **nvalletta**

Cedar Bark Grooming

Description

Looking for a qualified pet groomer in southern Utah? Look no further! The owner of Cedar Bark Grooming has been in the pet grooming business for over 25 years and has groomed a wide variety of dogs, ranging from 80-pound Rottweilers to 10-pound Yorkshire Terriers. Download the app to browse before and after photos, locate Cedar Bark Grooming on the map, and learn more about one of Cedar City's most experienced pet groomers!

Intended User

Dog and cat owners who are interested in getting their dog or cat groomed at Cedar Bark Grooming, or interested in learning more about the business (where they're located, browse before and after photos, etc.).
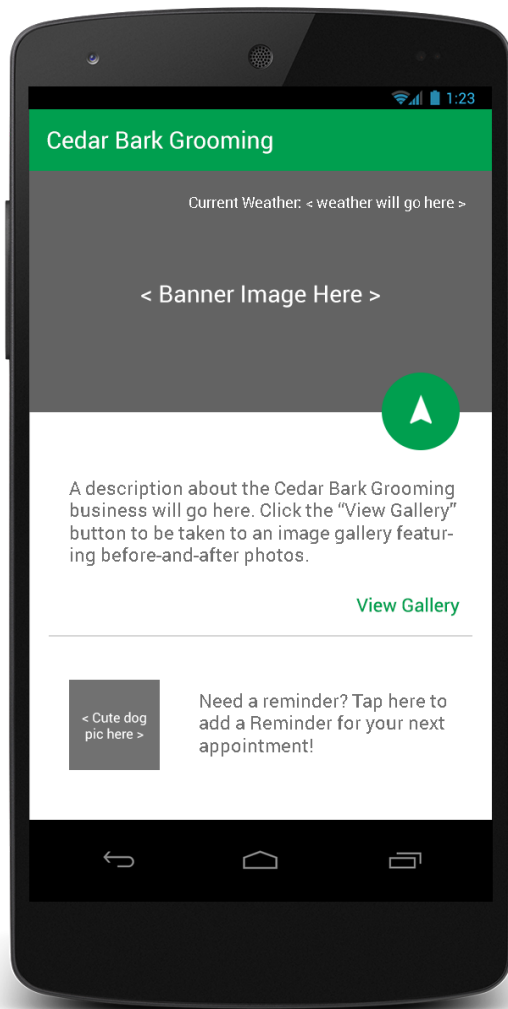
Features

List the main features of your app. For example:
- Displays images and information
- Uses the Location API to discover a user's location in order to calculate distance to Cedar Bark Grooming
- Uses Google Maps API to navigate from user's location to Cedar Bark Grooming
- Pulls weather information via OpenWeather to let users know what the current weather is at Cedar Bark Grooming
- Users can add a Reminder to make their next appointment (the user will be notified via Notification once the appropriate time has arrived)

User Interface Mocks

Screen 1

This is the Home screen, which will feature the following:
- The current weather and user's distance to Cedar Bark Grooming in the upper left
- A lovely banner depicting Cedar Bark Grooming
- A navigate button which, when tapped, will launch Google Maps with Cedar Bark Grooming's location for navigation purposes
- A "view gallery" button beneath the description which will open an image gallery
- Tapping the bottom view will launch the RemindersActivity, where the user can add, edit, or delete appointment reminders.
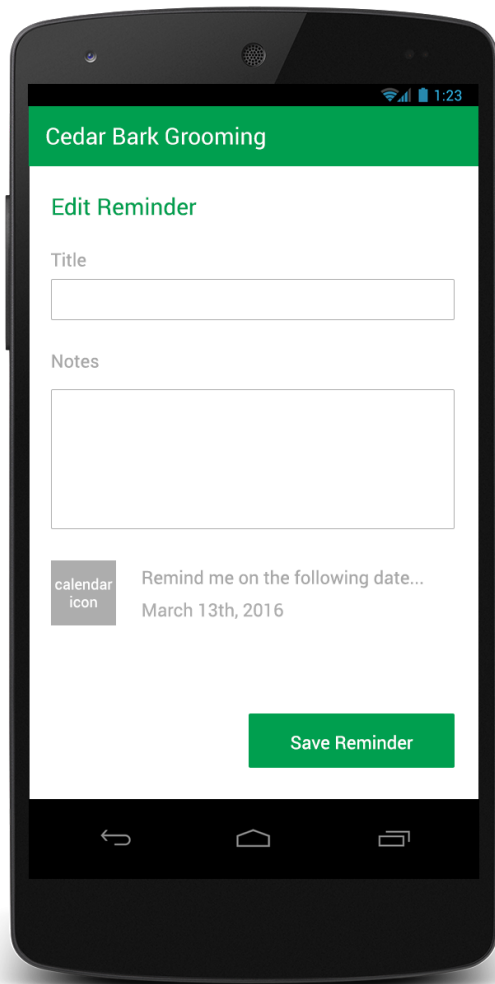
Screen 2
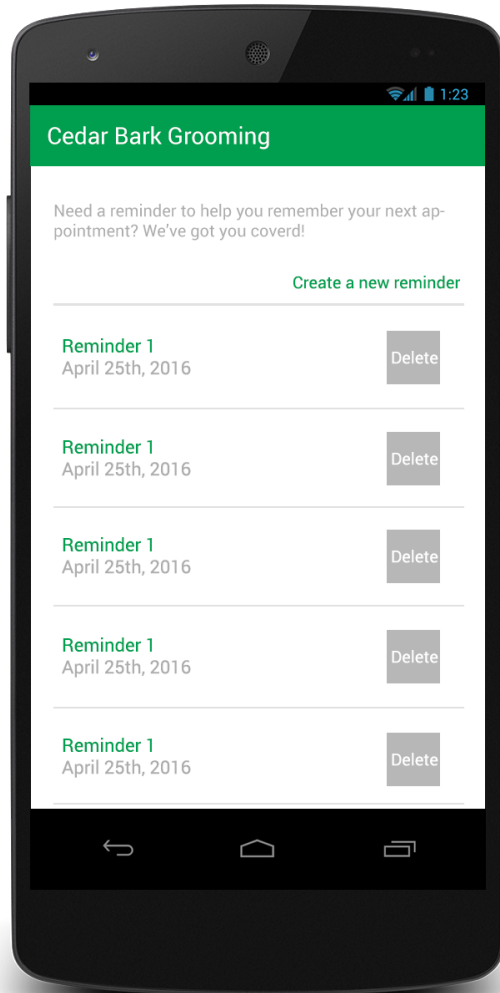
This is the Reminders screen, which will feature the following:
- A list of current reminders, complete with the date of the reminder and the ability to delete a reminder by tapping the icon to the right.
- A way for users to add new reminders by clicking "Create a new reminder"

Screen 3

This is the Edit Reminders screen, which allows users to save the selected reminder (or add a new one if the reminder is brand new). This screen asks for the following fields:
- A title: the user can enter a title for their reminder (optional)
- Notes: the user can enter notes for their reminder, such as: "make another dog grooming apopintment!"
- Date: tapping the view that contains the calendar icon and date information will allow the user to enter the date of the reminder. This field is required.

Screen 4

This is the Image Gallery, which will feature a scrollable collection of Cedar Bark Grooming images. Tapping an image will open a more detailed image view. (see screen 5)

Screen 5

Cedar Bark Grooming

< A larger, more detailed version of the
thumbnail chosen on the gallery screen. >
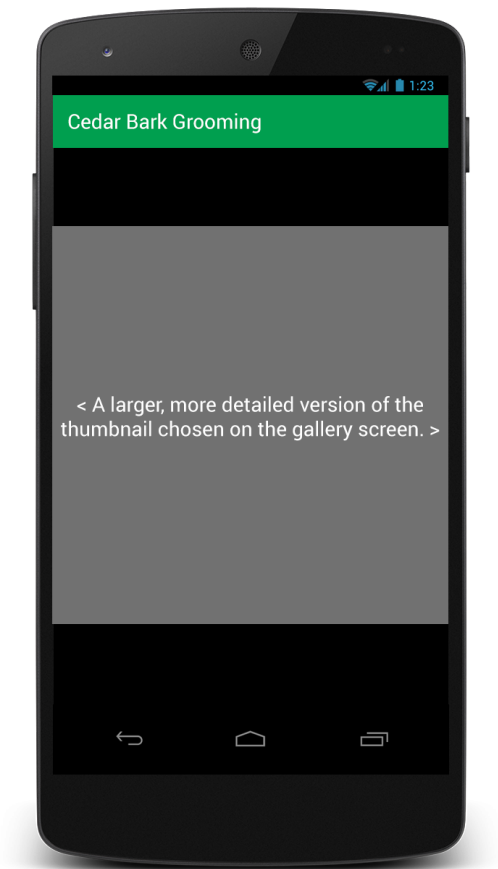
This is the detailed image screen, where the user can view a larger version of the image
thumbnail they tapped to get here. (from screen 4)

Key Considerations

How will your app handle data persistence?

The app will build a custom ContentProvider to persist information about the user's upcoming Reminders.

Describe any corner cases in the UX.

The user can easily back out of any screen using the system "Back" button. To avoid disruption, the app will primarily make use of SnackBars, only resorting to Dialogs when necessary.

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

- **Picasso** will handle the loading and caching of images
  - https://github.com/square/picasso
- **ButterKnife** will handle View binding to cut down on boilerplate code
  - http://jakewharton.github.io/butterknife/
- **IcePick** will handle saving and restoring instance state
  - https://github.com/frankiesardo/icepick
- **Dagger 2** will handle dependency injection
  - http://google.github.io/dagger/
- **RxJava / RxAndroid** will assist with composing asynchronous events
  - https://github.com/ReactiveX/RxJava
  - https://github.com/ReactiveX/RxAndroid
- **Gradle-RetroLambda** will give the project lambda support in Java versions < 8
  - https://github.com/evant/gradle-retrolambda

Next Steps: Required Tasks

Task 1: Project Setup

- Add all libraries/dependencies
- Add a blank activity and XML layout for each screen (MainActivity, GalleryActivity, RemindersActivity, ReminderEditorActivity)
- Add a blank fragment called ImageFragment for viewing a detailed Gallery images
- Ensure that project exists in source control (Github). Make the initial commit!

Task 2: Implement UI for Each Activity and Fragment

- Build XML layout for MainActivity
- Build XML layout for GalleryActivity
- Build XML layout for RemindersActivity
- Build XML layout for ReminderEditorActivity
- Build XML layout for ImageFragment

Task 3: Utilize Google Services & Retrieve Data

- Add code to discover the user's current latitude and longitude using Location API
- Add code to discover Cedar Bark Grooming's latitude and longitude using Location API
- Add code that will launch intent to open Google Maps to navigate to Cedar Bark Grooming from the user's current location

Task 4: Application Logic

- Set up the ContentProvider that will allow the app to persist Reminders
- Add code to enable the adding of new Reminders
- Add code to enable users to edit existing Reminders
- Add code to determine the distance between the user's current location and the location of Cedar Bark Grooming
- Add bitmap resources for the image gallery, and use Picasso to load each bitmap into a section of the grid.
- Add code that will open ImageFragment with a more detailed view of the selected image

Task 5: Bind Data to Views

- Bind data to the view that will display the user's distance to Cedar Bark Grooming on the MainActivity above weather information on the banner.
- Bind data to the view that will display weather information.
- Add a listener to the navigation button that will execute the code to launch Google Maps

---

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**"