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: bradzacher

Rain Notifier

Description

Every morning you'll receive a notification with the chance of rain and when.

Within the app you can review radar imagery and see hourly forecast graphs to figure out if you'll be getting wet today.

Intended User

Anyone who owns an outside.
People who have to do things outside each day.
People who ride bikes and want to know if they need wet weather gear.

Features

The problem is that I weather app just gives a whole day “it’s going to rain” or it’s not, and because it’s aggregated for the entire city, it’s usually wrong as well!

This app will specifically focus on the chance rain and try to give as location specific data as possible.

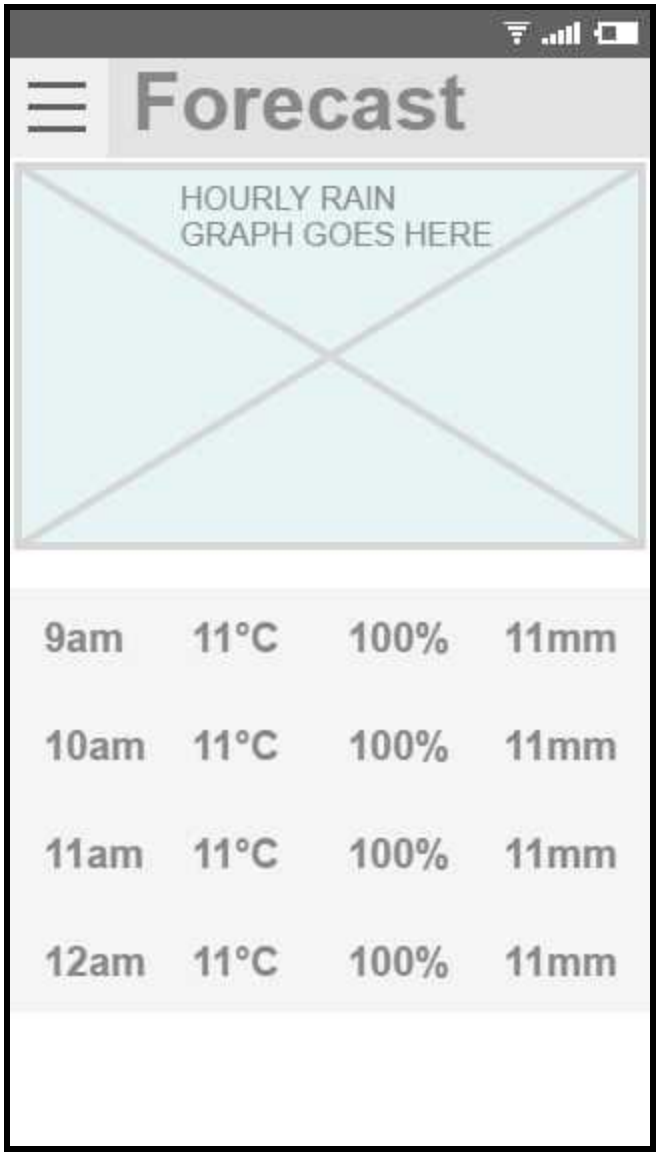
Currently looking at using weather.com’s api (https://www.wunderground.com/weather/api/d/docs).

- Fetches the user’s location - allows the user to change it if they want.
- Displays radar imagery on a map via a weather api.
- Displays hourly forecast graphs for the chance and amount of rain.
- Displays a notification every morning at a configurable time with details about the day’s rain

User Interface Mocks

These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1 - Main Forecast Screen

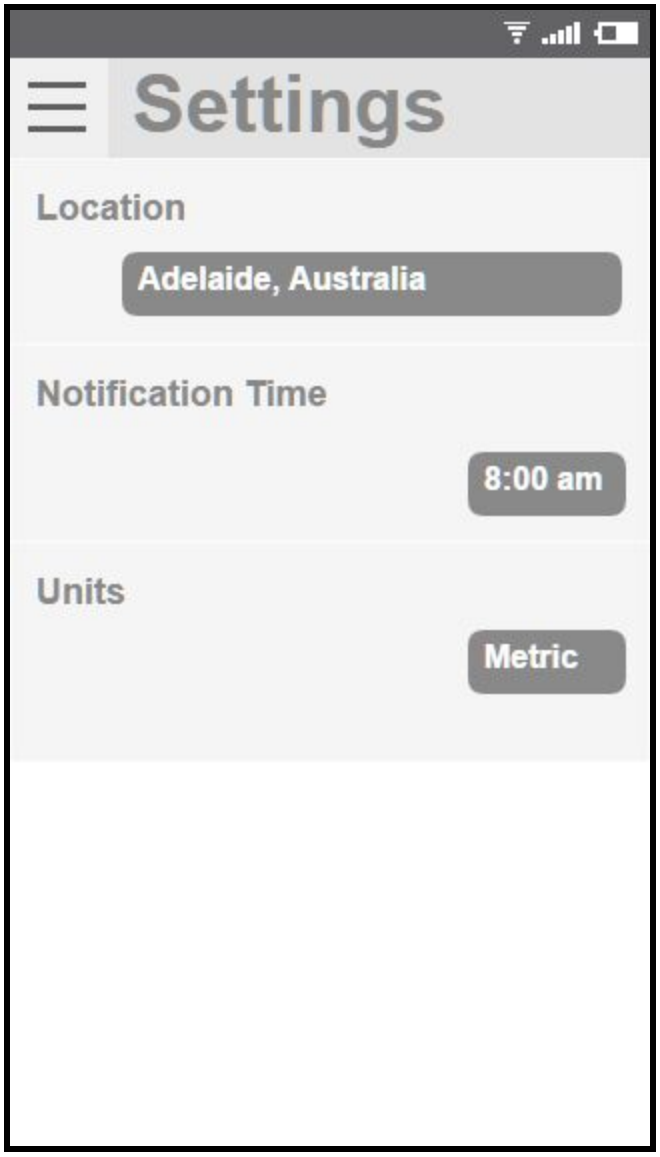


Shows a graph of the hourly rainfall chance as well as the rainfaill ammount.
Also shows data in a tabular form. The table will hide times in the past, with the most recent hour being at the top.
Additional screens can be accessed via an offscreen menu opened using the hamburger button.

Screen 2 - Radar Screen

Shows the precipitation radar.
User can fully interact with the map.
Map starts centered on the user's GPS location.

Screen 3 - Settings Screen



Allows the user to set app preferences.
Location will bring up a search screen for locations.
Notification time is set using the android time picker.
Units is set using a select list.

Key Considerations

How will your app handle data persistence?

Data will be pulled from the server and cached locally using HttpResponseCache.

Settings will be stored using SharedPreferences.

Describe any corner cases in the UX.

N/A

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

Butter knife because it makes UI dev work simpler.
AndroidAnnotations because it makes coding simpler.

Describe how you will implement Google Play Services.

Maps to display the radar imagery.
Location to determine the user's location to customise weather data accordingly.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

- Get API key for weather.com's api.
- Configure project in android studio.
- Configure libraries.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for the hourly forecast screen
- Build UI for the map screen
- Build UI for the settings screen

Task 3: Implement Google Play Services

- Implement location services.
- Implement maps.

Task 4: Settings!!

- Implement SharedPrefs
- Load defaults into prefs
- Store location in prefs
- Store notification time into prefs
- Store units into prefs

Task 5: Create API client

- Determine relevant api calls to wrap.
- Wrap using AndroidAnnotations.
- Implement HttpResponseCache.
- Fetch data based off settings.
- Overlay radar on map.
- Load data into forecast tables.

Task 6: Implement local notifications

- Implement AlarmManager.
- Read from API on alarm firing.
- Create local notification on based off API data.
- Schedule alarm based off user preference.

Add as many tasks as you need to complete your app.

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**"