

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**”

[Description](#)

[Intended User](#)

[Features](#)

[User Interface Mocks](#)

[Screen 1](#)

[Screen 2](#)

[Key Considerations](#)

[How will your app handle data persistence?](#)

[Describe any corner cases in the UX.](#)

[Describe any libraries you’ll be using and share your reasoning for including them.](#)

[Describe how you will implement Google Play Services.](#)

[Next Steps: Required Tasks](#)

[Task 1: Project Setup](#)

[Task 2: Implement UI for Each Activity and Fragment](#)

[Task 3: Your Next Task](#)

[Task 4: Your Next Task](#)

[Task 5: Your Next Task](#)

GitHub Username: flopshot

App name: Golf Ranger

Description

Problem:

In golf, the variables that go into the perfect swing are endless. There is, however, one variable that is vital to every golfer no matter his or her skill level. The variable in question is distance to target. In the early days of golf, players would judge their distance to target based on sight and instinct, alone. And while many players still use their senses to judge distance, technology has advanced far enough that targeting this distance is cheap, easy to use, and comes readily available in your pocket.

The Golf Ranger application allows any golfer to have the ability to obtain accurate distances to any target out on the golf course. Although there are separate devices that have been available for years, they usually are stand-alone, expensive, and don't sync with any other device. The Golf Ranger application comes built into your android device after install so you can take it anywhere your phone(or tablet) goes.

Intended User

The primary users of the Golf Ranger app will be golf players looking to gain accurate distances to targets.

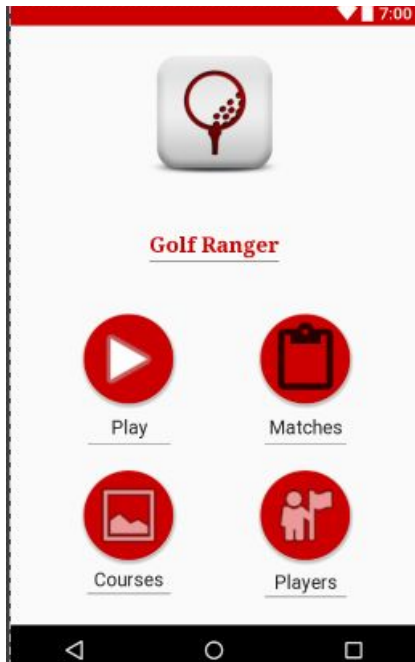
Features

The Golf Ranger application features:

- Overhead aerial map views
- Visual distance marking on the map
- Built in Scorecard
- Live terrestrial condition tracking of elevation and wind speed
- App Widget that provides summary of games played







User Interface Mocks

Home Screen










This is the screen that will be displayed on launch of application. The home screen will have options for starting a new round(play) view an old round(matches) our view players and courses that have been entered.

Matches Screen

	26-Nov-2016 TPC Canyons null
	22-Nov-2016 TPC Las Vegas TPC at The Canyons
	08-Nov-2015 Las Vegas National Golf Course
	31-Oct-2015 Desert Willow Golf Course Desert Willow
	09-Oct-2015 Callaway Golf Center Callaway
	19-Aug-2015

This is the Matches screen that will be shown to the user of previously played rounds by course and showing date of match creation.

Players Screen

refreshed: 24 seconds ago 	
Sean Najera Hcp: 36 8 rounds	You
Adam O'Donnell Hcp: 36 3 rounds	
Chris Carrello Hcp: 36 0 rounds	
Gene ODonnell Hcp: 36 2 rounds	
Jared Cooper Hcp: 36 0 rounds	
Matthew Formica Hcp: 36 4 rounds	
Mike Lively Hcp: 36 1 rounds	


This is the players screen that will be shown to the user of available players to add to a round. The user can also add new players to this screen.

Courses Screen

Las Vegas
Las Vegas C.C.
Las Vegas National G.C.
Desert Pines G.C.
The Wynn G.C.
The Club at Sunrise Sunrise
North Las Vegas
North Las Vegas Par 3
Las Vegas
Royal Links G.C.
Las Vegas G.C.
Stallion Mountain C.C.
Bali Hai G.C.

This is the Courses screen that will be shown to the user of available courses they he or she has manually entered. The user can also add new courses to this screen.


Detail ScoreCard Screen

	Hole :1, Par: 5
	Score: 2 (2)
	Putts: 1
	Penalties: 1
	Green In Regulation ?
	Chip ?
	Sand ?
	Fairway Hit ?

This is the Details Scorecard Screen where the user can see each individual hole's statistics of the current match and update as needed.

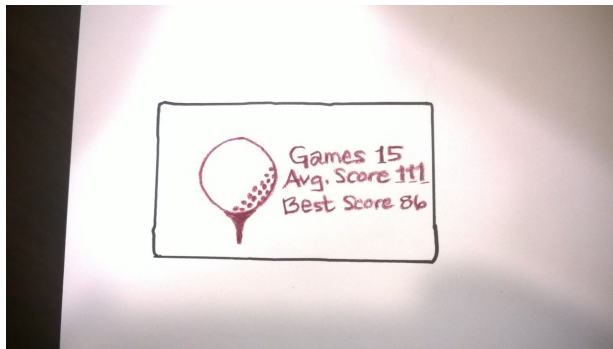
RecyclerView ScoreCard Screen

Hole	Par	Hcp	SN	MF	gO	aO
10	4	14				
11	4	4				
12	3	18				
13	4	2				
14	4	8				
15	5	6				
16	3	10				
17	4	16				
18	4	12				
Back	35		0	0	0	0
Total	71		0	0	0	0
+/- Par			0	0	0	0
Points			0	0	0	0



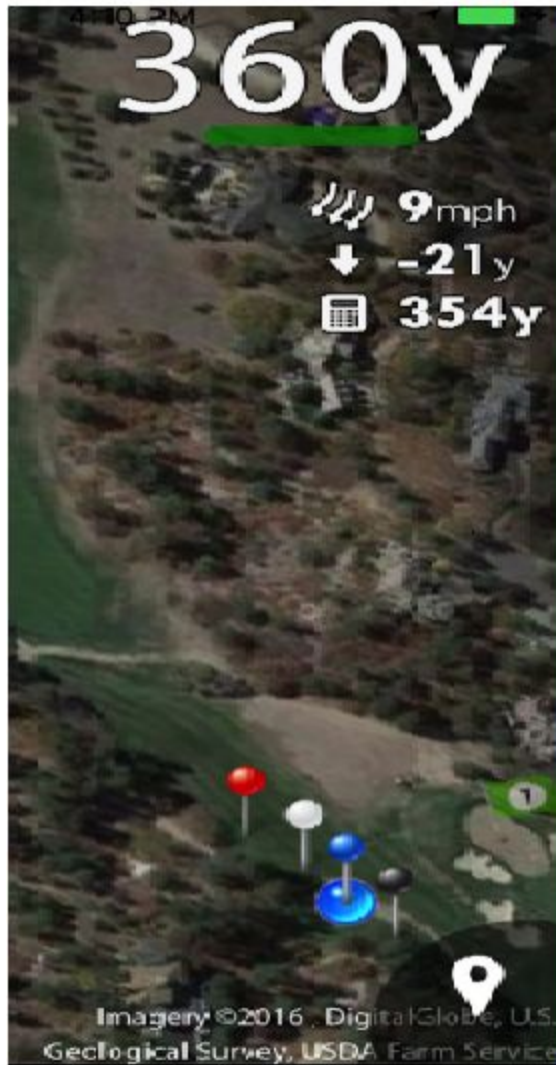
This is the Recyclerview Scorecard Screen where the user can see all holes of the match overall. Each hole, when clicked, takes the user to the details screen.

App Widget



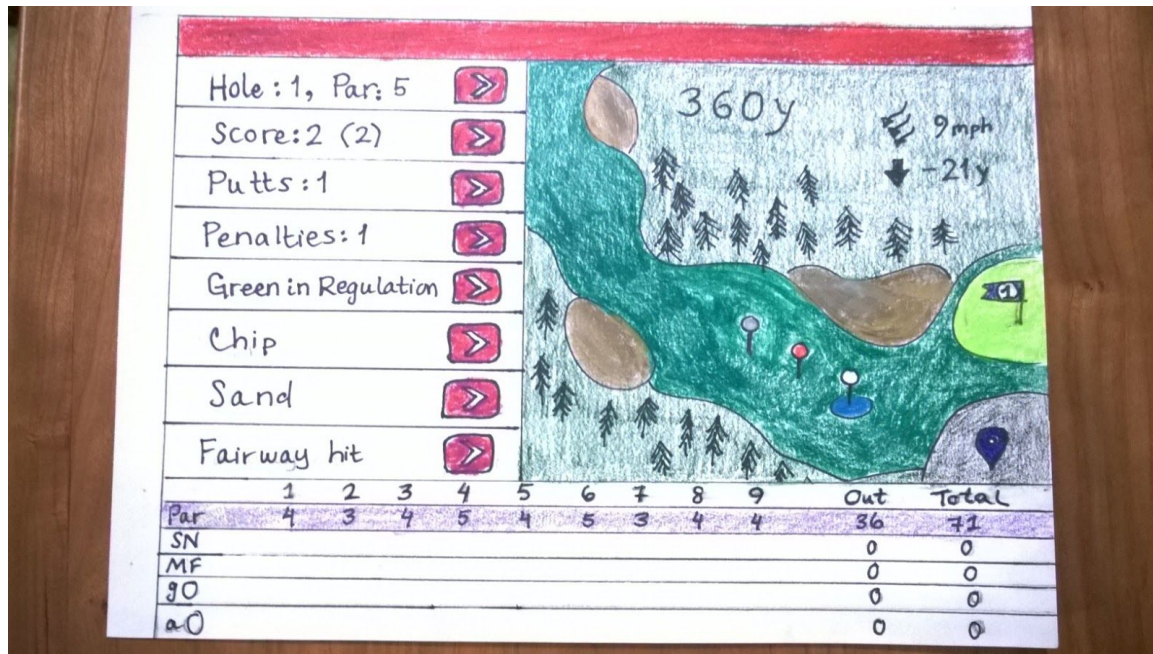
This is the mock app widget that will give the user an overview of the games they have played with simple metrics

Map Screen



And this is the map screen where the user will be shown distances to specified targets along with elevation and wind speed metrics

Tablet View



The tablet view will incorporate the recyclerview, details, and map in one layout.

Key Considerations

How will your app handle data persistence?

The app's built in scorecards will be saved automatically in a content provider. This content provider will hold player names, course names, and scores. Content loaders and content listeners will be used to retrieve the data and content resolvers will be used to update the data.

Describe any corner cases in the UX.

The app will rely heavily on location settings permission being granted. An entire alert protocol will be needed if at anytime the location sensors or networks are down.

The app will also rely heavily on api calls to update terrestrial conditions, so much care will have to be given to error cases in points of entry.

Finally, the app will be fairly battery intensive. Mitigating as much of this battery use will be top priority.

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

The app will be requesting terrain elevation from the Elevation Point Query Service Api from the united states geological survey <http://ned.usgs.gov/epqs>.

Since the wind data will be gathered using the Yahoo Weather Api, the app will be implementing a Yahoo Weather Api Wrapper Library called "yweathergetter".
<https://github.com/zh-wang/YWeatherGetter4a>

Describe how you will implement Google Play Services.

Google maps will be implemented to show the aerial views and displayed distances.
GoogleApi Location Services will be used as the location manager.

Required Tasks

Task 1: Project Setup

The Project will require many outside services and libraries. After initial project repo initialization, the services/libraries setup tasks can be broken down into these steps:

- Request Api key from Yahoo Weather Services
- Request Api key from Unites States Geological Survey
- Generate SHA key and certificate and setup billing for Google Maps services
- Setup library dependencies in gradle for google play services and yweathergetter
- Finally, get support library dependencies for compatibility and design.

Task 2: Implement UI for Each Activity and Fragment

This is a comprehensive list of UI's to implement:

- Build UI for Main/Launch Screen.
- Build UI's for RecyclerView of course and player select screen
- Build UI to add additional courses and players
- Build RecyclerView of summary score card, each hole is an item of the list
- Build hole detail view for more stats on each hole
- Build Map fragment to display distance and terrain data
- Tablet UI will have score summary score card RecyclerView, hole detail view and map fragment combined on screen
- Build App Widget

Task 3: Build Content Provider

The content provider will be built and backed by a sqlite3 database with three tables:

- Player table
- Course table
- Round table (master table)

The usual practice of provider contract, DBHelper, and content provider packaged into a data module will be used. The data will be synced from the scorecard ui to the database any time the activity screen enters the on pause state.

Task 4: Build Api ServiceIntent/Sync infrastructure

Most of the data will be provided by the location services manager and google maps, but the wind data and elevation data will have to be handled carefully. The wind data do not have to be called on demand and can be synced using a JobSync class. The elevation data will have to be called on demand as the user updates markers on the map and calculations need to be made on the fly. Therefore, the IntentService framework will be used to make those api calls

The the Google Api location services will be used in conjunction with runtime permissions framework for devices running Android OS 6.0 and above.

Task 5: Utility Library to calculate distances/elevation change

A library will be made by the developer to calculate this data and summarize to the user in the appropriate display/metrics.

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**"