How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

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: mdhsieh

Place Tracker

Description

Place Tracker allows users to keep track of all the places they've been to. They can use it, for example, to track where they eat or record their visits for doctor's appointments.

Intended User

This app is intended for families, small groups of friends, and individuals who would like to keep track of the different places they go to.

Features

There are three main features:

- Adds a place by current location
- Adds a place manually if it can't be located
- Saves place information such as name, address, and number of visits

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 Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

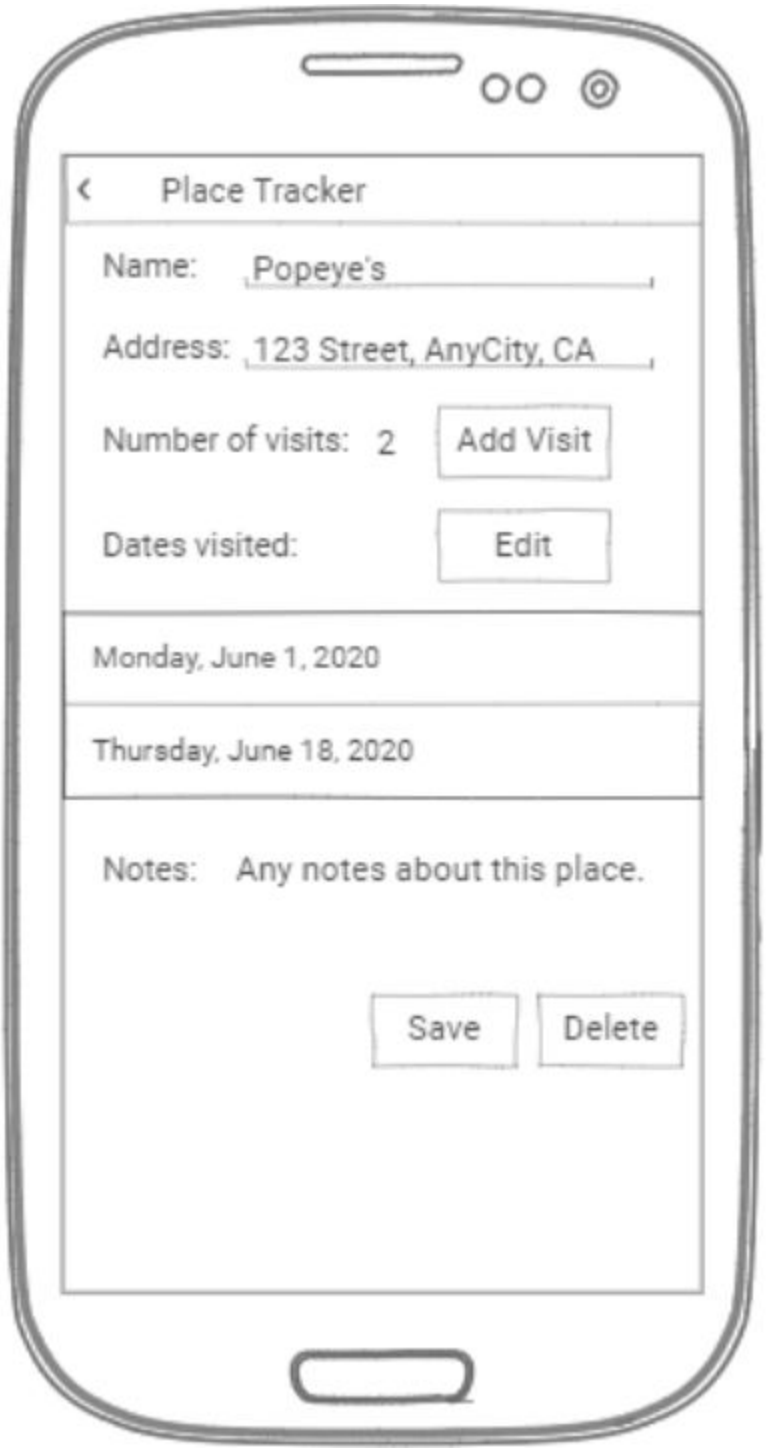
I used Ninjamock to make the following screens. The link to the mockup is: https://ninjamock.com/s/K111CGx

Place List Screen

This is the main screen where the user can add places and view their list of saved places. The user can search for a nearby place using the search bar or add a place manually using the

manual button. Clicking a place in the list will show its details screen which has info like name, address, and number of visits.

Place Details Screen



This is displayed when the user clicks a particular place from their list of saved places. The user can edit the info here. They can click the “add visit” button to record their latest visit to this place, which defaults to the current date. The user can also delete this place if they want.

Manual Place Details Screen

This is shown when the user clicks the button to add a place manually in the main screen. The user fills out the info and can then add this place to their list. The screen looks very similar to the place details screen.

Widget Screen

This is the app's widget. The user can see a saved place's name and address here.

Key Considerations

How will your app handle data persistence?

The app will store information locally on the user's phone. It will use the Room Database and ViewModel with LiveData. The class which defines the Room entity will need to hold a place's Place ID, name, and address. It will also need to hold the number of times the user visited a place and all the dates visited.

Describe any edge or corner cases in the UX.

If there are no places that match the user's search query, the app should show a text telling the user this.

If the user tries to add a place that already exists in their list, the list should stay the same and a text should inform the user about this.

If Internet connection is not available, there should be a message indicating this.

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

Picasso will be used to handle the loading and caching of images. For example, there can be images in the place details screen.

Describe how you will implement Google Play Services or other external services.

The Google Places SDK will be used to enable autocomplete in the search bar and to search nearby places. Place information will be taken from the Google Places SDK as well.

Google AdMob will be used to display a banner ad at the bottom of the screen.

Next Steps: Required Tasks

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

Task 1: Project Setup

The app will be written in Java.

To set up, each of the required libraries should be added as needed. The search bar will require the Google Places SDK. The place list requires Room's component libraries. Loading and caching images requires the Picasso library.

In the app build.gradle file:

- Add Google Places SDK dependency
- Add all the Room dependencies
- Add Picasso dependency

These dependencies should be the latest stable versions. The app should be using AndroidX because Version 2.0.0 and later of the Places SDK is dependent on AndroidX.

The app will also need an API key to use the Google Places SDK.This API key needs to be restricted to this app only. In the Google Cloud console:

- Make a new project for this app and create a new API key
- Add restriction to this app only

Task 2: Implement UI for Each Activity and Fragment

The app should keep all Strings in a strings.xml file and enable Right To Left layout switching on layouts. To support accessibility, UI components should have content descriptions and the app should allow navigation using a D-pad.

There is one MainActivity which contains the search bar and list of places.

- Build UI for MainActivity
    - Text, search bar, manual button, picker button
    - Menu to hold settings icon
- Build UI for the place list
    - RecyclerView to display all saved places, each with a name and address

- Build UI for place list DetailActivity
    - Create layout
    - EditText and buttons, RecyclerView or collapsing text to hold the dates
- Build UI for ManualPlaceDetailActivity
    - Create layout, add EditText to allow user input

Task 3: Implement Google Places SDK

First, the Google Places SDK needs to be initialized and the API key used to get the user's current location.

The next task is to create the search bar where the user can search for a nearby place and select it.

- Create autocomplete search bar in UI
- Get search results based on user's current location and query
- Get Place ID, name, and address of selected search result

Task 4: Implement Place List

The app will use an IntentService to update the Room Database with place info from the Places SDK using the Place ID.

Now that the app has place info, a Room Database needs to be made to save that info:

- Create Place model class
- Create Room Entity using that class
- Create Data Access Object
- Wrap class with LiveData
- Create Room Database
- Create ViewModel

When the user selects a place from the search bar, the place should be inserted to the database. A place list needs to be made to show the places stored in the database.

- Populate Room Database with newly selected places
- Connect place list RecyclerView with Room Database, and display the name and address of each place

Task 5: Implement Place Details Screen

When the user clicks a place in his/her list, the details screen should display that place's info.

- Get address and name of that place from database

- Get dates and number of previous visits from database, if any
- Add “add visit” button functionality, which should add today’s date and increase number of visits by one
- Allow user to edit individual dates
- Allow the user to delete that place from the database. If the user deletes a place he/she should be brought back to the updated place list screen

Task 6: Handle Error Cases

Display a message if a UX corner case occurs.

Display message if:

- No Internet connection
- No place found
- Trying to add existing place

Check that all basic functionality is working correctly, with no crashes.

- Database correctly inserts, deletes, and updates places

Task 8: Implement Widget

Create the widget:

- Create widget
- Get place’s name and address from the list
- Open the app when the widget is clicked

Task 9: Create Build Variant

Create a release version of the app

- Modify app build.gradle to include release build type
- Add signing configuration
- Check that installRelease in Gradle pane installs the release version of the app

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

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"