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

GitHub Username: Giusan82

EasyTrip

Description

EasyTrip is your pocket travel guide where you can discover highlights and hidden gems in over 50,000 destinations worldwide, find point of interest nearby you, such as hotels, sights, activities, restaurants and so on. They can be found simply searching them by proximity or by keyword and mark them as your favorite.

Intended User

EasyTrip is intended for Travelers who are looking for new interesting places to visit or find service nearby their current location.

Features

- Find point of interest nearby you searching by proximity or by keyword.
- Extensive information about interested places.
- Mark interested place as favorite.
- Widget with weather forecast of your current place and/or your chosen destination place.

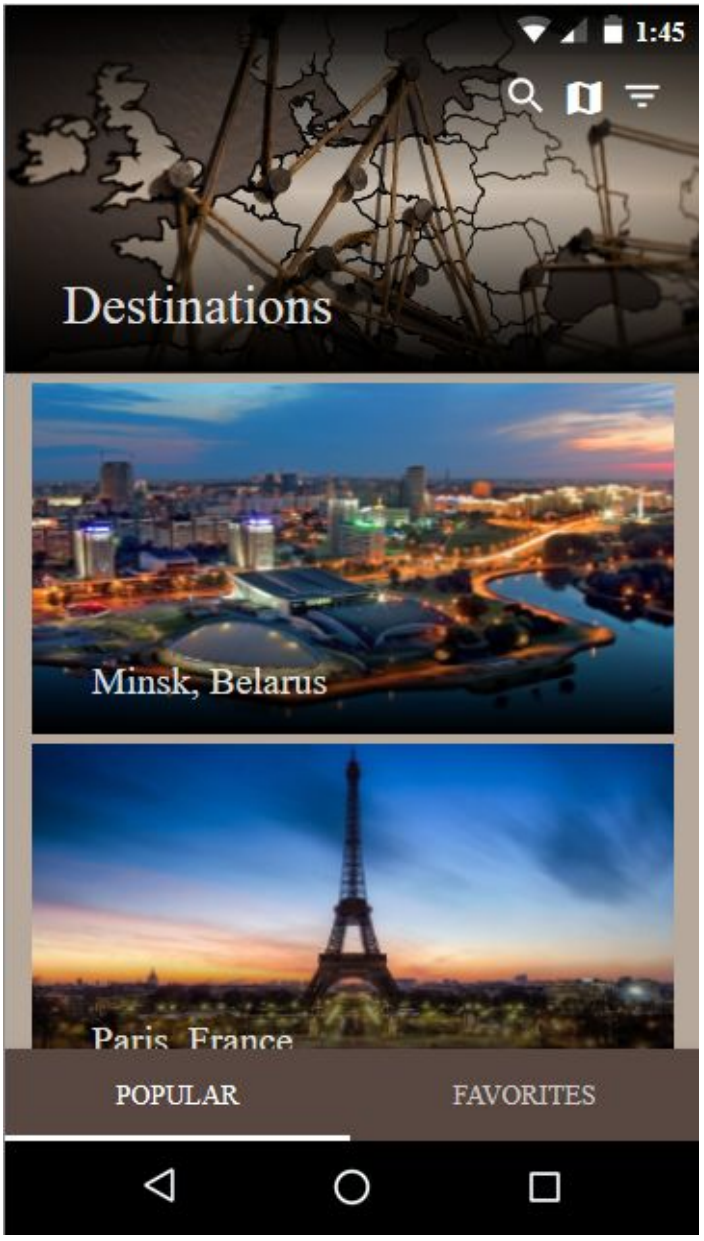
User Interface Mocks

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.

Screen 1

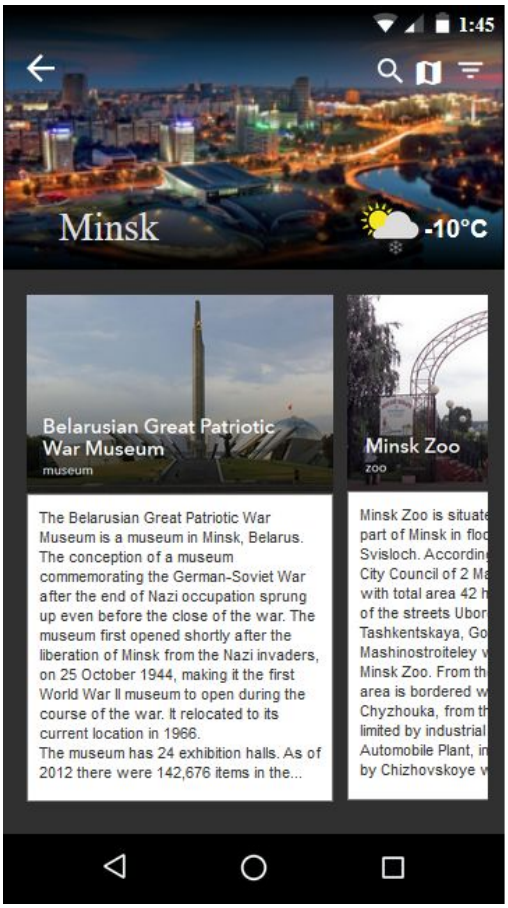
Splashscreen with some citation.

Screen 2



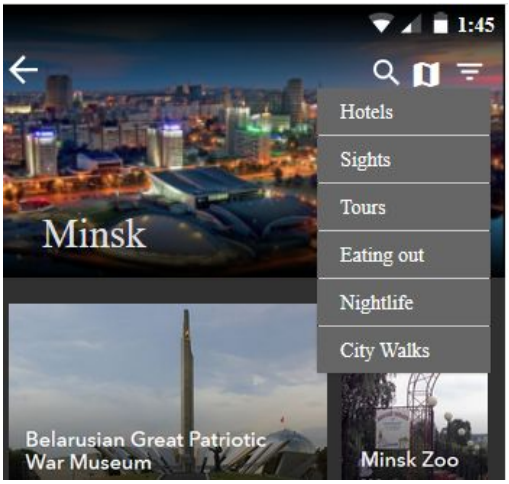
This is the Main screen where the user can find recommended destination, search for an interesting destination using the search icon on top or find nearby place to the user using the map icon on the top. Here the user can also find saved places that intend to visit.

Screen 3



After clicking on a destination place of the previous screen, the app open a screen with recommended places of that destination with its current weather.

Screen 4

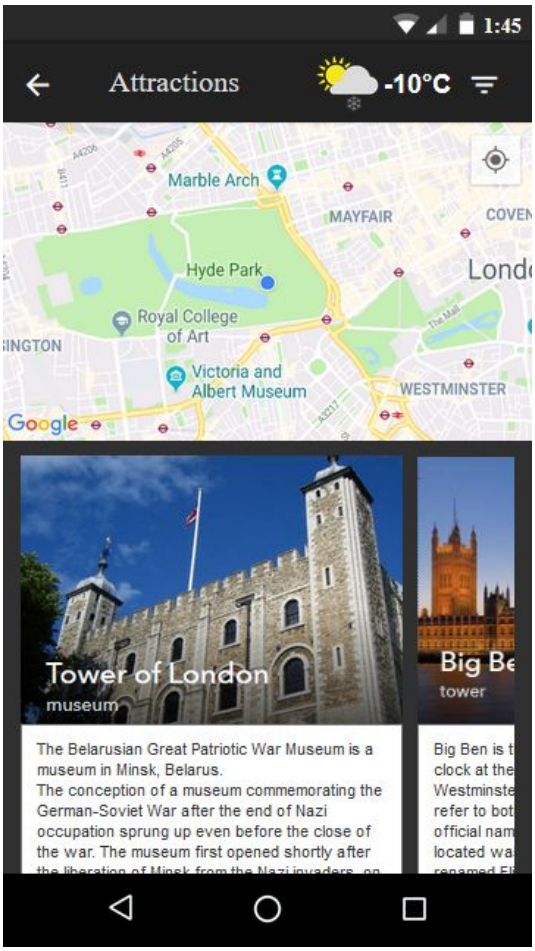


Clicking on the filter icon on top it is possible filter the place by category.

Screen 5

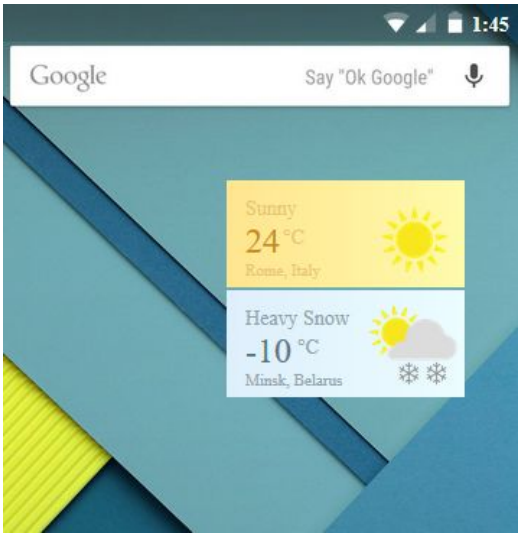
After clicking on a place, the app open a detailed screen with information of that place.

Screen 6



if the map icon is clicked, this screen is opened where it is shown to the user the recommended places nearby to him that can be filtered for category and the local weather.

Screen 7



Here it is shown the app widget that display the local weather and the weather of some destination chosen by the user.

Key Considerations

How will your app handle data persistence?

The data of Top Destination on the home screen and the weather data are cached locally into SQLite database and made accessible using a Content provider for minimize the server calls. While the data of nearby places are fetched directly from Google Server using an AsyncTask.

Describe any edge or corner cases in the UX.

- If the destination/places searched by the user return zero items, the app show an empty view
- if the app cannot fetch data from server, the app show a message that inform the user of the problem without crash.
- if the location permission is not granted the app show a message without crash.
- if the app cannot determine the current position, because the gps is turned off, the app show a message without crash.
- the image are resized to avoid crash due to out of memory issues.
- the app handle unexpected input data from the api request without crash.

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

Glide will be added to handle the loading and caching of images.
Butterknife for binding Android view.
Volley for handle the http requests from weatherbit.io, triposo.com and google places api.
Google Play Services, as places and location for determine the current location and find the point of interest nearby of it.
Recyclerview, cardview and design android support library for displaying the data. Android support palette, to extract prominent colors from an image and use it to set color of the actionbar.

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

1. Google Location service for determining the current location, display it on map.
2. Google Places Service to provide access to Google's database of local place and business information nearby the user.
3. Google AdMob for displaying banner ads and Interstitial ads. The banner will be shown at the bottom of the screen, while the interstitial will be shown when the user click on one place to see it details.

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

- Find Api service that provides data related to travel tour, nearby places and weather, as triposo, google places and weatherbit respectively.
- Read the documentation and terms of use of relative api.
- Add and configure libraries with the respective dependencies.
- Setup the api request to fetch all data you need from the api server with volley library.
- Setup the current location of that device using the google places and location library and display it on map.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for a SplashActivity.
- Build UI for MainActivity where a list of destination is displayed based on preferences, such as top destination, with a tab layout where the user can swipe from suggested to favorite destinations and vice versa.
- Build UI for point of interest such as hotels, sights and other suggestions relative to that selected city.
- Build UI that shows detail of a selected place.
- Build UI for show the current place on map and the nearby point of interest.

Task 3: Implement Data Models and Json parse

- Create a class for data models used also for Json parse with Gson library.

Task 4: Data Persistence and Services

- Setup SQLite database and Content Provider
- Setup a favorite destination/place tab.
- Setup a JobDispacter service for caching data at regular intervals from weather api.

Task 5: Implement Widget

- Setup a list Widget that shows weather of current and/or prefered place.

Task 6: Preferences

- Implementing the search filters and settings preferences

Task 7: Implement Paid and Free Flavors

- Break up the project into free and paid module and add the ads into the free flavor.

Task 8: Testing

- Implementing connected test to verify that all view respond as expected

Task 9: Polishing the UI

- Completing the UI to meet Material Design specifications.
- Add standard and simple transitions between activities.

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**"

Rubric

App validates all input from servers and users. If data does not exist or is in the wrong format, the app logs this fact and does not crash.
App includes support for accessibility. That includes content descriptions, navigation using a D-pad, and, if applicable, non-audio versions of audio cues.
App keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.
App provides a widget to provide relevant information to the user on the home screen.

App integrates two or more Google services. Google service integrations can be a part of Google Play Services or Firebase.
App theme extends AppCompat.
App uses an app bar and associated toolbars.
App uses standard and simple transitions between activities.

App stores data locally either by implementing a ContentProvider OR using Firebase Realtime Database OR using Room. No third party frameworks nor Persistence Libraries may be used.

Must implement at least **one** of the three
If it regularly pulls or sends data to/from a web service or API, app updates data in its cache at regular intervals using a SyncAdapter or JobDispacter.
OR
If it needs to pull or send data to/from a web service or API only once, or on a per request basis (such as a search application), app uses an IntentService to do so.
OR
It it performs short duration, on-demand requests(such as search), app uses an AsyncTask.

If Content provider is used, the app uses a Loader to move its data to its views.
If Room is used then LiveData and ViewModel are used when required and no unnecessary calls to the database are made.