GitHub Username: melomg

WonderAndWander

Description

WonderAndWander app is a simple app that helps you to decide whether a city is the one you are looking for, in every aspect. You can search cities, see most important things from safety to cost of living about cities. WonderAndWander app will store the cities you favorited if you logged in. This way you never need to remember which city you liked once upon a time. It also helps you to compare the cities so you can decide better. Don't wait, wonder and wander now.

Intended User

For everyone who is seeking a nice travel or wants to change job and move from his/her city or for students who will start journey of their life and wonders which city can make best university memories, or event just sits and wonders which city is the best.

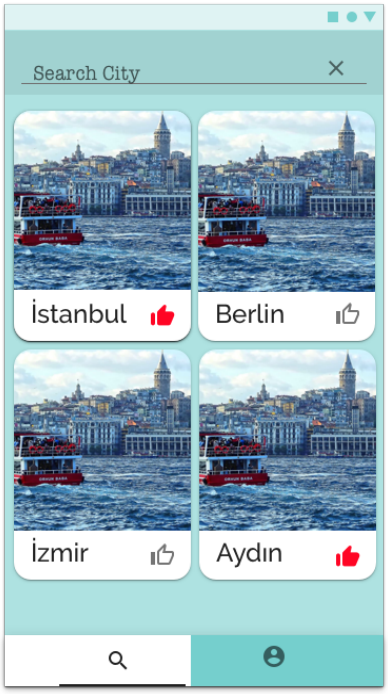
Features

Main features of the WonderAndWander app:

- Logs a user in
- Makes a city search and shows basic information about selected city. (Housing, Cost of Living, Travel Connectivity, Safety, etc.)
- Favorites a city and stores them in Firebase Realtime Database if the user logged in.
- Compares two cities's basic information.

User Interface Mocks

Screen 1

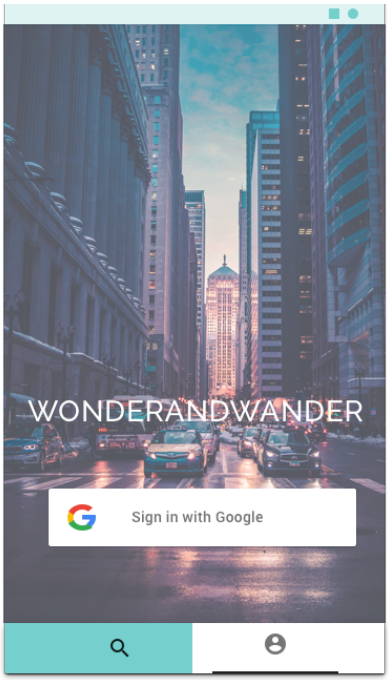


First screen users will see when they open the app. Users can search city they are looking for by typing the city name and it will appear at the same page as card item.

Screen 2

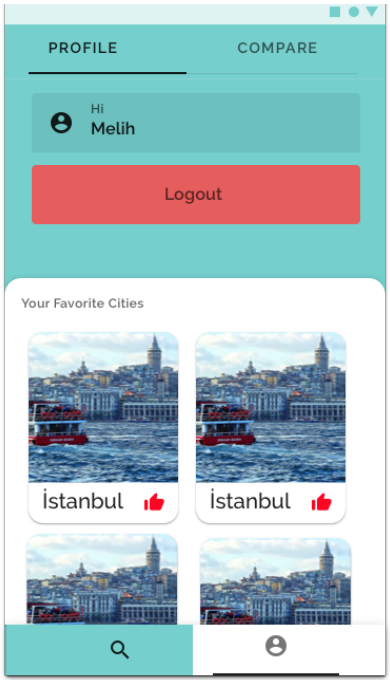
When users click the searched city, they will see basic information about the city. If the user logged in, they will be able to add the city to compare list.

Screen 3



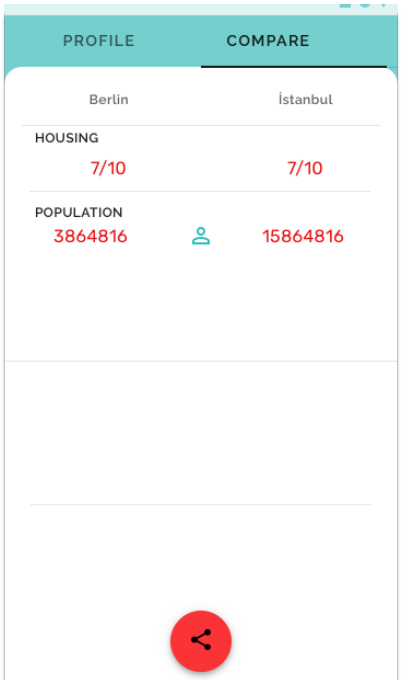
Users will be able to login to the app by Firebase UI Auth. This will give ability to store their favorite cities at Firebase Realtime Database.

Screen 4



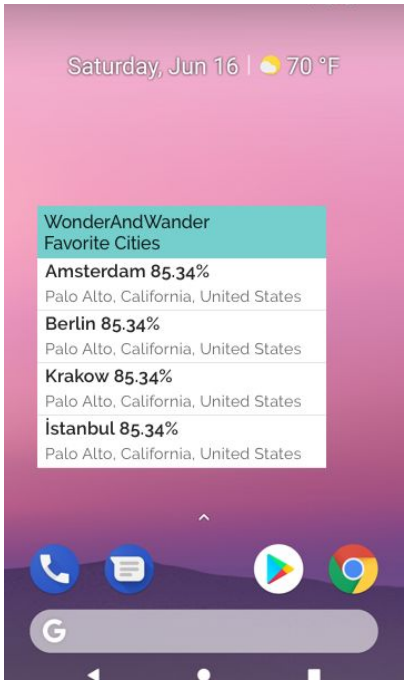
When users logged in to the app, they will be able to add a favorite city from search result and see their favorite cities in profile tab.

Screen 5



When users logged in to the app, they will be able compare two added cities from city detail page.

Screen 6



Favorite cities will be fetched from local database and will be updated when local database changed. Widget list item will show the name of the city, the teleport rating of the city and area and country name that city belong to.

Key Considerations

How will your app handle data persistence?

- App will use AsyncTask for searching cities.
- App will store favorited cities in both as local by using Room to store data at SQLite and remote by using Firebase Realtime Database. If there is no internet app will show local data otherwise app will show the fetched data from Firebase Realtime Database and update local data.

Describe any edge or corner cases in the UX.

- App includes support for accessibility. That includes content descriptions, navigation using a D-pad.
- 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 keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.
- App theme extends AppCompat.
- App uses an app bar and associated toolbars.
- App uses standard and simple transitions between activities.
- UI mocks depict interaction stories that adhere to Core App quality guidelines.
- App first launches main activity screen which contains two screen: Search and User. Navigation between these screens will be handled via bottom navigation bar. When users see cities of search result and then they click one of the cities, City Detail screen will show up as new screen (as activity). From there, users can add the city to the compare list. When they navigate back, they will go back to the tab page where they come from. (Users can also go to the City Detail screen from Profile tab in User Navigation Tab by clicking one of favorited cities). If users are not logged in they will see the Login screen when they click the User tab but if they are, they will see User screen which contains two tabs: Profile and Compare. When users logout from the app, Profile and Compare will be gone and they will see the Login screen at User tab again.

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

- Support Libraries - version 27.1.1: for better design and compatibility.
- Android Architecture Components - version 1.1.1: for MVVM design pattern and clean code.
- Room - version 1.1.0: for easy data storage by providing an abstraction layer over SQLite.
- Dagger - version 2.15: for dependency injection to write more testable and maintainable code.
- Retrofit - version 2.3.0: a rest client which makes easy to retrieve and upload JSON via a rest based web service.
- Okhttp3 - version 3.10.0: an HTTP & HTTP/2 client that covers many things.
- Glide - version 4.6.1: provides a framework for image loading.
- Timber - version 4.6.0: a logger which provides utility on top of Android's normal Log class.
- Leak Canary - version 1.5.4: A memory leak detection library.
- Map Sdk - version 12.0.0: to show location of cities.
- Firebase UI Database - version 4.0.1: to store users' favorite cities at Firebase Realtime Database
- Firebase UI Auth - version 4.0.1: to authenticate the user
- Ads Play Services - version 15.0.1: Ads will be shown to free app's users. The app displays test ads in debug variant.

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

- Teleport API (https://developers.teleport.org/api/): to fetch information about cities. It gives a nice documentation for RESTful API and free to use.
- Map Play Services: will be used to show location in a detail of city screen with lite mode.
- Firebase UI Auth Service: This service will be used to log the user quickly. I need the login mechanism because I will store users' favorite cities at remote. So when users logged in and favorite some cities, they will be able to access these cities from everywhere including iOS and web for future considerations since the service has same authentication steps for all clients.
- Firebase Database Service: This service will be used to store registered users' favorite cities. The reason I'd like to use this service is that data is synced across all clients in real time, and it still remains available even when an app goes offline.

Next Steps: Required Tasks

Task 1: Project Setup

Main aim of the app is to create a teleport app that will have search, display cities and save cities to users' favorite functionalities. App will be written solely in the Java Programming Language and will utilize stable release versions of all libraries, Gradle, and Android Studio.

As a pre-requisite, the app will be configured for four services. Firebase UI Auth Service, Firebase Database Service, Map Play Services and Ads Play Services integrations should be completed.

- Firebase UI Auth Service: See the link for configuration steps.
- Firebase Database Service: Setup Firebase Realtime Database and configure ProGuard. For more information about configuration and setup see the link.
- Map Play Services: Lite Mode of Map services will be used in the app to show the location of selected city. Get an API key from the Google Cloud Platform Console for Map SDK and follow the configuration steps.
- Ads Play Services: There will be two different build configurations as free and paid app. This service will be used to show ads in free app.

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for SearchFragment
- Build UI for UserFragment
- Build UI for LoginFragment
- Build UI for ProfileFragment
- Build UI for CompareFragment

- Build UI for FavoriteBottomSheetDialogFragment

Task 3: Create Build Variant

- Create a build variant as free and paid.
- Configure the dependency of ads to show it in only free build variant.

Task 4: Implement Google Play Services

- Implement Firebase UI Auth Service
- Implement Firebase Database Service
- Implement Map Play Services
- Implement Ads Play Services

Task 5: Handle Error Cases

- Show an error if the app is offline while searching.
- Show an error view if something went wrong while searching or favouriting a city or logging to the app.

Task 6: Show Search Result

- Get search result from entered city name and show them.
- Show a message when users pressed a favorite if they are logged in.
- Add city to favorite cities of users if user logged in and they clicked to favorite icon.
- Remove city from favorite cities of users if they clicked to unfavorite icon.

Task 7: Show City Detail

- Show city detail of pressed item of search list.
- Show a message when users pressed a favorite if they are logged in.

Task 8: Add City to Compare

- Add city to compare list in city detail if user logged in.
- Forward user to LoginFragment if user not logged in.

Task 9: Authenticate The User

- Authenticate the user with Firebase UI Auth.
- Save the user at both local by using Room and at remote with Firebase Realtime Database.

Task 10: Logout the user

- Remove the user related information from the app.
- Show the LoginFragment when user logout.

Task 11: Show User Information

- If the user logged in show user information on Profile.
- Show user's favorite cities.
- Remove a favorite city if user pressed a unfavorite icon on it.

Task 12: Compare Two Cities

- Show a message if there aren't two cities to compare.
- Show comparement of two cities.

Task 13: Add A Widget For Favorite Cities

- Create a widget that will show users' favorite cities and update the list when local data has changed.
- Show an empty view incase users are not logged in or there is no favorited city.