GitHub Username: Ezike

MyWeather

Description

Get real-time weather updates and daily weather summary for your current location with MyWeather app. Weather updates for other locations and current time in those zones are also delivered in an intuitive interface.

Intended User

The app will be useful to anyone. But travellers who need quick weather updates for their destinations will find it very handy.

Project Requirements

- App will be written solely in Java
- App will use stable versions of all libraries and dependencies including Gradle version 4.10.1 and Android studio 3.3.2
- Accessibility features like content description, RTL layout support are included.
- App theme extends from AppCompat and uses supported material design components
- Resources (Strings, colors, dimens) are stored in their respective resource folders
- App includes a homescreen widget that displays relevant information

Table below shows the summary of libraries used in the project and their stable releases

Library	**Version**

Android support library	`Implementation ‘com.android.support:appcompat-v7:28.0.0’`
	`Implementation ‘com.android.support:cardview-v7:28.0.0`
	`Implementation ‘com.android.support:palette-v7:28.0.0’`
	`Implementation ‘com.android.support:recyclerview-v7:28.0.0’`
	`Implementation ‘com.android.support:design:28.0.0’`
	
	`Implementation ‘com.android.support.constraint:constraint-layout:1.1.2’`

Room, Viewmodel, Livedata & Lifecycle	`Implementation “android.arch.lifecycle:extensions:1.1.1”`
	
	`annotationProcessor "android.arch.persistence.room:compiler:1.1.1"`
	
	`annotationProcessor "android.arch.lifecycle:compiler:1.1.1"`
	
	`Implementation "android.arch.persistence.room:runtime:1.1.1"`

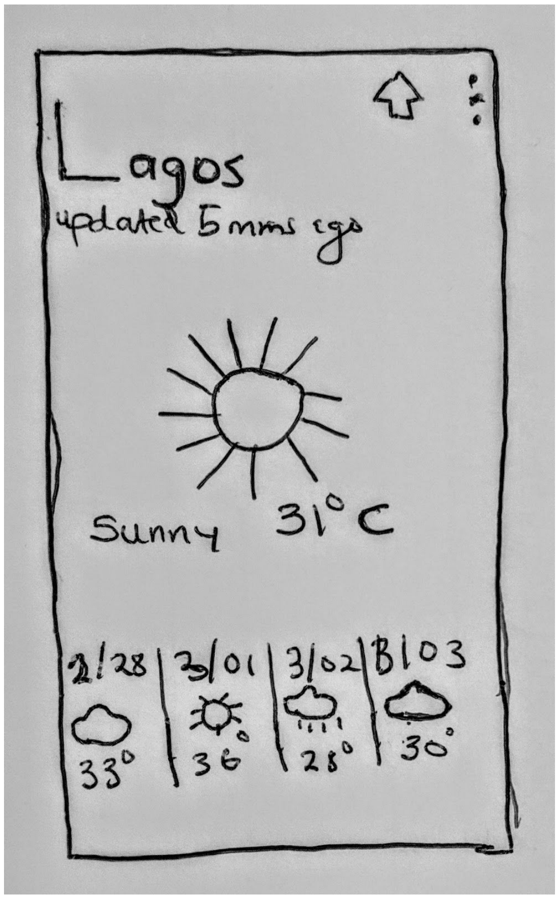
Firebase	`Implementation`

	`‘com.firebase:firebase-core:16.0.8’`
Timber	`Implementation ‘com.jakewharton.timber:timber:4.7.1’`
Dagger 2	`Implementation ‘com.google.dagger:dagger-android:2.16’ Implementation ‘com.google.dagger:dagger-android-support:2.16’ annotationProcessor com.google.dagger:dagger-android-processor:2.16`
Glide	`implementation 'com.github.bumptech.glide:glide:4.9.0' annotationProcessor 'com.github.bumptech.glide:compiler:4.9.0'`
Retrofit	`implementation 'com.squareup.retrofit2:retrofit:2.5.0'`
OkHttp	`implementation 'com.squareup.okhttp3:okhttp:3.13.1'`
Moshi	`implementation("com.squareup.moshi:moshi:1.8.0")`

Features

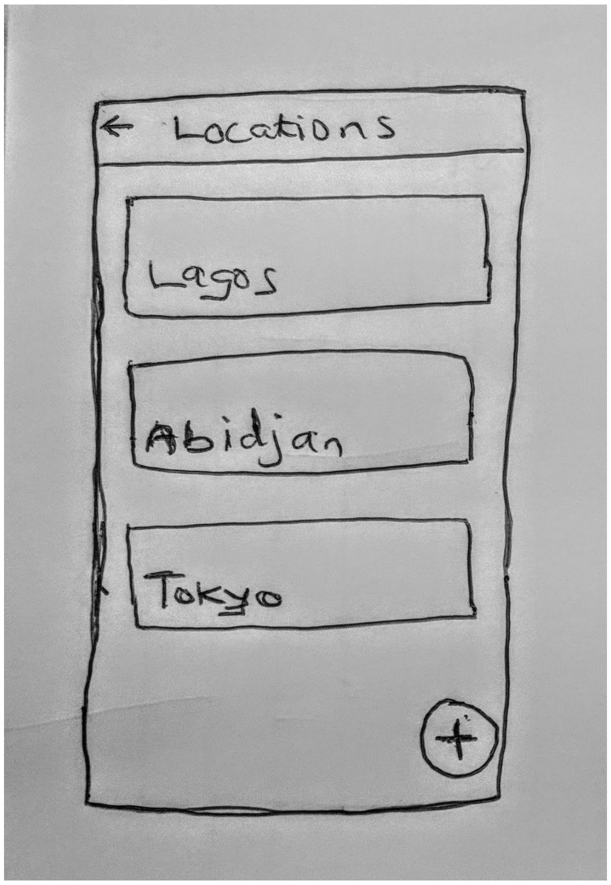
- Displays current weather info for a location
- Sends daily notifications with weather summary
- View weather conditions in other locations
- View weather summary in homescreen widget
- Share weather info

User Interface Mocks



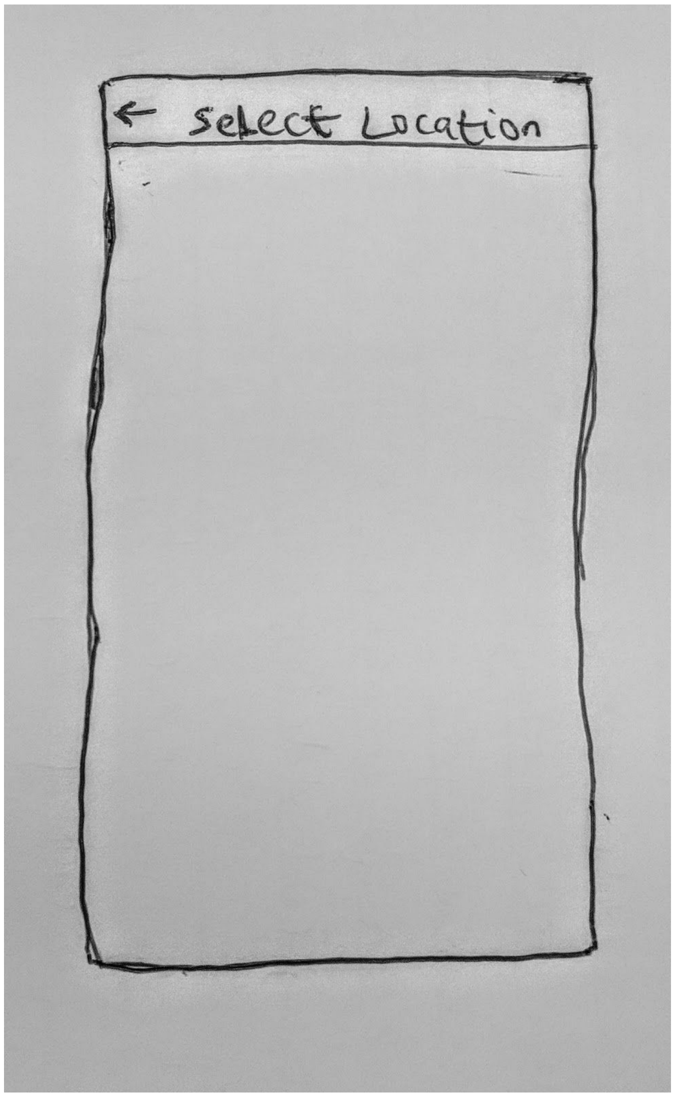
The screen displays weather information for a selected location

Screen 2



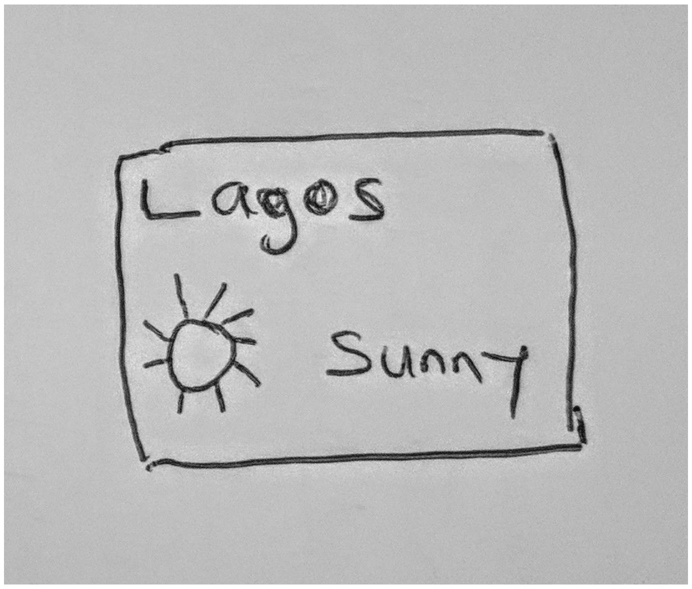
This screen displays all added locations and allows user add new locations.

Screen 3



The screen shows the search view for choosing new location

Screen 4



This is the app widget interface on the home screen

Key Considerations

How will your app handle data persistence?

Weather data will be persisted using Room Database. Sharedpreferences will also be used to store other necessary information.

Describe any edge or corner cases in the UX.

If mobile data is off, use snackbar to inform user and use snackbar action to allow user retry data loading
If location permissions are denied, exit the app and request permissions on app relaunch
If GPS is turned off, display a dialog telling user to turn on GPS
Use empty views incase there's no data available for display

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

Retrofit will be used for making network calls
Dagger2 will be used for dependency injection
Timber will be used for adding log messages

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

The app will display test ads using Google adMobs service. The ads will be unobtrusive in order not to ruin the user experience.
Google analytics or Firebase crashlytics will be used for error logging and reporting.

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

- Add gradle dependencies for third party libraries
- Obtain api key for fetching weather from remote source
- Add packages for different components of project (data, ui, utils, etc)
- Add permission for internet in the manifest
- Add permission for location in manifest

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity that displays weather for current location
- Build UI for locations List activity
- Build UI for widget

Task 3: Setup Data layer

- Add Entities, Dao and Room Database
- Retrieve weather data from api with retrofit
- Create Network data source and Repository classes for exposing data to other components like viewModel
- Add intent service that ensures local data is updated at regular intervals
- Move all object creation and dependencies to dagger modules

Task 4: Display Data on main screen

- Create layout for main weather activity
- Add viewmodel class to deliver data to the UI
- Retrieve locally stored weather data from the Repository and display on the main screen

Task 5: Setup add Location screen

- Create layout for locations list with RecyclerView
- Add a floating action button that allows user add new location
- Display the chosen location on the list and add it to the main screen
- Implement swipe to delete for removing locations

Task 6: Setup App widget

- Create layout for App widget
- Add Widget provider
- Display today's weather for selected location in the widget

Task 7: Polish UI

- Use material components to ensure a unified design system across the app
- Move resources (strings, dimens) to their respective resource files.