GitHub Username: Monotoba

Aviation Weather App

Description

Pilots are required by law to be informed of the weather in their flight path. While this app is no replacement for getting the weather from the local flight service station, it provides a convenient application for viewing the weather at stations of interest.

With that said, here are the issues that this application aims to solve:

- Pilots want a quick and easy way to obtain aviation weather before they arrive at the airport.
- Weather is a safety issue for pilots and they often need regular updates. This app provides regular weather updates.
- METAR data is hard to read at a glance. This app provides that data in a human readable format.

Intended User

This application is intended for private and commercial pilots who need a quick and simple way to monitor the weather for a set of chosen stations.

Features

List the main features of your app:

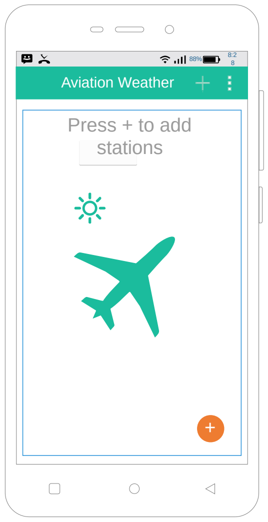
- Collects information and Persists that information locally via SQLite database using ROOM DAO
- Updates weather data for selected stations
- Displays important weather data for selected stations world wide

Requirements to be Met

- Java, SQL, and XML languages will be used for the development of this app
- App will keep all user facing strings in a strings.xml file and will enable RTL layout switching on all layouts
- This app will include support for accessibility including content descriptions, navigation and using a D-pad

User Interface Mocks

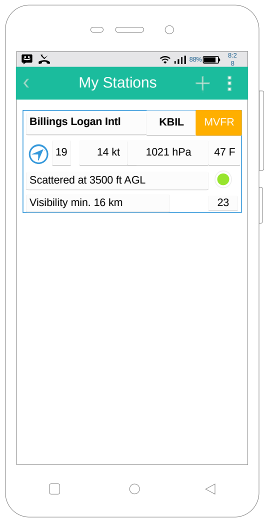
Screen 1: Home Screen



1. Home Screen -
No Stations Added

Screen 1. Home screen mock-up: This screen greats the user the first time the app is opened or when there are no stations selected. A message is displayed to notify the user to add stations by pressing the “+” button in the toolbar or on the floating action button. A drop-down menu is provided to allow the user to configure unit settings.

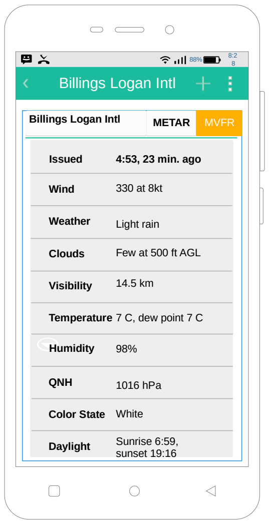
Screen 2: Home Screen with Station Added



2. Home Screen - 1
Station Added

Screen 2 shows the home screen with a single station added. Some of the details of the station listing can be seen here. They include: Airport name, IATA or ICAO code, Visual indicator of current flight rules, Wind direction and speed, Pressure, Temperature (F or C as selected in the settings), Human readable cloud cover statement, human readable visibility statement, and report age. The button show is used to select a station to display in the device's home screen widget. Only a single station may have this button enabled. If enabled, a subset of station data will be displayed in the devices home screen widget.

Screen 3: Weather Detail Screen

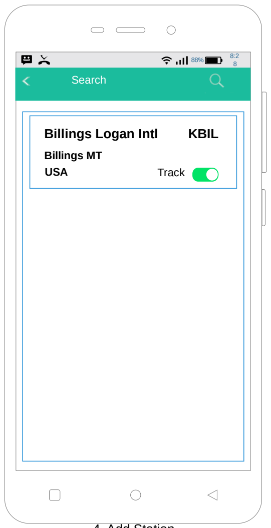


Issued	4:53, 23 min. ago
Wind	330 at 8kt
Weather	Light rain
Clouds	Few at 500 ft AGL
Visibility	14.5 km
Temperature	7 C, dew point 7 C
Humidity	98%
QNH	1016 hPa
Color State	White
Daylight	Sunrise 6:59, sunset 19:16

3. Weather Detail Screen

Screen 3 shows the a weather detail. Here additional information regarding the station's weather conditions can be found.

Screen 4: Add Station Screen

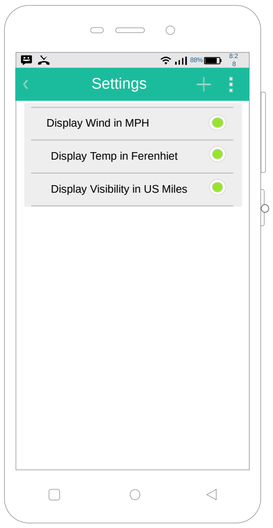


Billings Logan Intl KBIL
Billings MT
USA
Track

4. Add Station Screen

Screen 4 show the Add Station screen. Here the user can search for stations by ICAO codes or city name. The results are displayed in the list below. The green slide button on the result item card is used to turn station tracking on/off for the selected station.

Screen 5: Settings Screen


Settings

Display Wind in MPH ●

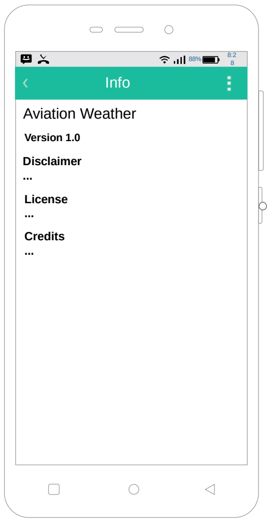
Display Temp in Ferenhiet ●

Display Visibility in US Miles ●

5. Settings Screen

Screen 5 Shows the settings screens where the user may set unit display preferences.

Screen 6: Information Screen


Info

Aviation Weather

Version 1.0

Disclaimer
...

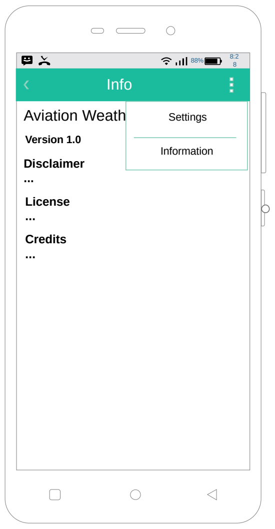
License
...

Credits
...

6. Information Screen

Screen 6 shows the various types of information that shall be included on the information page. A disclaimer stating that the app is not a replacement for weather data from formal flight service stations and that it should not be used for navigation purposes shall be included. Along with a statement of release of liability. In addition, the release version software license and credits for pen source software shall be included in this page.

Screen 7: Widget



Aviation Weath

Version 1.0

Disclaimer
...

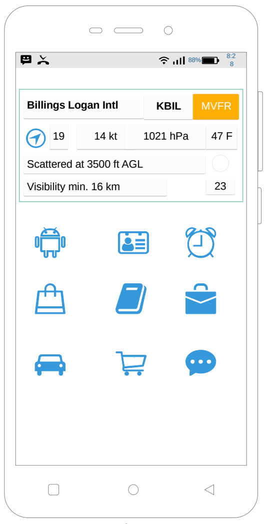
License
...

Credits
...

7. Drop Down Menu

Screen 7 shows the drop down menu from which the user may navigate to the settings or information screens.

Screen 8: Widget



Billings Logan Intl		KBIL	MVFR
19	14 kt	1021 hPa	47 F
Scattered at 3500 ft AGL			
Visibility min. 16 km			23

8. Device Home Screen Widget

Screen 8 shows the applications widget in a representative setting on the device's home screen. Clicking on the widget will open the application to the station list (the apps home screen).

Key Considerations

How will your app handle data persistence?

The application will handle data persistence by using ViewModel, SharedPreferences, and Room. Room shall be used to store data between network calls, and maintain a short history of station data. I plan on adding a feature in the near future that will use this weather history. SharedPreferences shall be employed for storage of user settings such as those for selected display units.

Describe any edge or corner cases in the UX.

The application will always open on the main screen. This screen shows the list of selected stations. Users can navigate to any listed station's detail view by tapping that station's view in the watch list. Pressing back on all other screens will take the user back to the main screen. If they click back on the main screen, it will take them out of the app. Up button navigation shall also be employed and shall navigate the user back to the previous screen. The up button shall be removed from any screens where it's use seems inappropriate, such as the home screen.

Describe the version of Android Studio and Gradle you will be using.

- I will be using Android Studio >= 3.1.4, which is a stable version
- However, as I try to keep Android Studio and Gradle up to date, it is possible these versions shall change during development. I try to always use the latest stable versions.
- I will be using Gradle >= 4-4 which is a stable Gradle version

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

- Retrofit 2.4.0 - Handles making network calls to API end points. Hides all the dirty details of implementing network calls.
- OKHttp 3.1.10 - Handles the actual network requests for Retrofit.

- SimpleXml Converter 2.4.0 - Handles conversion of XML to POJO.
- Gson 2.4.0 Converter 2.4.0 - Handles conversion of Json data to POJO.
- Butterknife 8.8.2 - Minimizes boilerplate code

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

I plan to use the following Google Play Services:

- Analytics - Will use analytics to track various usage metrics
- Admob - Would like to publish to the store at some point as a free app with ads

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

Write out the steps you will take to setup and/or configure this project. See previous implementation guides for an example.

1. Create new project
2. Configure libraries
3. Add required permissions to manifest
4. Setup any needed api keys so it is being used securely in the project and won’t show in the github repository
5. Implement pojo and retrofit classes and successfully make request and receive response data
6. Create each Activity that is required per the specs listed below in this document
7. Create separate packages and place java class files in their respective package to keep things organized

Task 2: Successfully make network call and parse response data

- Create retrofit interfaces with endpoints
- Create data repository class that creates Retrofit instances then make the network calls
- Log API response for testing

Task 3: Create ViewModel using LiveData

- Create ViewModel Class
- Create LiveData variable that stores a List of Station objects
- Use Repository to make network call then return results
- Use Repository to save API response data into Room database

Task 4: Create Room database

- Configure Room to be used with Station Object Database and LiveData
- Update database based on LiveData

Task 5: Implement UI for Each Activity/Fragment

List the subtasks. For example:

- Build UI for MainActivity/Fragments
- Build UI for AddStationActivity
- Build UI for StationDetailsActivity
- Build UI for SettingsActivity
- Build UI for InfoActivity

Task 6: Setup Observables in activity to pull the Station data

- Create ViewModel object in MainActivity
- Use ViewModel object to call its method that gets the weather data
- Chaining a call to observe
- Pass in the context and a new Observer
- On change, set the Station data to a variable

Task 7: Use Job Scheduler that runs a network call and refreshes data every 15 minutes

- Create JobService class and override the onStartJob and onStopJob methods
- Create service and run service in overridden methods on background thread.
- Register job in the manifest

- Create JobInfo object
- Schedule recurring Job that runs network call and refreshes data.

Task 8: Add Google Play Analytics and Admob

- Configure Google Play Analytics Services.
- Configure Google Play Admob to show an ad at the bottom of screen

Task 9: Create a widget to display station data for chosen station

Describe the next task. List the subtasks. For example:

- Create the a StationWidgetProvider class that extends AppWidgetProvider
- Create the StationWidget class that extends RemoteViewsService. This will also include a separate class called StationRemoteViewsFactory which implements RemoteViewsService.RemoteViewsFactory
- Create a class called StationWidgetService that extends IntentService

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**"