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: tservo

Mood Tracker

Description

This app logs a user's mood, from depressed to manic, for each day, using a UI that is as unobtrusive as possible. The app will graph moods on a biweekly, monthly, 3 month, 6 month, and year frame to allow patterns to be found. There is a corresponding widget that will make it easy to add/update the mood for the current day. A daily reminder around bedtime will be sent if there hasn't been a mood update that day.

Intended User

People with mood disorders, such as Bipolar Disorder, Anxiety, and Depression. Eventually treatment providers can use the reported information for insights on how to treat the user's mood disorder.

Features

List the main features of your app. For example:
Saves mood of user: from depressed, neutral, manic, using a UI that will be natural and easy-to-use/
View/edit mood of a given day up to 30 days previous.
Displays a graph of mood over time from two weeks to 1 year.
Stores the information using Firebase Realtime database, to sync multiple Android devices to the same data.
Offers a reminder at the morning and evening to enter your mood if not already entered.

User Interface Mocks

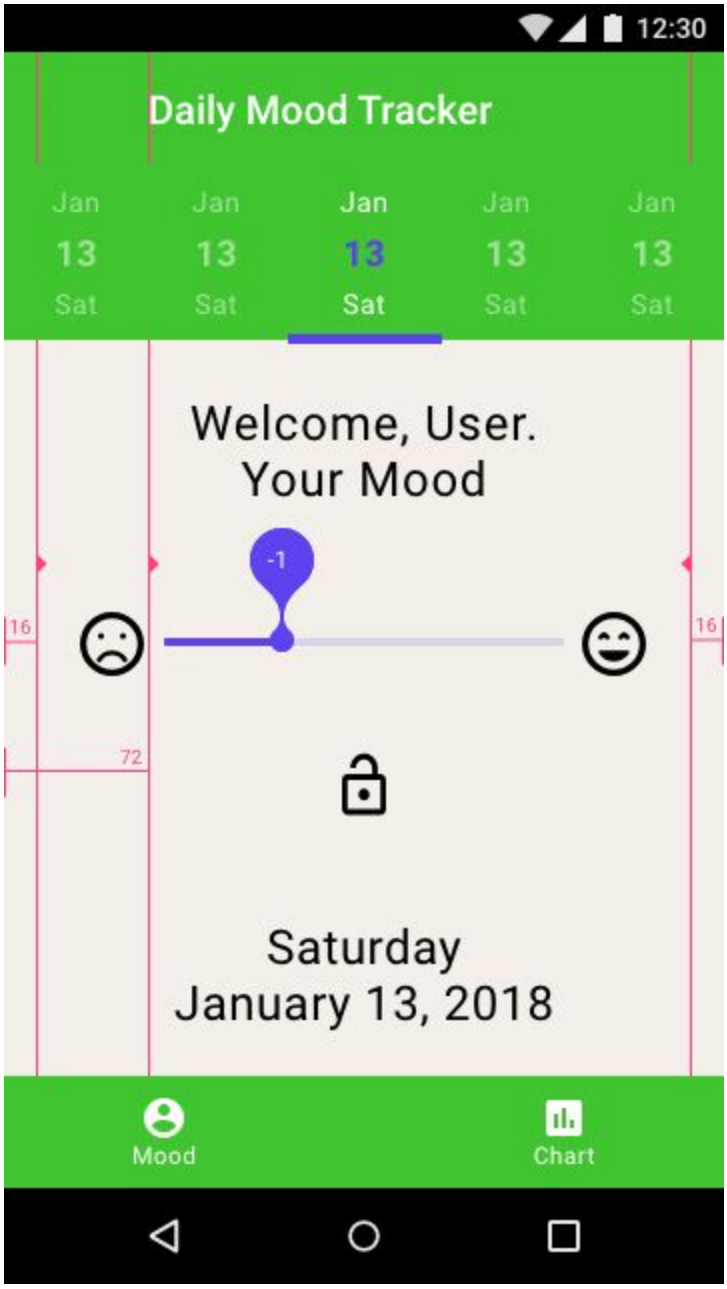

Figure 1: Enter your mood. After signing in, the user
Will end up at a screen that will allow entering/modifying the mood score for the given day.
A horizontal sliding calendar will allow for selecting a day up to 30 days previous.
When the user has logged in a mood score, they can click on the lock icon so that it won't be inadvertently changed by an unintentional swipe. In addition, if a mood has been swiped, moving away from the screen will save and automatically lock the score. The user may unlock the score to change it at any time.

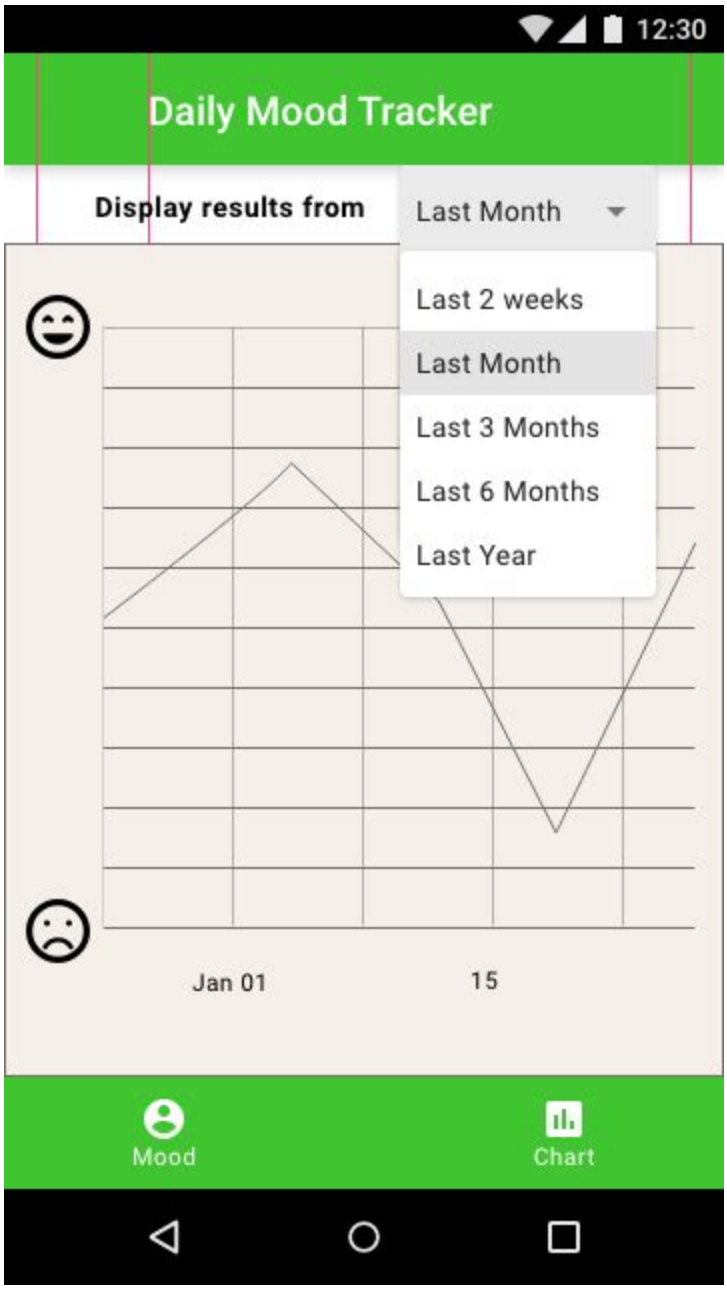

Figure 2: Home Screen, Graph. By clicking on the chart tab,
The user will be presented with a line graph that will show the moods over the specified time frame.

4

Figure 3: Widget View. If a mood has not been logged in the current day, the widget will display the view on the left, with prompts to add a mood. Once the mood is logged, the widget will display descriptive information as to what the level of mood logged is, and prompt with the option to edit the mood.

Key Considerations

The app will keep all strings in a strings.xml file, and enable RTL layout switching for all layouts, for internationalization/localization. The app will have features for accessibility, including content descriptions and navigation using a D-pad. The app will handle error conditions, including lack of network connectivity, gracefully and without crashing.

How will your app handle data persistence?

The app will use a Firebase Realtime Database to hold/sync up data. Will use google sign-in to access mood data across devices per user.

Describe any edge or corner cases in the UX.

The user has not yet logged a mood for the given day. If possible, suggest a score based on previous data / “best guess” method, which will be the previous day’s score. This score will be shown in a low-priority way, emphasizing that it is only a suggestion and that it won’t be counted as a log. This behavior will help the user track their moods relative to the previous day’s / typical pattern.

The user should have maximum ability to change the score, while making it intuitive to “commit” the score, protecting it against accidental swipes. As such, there will be a lock icon on the screen, which will show if the score is “locked” or can be changed. The score will be “locked” and committed to the table when the user explicitly clicks on the lock, or if the user makes it clear that they are not going to alter the score, such as moving the app to the background or switching to the graph view.

There will be the option to display a master-detail layout on tablets, showing both the log screen and the graph screen. If the screen is narrow, preventing the graph from displaying fully, there will be an easy way to scroll the graph so that it is readable.

If no network connectivity, show screen that alerts the user to the issue.
If not logged into google sign-in, prompt the user to do so.
The app requires Firebase Realtime Database for data persistence, so it needs to have access to the network and a valid login.

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

Butterknife will be used to make setting up views more easy.
GraphView (www.android-graphview.org) will be used to show the graph display of mood data.
Horizontal Calendar (https://github.com/Mulham-Raee/Horizontal-Calendar) will be used to scroll through last 30 days of mood results for view/edit.

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

Describe which Services you will use and how.
Firebase Realtime Database: https://firebase.google.com/docs/database/

https://firebase.google.com/docs/auth/android/google-signin
https://firebase.google.com/docs/auth/users
-- this will handle using the user's google account to sign in as a Firebase User

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 project will be entirely done in Java, with the following libraries and software:
Android Studio 3.2.1

Gradle 4.6
Butterknife https://github.com/JakeWharton/butterknife - 9.0.0-rc1
GraphView http://www.android-graphview.org/ - 4.2.2
Horizontal Calendar https://github.com/Mulham-Raee/Horizontal-Calendar - 1.3.4

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
    - Set up tabs for mood and graph fragments
    - Use master/detail pattern for tablets
- Build Fragment for entering in the mood
    - Use horizontal calendar, with the slider to show mood
    - Color the slider appropriately, use appropriate icons, and descriptive text to effectively provide a simple UX to enter the mood.
    -
    - Check how it looks on portrait, landscape, and tablet
- Build Fragment for handling the graph
    - Use GraphView to handle a line graph that will display data over the proper period of time.
    - Be sure that GraphView creates a graph that can be read on narrow displays.
- Build Widget
    - Create Icons/ Suitable color palette for the widget

Task 3: Implement Google Play Services

- Link up Firebase and Signin from dev account, and get working on the main app.
- Handle data persistence with Firebase Realtime Database. Be sure that
- Data can be accessed on multiple devices with a single login.
- Create a gradle install for debug, and release builds.

MainActivity will now be able to store/retrieve moods logged.
Implement the login screen, the ability to store the login in userpreferences, and show a loading/no network screen as necessary.
Mood fragment will be able to record moods to be logged

- The lock icon will be able to “lock” moods, and “unlock” moods for editing.
- Existing moods shown will always default to “locked” Entered moods will be automatically recorded/ “locked” when the user navigates away from the current day’s screen, or after 30 seconds of inactivity. There will be a button to cancel/reset entered moods to the default saved or an empty value.

Graph fragment should be able to read saved data to handle appropriate periods of time.

Task 4: Widget, Firebase JobDispatcher, and Notifications

- Widget will now link to MainActivity, with the Mood fragment focused on the current day.
- Widget must be able to access the persistent data and know if there’s a new mood entered, or if the day has changed, thus requiring a new mood for that day. Likely this will be done with JobDispatcher.
- Notifications will be controlled by JobDispatcher, checking if it is at a certain time the user wants to be notified, and that there has been no mood entered in already.

Task 5: Clean up unused resources!

https://stackoverflow.com/questions/39931519/how-to-clean-up-unused-resource-files-in-android-studio

1. `Analyze` > `Inspect Code` and find **Unused Declarations and Methods**.
2. Android Studio -> **Menu** -> **Refactor** -> **Remove Unused Resources**.

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**”