GitHub Username: David-Jackson

Active Journal

Description

This app will provide a simple method of recording location-based activities into a journal. These activities will include walking/running, biking, hiking, driving and sailing/boating. In addition to location, the journal will also include the ability to add text and photo entries as well as statistics pertaining to the activity (speed, total distance, inclination, etc.). This app will give users an alternative to record their activities while seamlessly incorporating location data.

Intended User

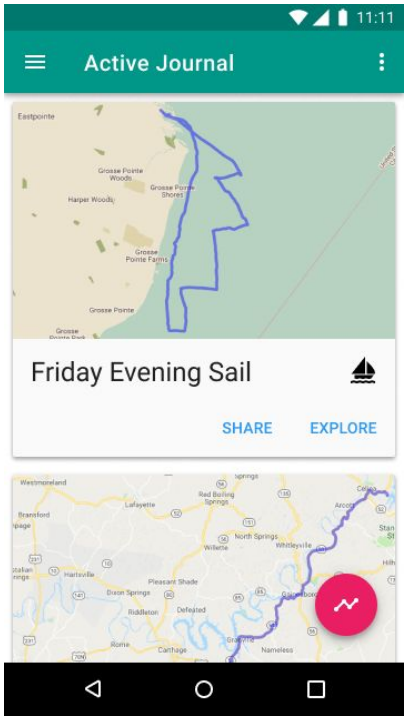
This app will be intended for people who like to travel or like being outdoors and want a simple way of recording their activities.

Features

- Logs location during activities.
- Allows users to add photos and text during or after the activity.
- Allows users to import past activities from Google Fit API.
- Allows users to view past activities.
- The app will pull static map images with path data from the Google Maps Static API on a per request basis and will use an Intent Service to do so.
- The app will be solely written in the Java Programming Language.
- The app will retrieve all hardcoded strings from the strings.xml file.
- The app will be designed with RTL layout switching to support accessibility on RTL supported languages.

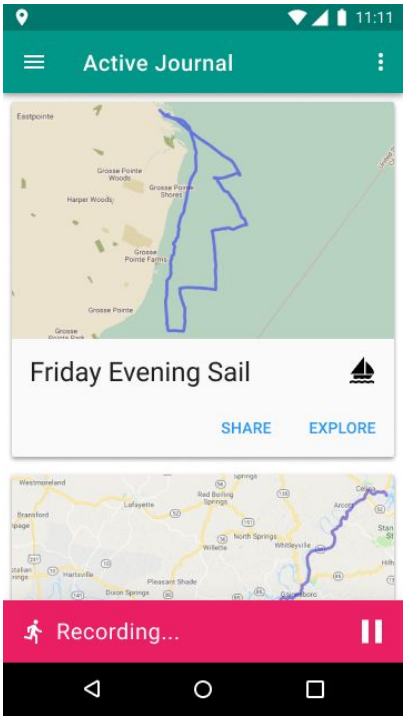
User Interface Mocks

Start Screen



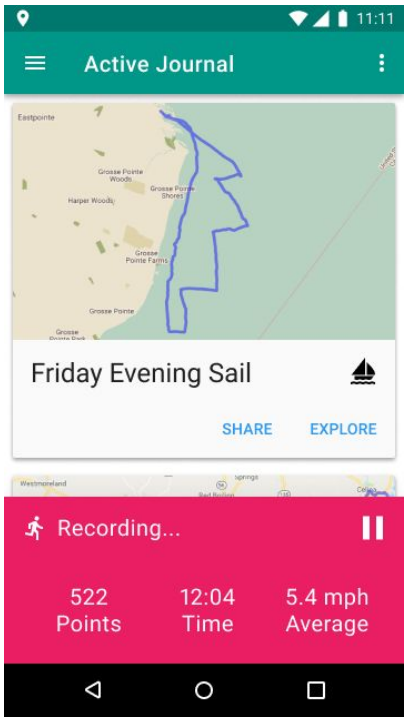
This is the launcher activity for the app. When a user starts the app, they will be greeted with a list of their journal entries as well as a floating action button to start recording a new activity. When it is the user uses the app for the first time, there will be an example entry that the user can use to learn the app's functionalities.

Recording Screen



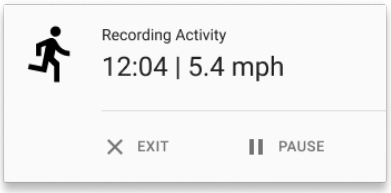
When the user presses the floating action button, it will morph into this bottom sheet that persists throughout the app while recording. Pressing the pause button stops recording.

Expanded Recording Screen



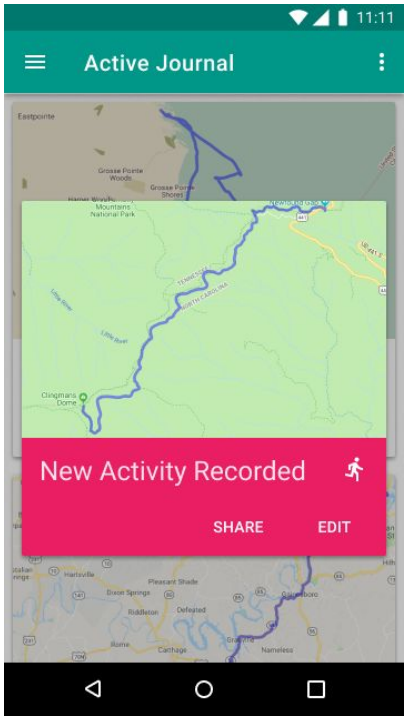
When the user presses or swipes up on the bottom sheet, it will give additional information about the activity.

Foreground Notification



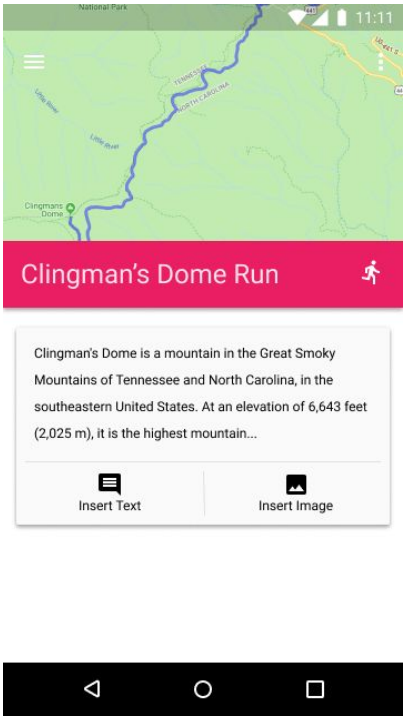
When the user navigates away from the app while recording, a foreground notification will be shown, giving information and options about the activity.

New Trip Screen



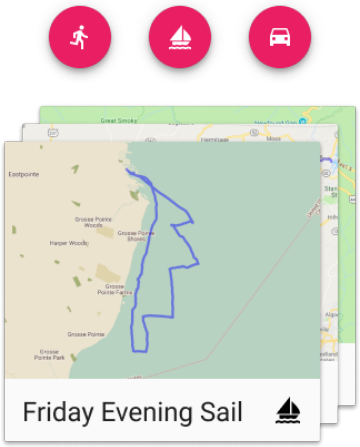
When the user stops an activity, a card will be shown giving the user options to edit or share the trip.

Edit Trip Screen



When the user edit an activity, a screen will be shown that allows the user to add text and images to an activity.

App Widgets

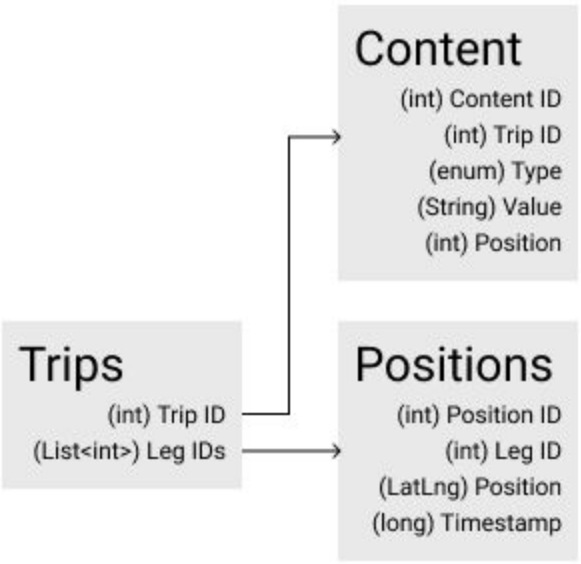


There will be two different options for app widgets. The first type will be a button to quickly start recording an activity. The second type will be a ViewFlipper of the user's previous activities.

Key Considerations

How will your app handle data persistence?

Data will be persisted by default on the device using Room (accompanied by LiveData and ViewModel). Here is an example DB Schema:



For the Content Table, the type column will indicate whether the content is text or an image. The idea for the detail screen is to have a RecyclerView with two different ViewHolders, one for text and one for images. Then the user will be able to insert text and images throughout the entry, with the position column of the content dictating the location in the RecyclerView.

Describe any edge or corner cases in the UX.

One edge case for the UX is what to do when the app loses focus while recording an activity. In this case, a foreground notification will be shown to let the app continue running and let the user control the app from the notification drawer.

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

The app will use Picasso as an image loading library to load images returned by the Google Maps Static API. Butterknife will be used to simplify view binding. Android Jetpack libraries (Room, LiveData, ViewModel) will be used for simplifying boilerplate code. Here is a summary of the library and tool versions that will be used:

Library/Tool	**Version**
Android Studio	3.1.4
Gradle	3.1.4
Jetpack (Room, LiveData, ViewModel)	1.1.1
Picasso	2.71828
Butterknife	8.8.1

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

The Location API will be used to listen for changes in device location while recording activities. Google Maps Android SDK will be used for map viewing, polyline path encoding, and distance calculations. Firebase Analytics will also be used to track app performance and user engagement. The Google Fit API might be used to import and export pastactivities to the app.

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

- Configure libraries.
- Create Project/API Keys/google_services.json in Google API Console.

Task 2: Build the Backend

- Setup Room database for backend storage of activities and content.
- Fill database with sample data to be used in creation of UI.

Task 3: Implement UI for Each Activity and Fragment

- Build UI for MainActivity.
- Build UI for Morphing Floating Action Button.
- Build UI for List Viewholders.
- Build UI for Detail Activity.

Task 3: Implement ViewModel

- Setup the app with ViewModel and LiveData for backend data.
- Look into the possibility of using LiveData to connect the foreground service and the UI.

Task 4: Build Foreground Service

- Build Foreground Service for recording location.
- This includes a Foreground Notification.
- Hook Service to Morphing Floating Action Button when app is visible.

Task 5: Implement Main Activity interactions

- Build logic for displaying activities.
- Build logic for navigating app.

Task 6: Implement Edit Activity interactions

- Build insert new content (text/image) logic.
- Build logic for arranging content.

Task 7: Implement Widgets

- Build widget for starting activity recording.
- Build widget for flipping through previous activities.

Task 8: Implement Activity Importing

- Connect to Google Fit API.
- Build a converter from the Google Fit format to the app's data format.