Family Map Client Specification

Last Modified: January 25, 2019

Contents

Acknowledgements **4**

Introduction **4**

- Purposes 4

Family Map Client: A Quick Overview **4**

Activities and Fragments **5**

- Main Activity 5
- Login Fragment 6
  - Layout 6
  - Functionality 6
- Map Fragment 7
  - Layout 7
  - Functionality 8
- Person Activity 8
  - Layout 9
  - Functionality 9
- Event Activity 10
  - Layout 10
  - Functionality 11
- Settings Activity 11
  - Enable/Disable Relationship Lines 11
    - Spouse Lines 12
    - Family Tree Lines 12
    - Life Story Line 12
  - Filters 12
    - Filter by Gender 12
    - Filter by Side 13
  - Logout 13
  - Up Button / Options Menu 13
- Search Activity 14
  - Layout 14
  - Functionality 15

Additional Information **15**

Extra Credit **15**

- Properly Handle Device Rotation (1% extra credit) 15

Make Settings Persistent Across an App Restart (1% extra credit) 15
Properly Handle Multiple Markers at the Same Location (3% extra credit) 16

Resources **16**
Family Map Server 17
Map Libraries for Android 17
Drawing Icons 18
JSON Parsing 19

Automated Tests **19**

Source Code/Test Case Evaluation **20**

Grading **20**

Acknowledgements

The Family Map project was originally created by Jordan Wild. Thanks to Jordan for this significant contribution.

Introduction

Family Map is an Android application that provides a geographical view into your family history. One of the most exciting aspects of researching family history is discovering your origins. Family Map provides a detailed view into where you came from.

Purposes

The purposes of this project are to dig a little deeper into the following:

- Object-Oriented Design
- User Interface Programming
- Native Android Development
- Calling external services through Web APIs
- Unit Testing

Family Map Client: A Quick Overview

The Family Map Android application consists of five main views:

- Main Activity (Login and Top-level map)
- Event Activity (Lower-level maps)
- Person Activity
- Settings Activity
- Search Activity

The Family Map Client application uses two external services:

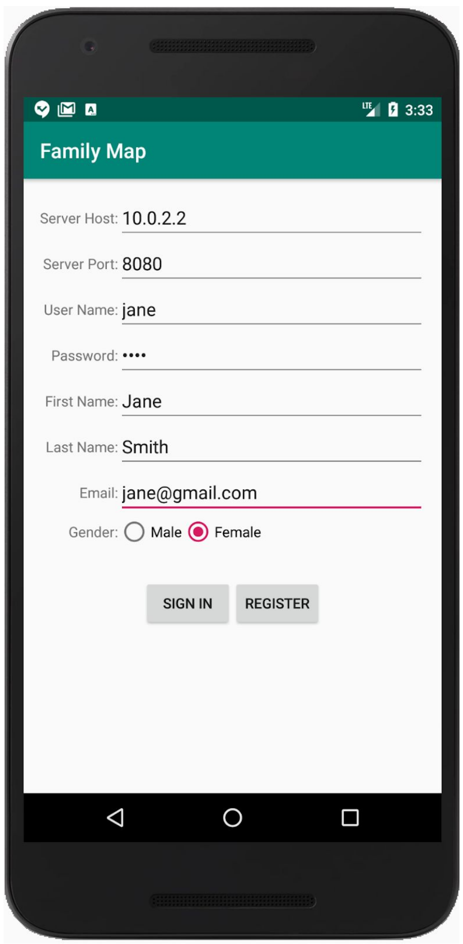
- Family Map Server - Used for user management, data generation, and requesting data (this is the Family Map Server you created in the previous project)
- Google Maps v2 for Android or Amazon Maps v2 for Kindle - Used for displaying maps

Activities and Fragments

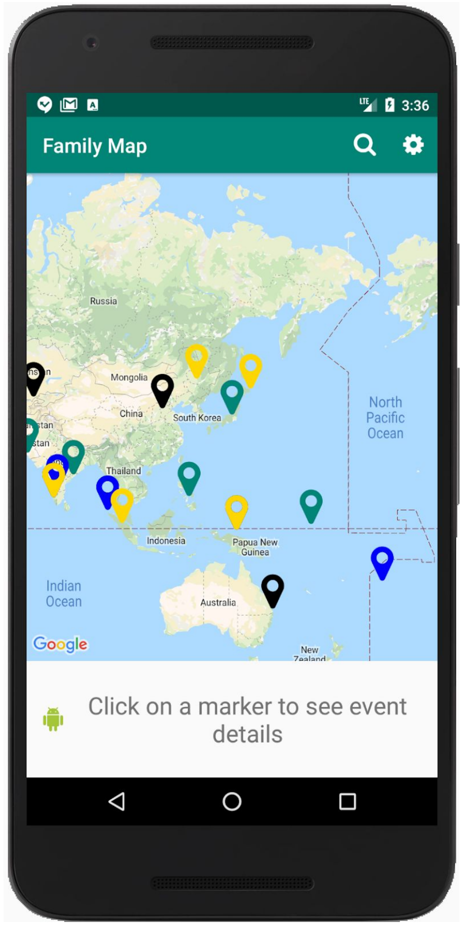
The following subsections describe the layout and functionality of each of the activities and fragments that make-up the Family Map Client app.

Main Activity

When the application runs, it displays the Main Activity. Before the user logs in, the Main Activity displays a Login Fragment, as shown below. The Login Fragment allows the user to login if they already have a user account, or to register to create a new user account.



After successful sign-in or registration, the user is logged in, and the Main Activity switches from the Login Fragment to the Map Fragment, which displays markers for the events in the user's family history, as shown below.



Login Fragment

Layout

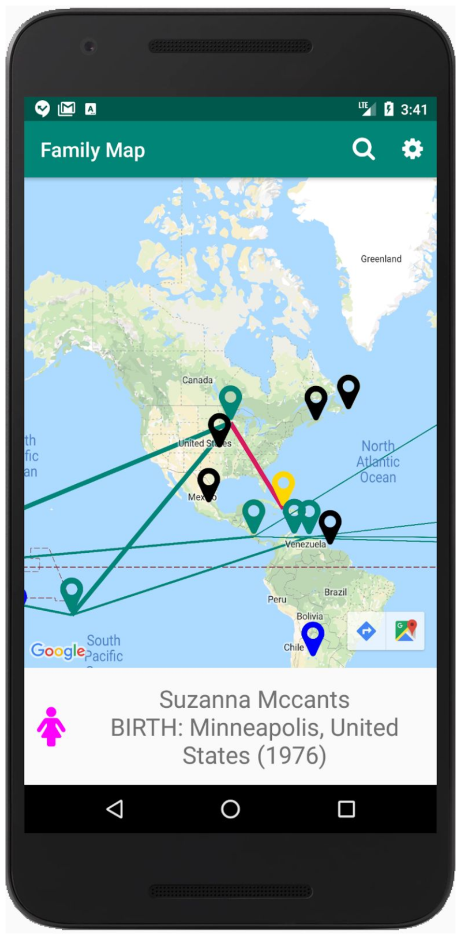
On the Login Fragment's layout, there are editable text fields where users can enter their server hostname / IP address, server port number, user name, password, first name, last name, and email address. There are also two radio buttons that allow users to specify their gender. Additionally, there are two buttons that allow the user to initiate sign-in with an existing user account or creation of a new user account. The Login Fragment does not display an up button or an options menu.

Functionality

In order to log in or register, the user must specify the server host and server port, which allows the app to contact the Family Map Server. To log in to an existing user account, the user must specify user name and password values, and click the "SIGN IN" button. The "SIGN IN" button is disabled unless the server host, server port, user name, and password values are filled in. To create a new user account and then log in to the new account, the user must specify values for all fields, and then click the "REGISTER" button. The "REGISTER" button is disabled unless all field values are filled in. If the sign-in / registration operation succeeds, the Login Fragment synchronizes the application with data from the Family Map Server. Synchronization pulls the data from the Family Map Server and stores it so the application can access it later. If sign-in / registration or data synchronization fails, the Login Fragment displays an error message (i.e., Android "toast"), and allows the user to retry the operation. If sign-in / registration and synchronization succeed, the Main Activity displays a Map Fragment (described next).

Map Fragment

The Map Fragment takes data from the Family Map Server and displays it on an interactive map. Data is displayed according to the application's current settings (see the Settings activity described later for more details).



Layout

On the Map Fragment's layout, there is an interactive map, and a place for event information to be displayed. When displayed in the Main Activity, the Map Fragment does not display an up button, but it does have an options menu that contains Search and Settings icons/options.

Functionality

The options menu items allow the user to navigate to the Search Activity and Settings Activity. The interactive map allows the user to zoom in and out, select events from their family history, and view information about those events. Events are displayed on the map by markers.

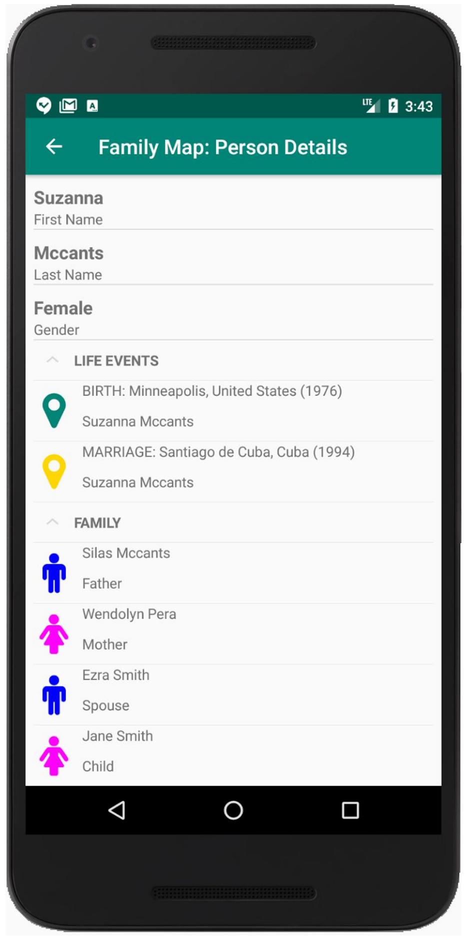
The markers are color coordinated based on event type and dynamically assigned. This means that each event with the same event type have the same colored icon. You can not assign all event types the same color and you can not just have a default color for events you do not recognize. Due to the limited number of pre-made colors in Android you may reuse colors for different event types after exhausting the list of colors. The markers are clickable, updating the event information display when selected. In cases where you have multiple events at the same location, the last marker placed at that location will cover all the other markers at that location. That is fine for this assignment. However, you can earn extra credit for properly handling situations where multiple event markers appear at the same location. See the Extra Credit section for details.

The map also displays lines that represent relationships between events. The Map Fragment draws the line types that are enabled in the Settings Activity (i.e., life story, spouse, and family tree lines). The Map Fragment filters events based on filters defined and enabled in the Settings Activity. See the Settings activity section for more details.

The event information display shows information about the selected event including the person's gender, name, event description, and event date. When the event information display is clicked, the Person Activity is created to display information about the person associated with the event (described next).

Person Activity

The Person Activity displays information about a person, including first name, last name, gender, and information about the person's family members and life events.



Layout

On the Person Activity's layout, there is a line for every descriptive field about a Person. There is also a collapsible list that displays the person's parents, spouse, and children. There is another collapsible list that displays a list of all events (that aren't being filtered) in the person's life story, listed in chronological order. There is an up button and no options menu.

Your events and family collapsible lists must be implemented using a single [ExpandableListView] component.

A person's life events are ordered chronologically as follows:

1. Birth events, if present, are always first (whether they have a year or not)
2. Events sorted primarily by year, and secondarily by event type normalized to lower-case
3. Death events, if present, are always last (whether they have a year or not)

Functionality

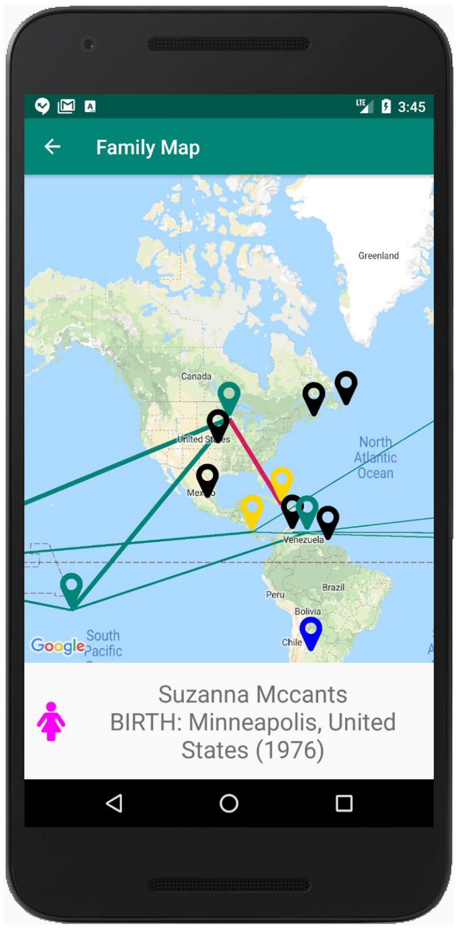
Clicking on a person in the relationship list triggers the creation of a new Person Activity with the information of the selected person. Clicking on an event triggers the creation of the Event

Activity, with the map zoomed in and centered on the location of the clicked-on event (described next).

Clicking the up button returns the user back to the top-level Main Activity. (Note: the up button does not open a new Main Activity, but rather returns to the original Main Activity by closing the current activity and all other intermediate activities that led to it.)

Event Activity

When the user clicks on an event in the Person or Search activities, an Event Activity is created to display information about the selected event. The Event Activity uses the same Map Fragment as the Main Activity to display an interactive map of events. However, when an Event Activity is created, the event clicked on by the user in the Person or Search activity is automatically selected and centered on the map, as shown below.



Layout

The Event Activity layout contains a Map Fragment that displays an interactive map and a display that shows information about the currently-selected event. When displayed in the Event Activity, the Map Fragment displays an up button and no options menu.

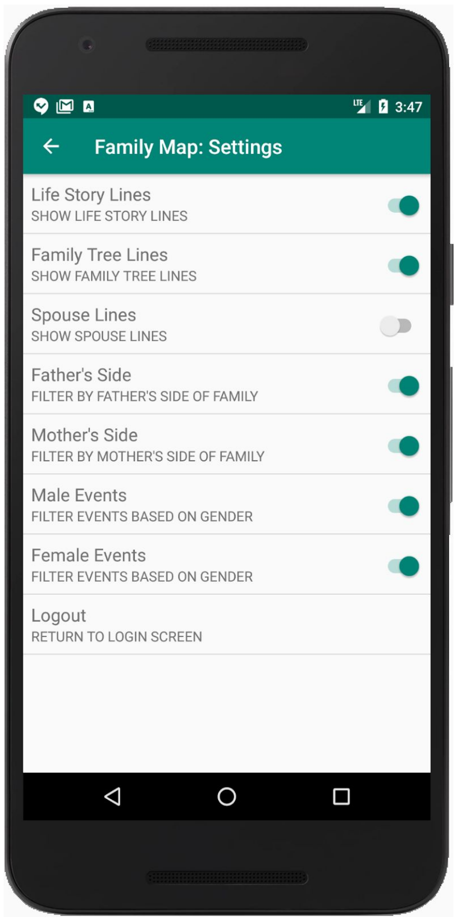
Functionality

The functionality of Event Activity is the same as that of Map Fragment described previously, with the added requirement that the event being viewed is automatically selected and centered on the map.

Clicking the up button returns the user back to the top-level Main Activity. (Note: the up button does not open a new Main Activity, but rather returns to the original Main Activity by closing the current activity and all other intermediate activities that led to it.)

Settings Activity

The Settings Activity allows the user to manage their account and modify certain aspects of the map's appearance. On the Settings Activity, there is a line for every actionable setting, as shown below.



Enable/Disable Relationship Lines

Enable and disable the types of relationship lines to be drawn on the map. Each line has a different color. The types of relationship lines are defined as follows:

Spouse Lines

A line is drawn linking the selected event to the birth event of the person's spouse (i.e., the person associated with the selected event). If there is no birth event recorded for the spouse, the earliest available event for the spouse is used instead. If the person has no recorded spouse, or the recorded spouse has no events, no line is drawn.

Family Tree Lines

Lines linking the selected event to the person's ancestors (i.e., the person associated with the selected event) are drawn as follows:

- A line is drawn between the selected event and the birth event of the selected person's father. If the person's father does not have a birth event, the earliest available event for the father is used instead. If the person has no recorded father, or the recorded father has no events, no line is drawn.
- A line is drawn between the selected event and the birth event of the selected person's mother. The same logic that applies to the father also applies to the mother.
- Lines are drawn recursively from parents' birth events to grandparents' birth events, from grandparents' birth events to great grandparents' birth events, etc. including all available generations. In all cases, if a person's birth event does not exist, the earliest event in the person's life should be used instead of their birth event. <u>As lines are drawn recursively up the family tree, they should become progressively and noticeably thinner</u>.

Life Story Line

Lines are drawn connecting each event in a person's life story (i.e., the person associated with the selected event), ordered chronologically. (See the Person Activity section for information on how events are ordered chronologically.)

Filters

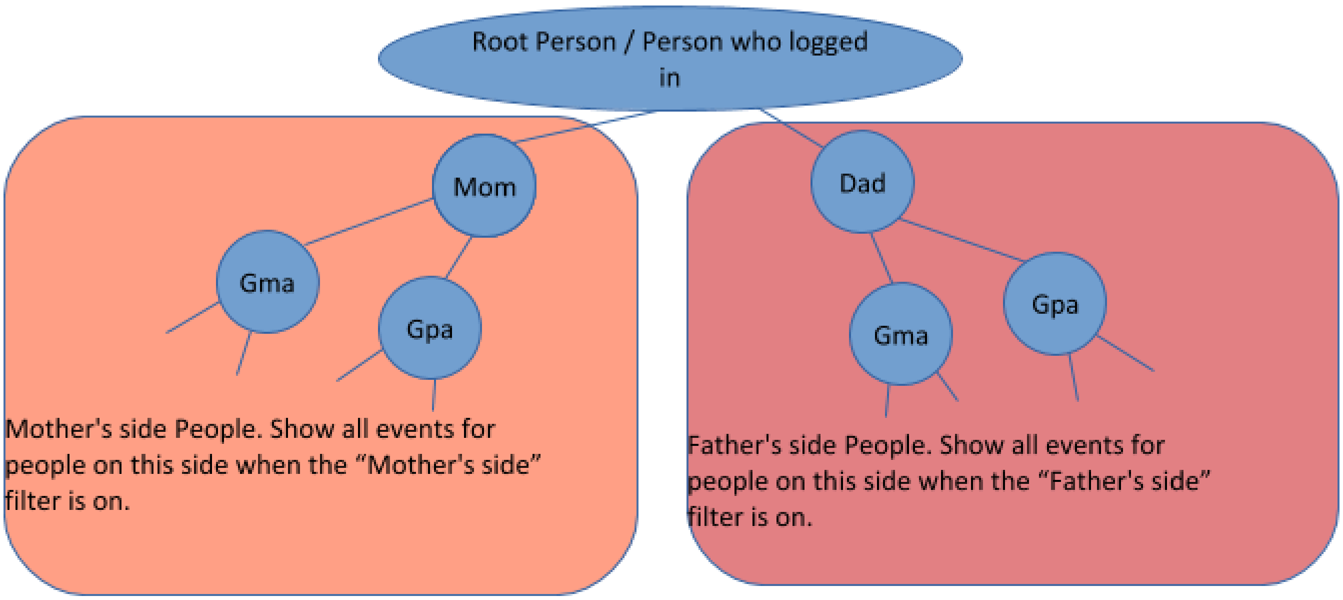
Filters settings allow the user to select which filters are enabled and disabled, as shown below. Changing the filters changes which events are drawn on the map, shown in search results (search activity) and shown in the person's details (person activity). Events that are filtered should not appear on the map in MapFragment, in a search result in Search Activity, or as a life event in Person Activity. Filter settings control only which events are visible and have no effect on which people are visible. For example, if female events are being filtered, females would still appear in the search results of the search activity, but events associated with females would not appear. The following filters are supported:

Filter by Gender

These filters allow the user to filter events based on gender. The map updates to only show events associated with people of the selected genders. By default, the map shows events associated with both genders.

Filter by Side

These filters allow the user to select which sides of their family will be displayed (Mother's side and/or Father's side). To determine "Mother's side" and "Father's side", start with the person who has logged in. That person is the root of the tree and all ancestors from that person's mother's side are classified as "Mother's side". Ancestors from the logged in user's father's side are classified as "Father's side". When someone logs in, one of the items returned in the JSON is the Person Id of the person who just logged in. This is the Person Id of the root of the tree. (See picture below). By default, the map shows both sides of the family.



Logout

Logging out destroys the user's session and returns them to the Main Activity. When it is re-displayed, the Main Activity presents the Login Fragment so the user can re-login if they desire.

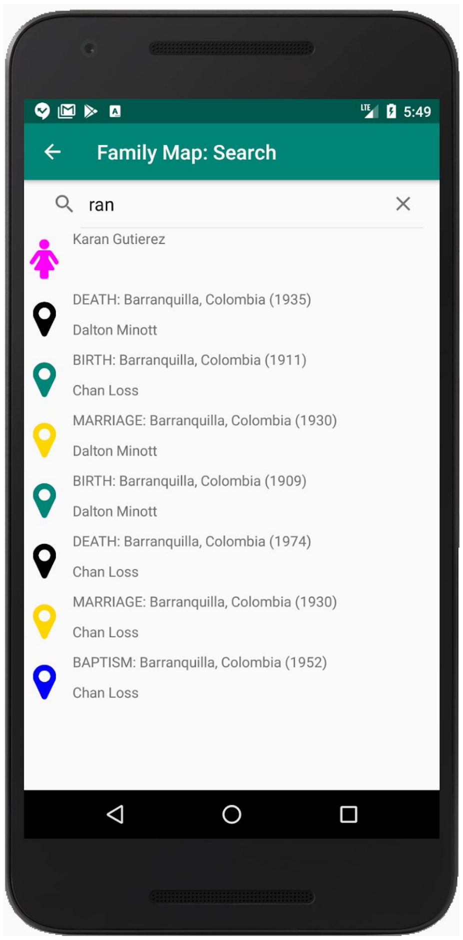
Up Button / Options Menu

The Settings Activity has an up button, which returns the user to the Main Activity. The Settings Activity has no options menu.

In a production Android app, the settings for map type and relationship lines would need to remain persistent across an app restart. However, to simplify this assignment and make it easier to complete within the available time, making the settings persistent is not required but can be done for extra credit. See the Extra Credit section for details.

Search Activity

The Search Activity allows the user to search for people and events that match a specified search string. The user can enter a single search string. The specified string is searched for in people's first and last names, and in events countries, cities, event types, and years. Case is ignored. All matching people and events (that aren't filtered) are listed in the search results. A search string may match in any part of a string. For example, a search string of "ran" would match a first name of "Karan" and events that occured in "Barranquilla" as shown below.



Layout

The Search Activity's layout displays a search input that accepts the user's search string. It also displays a list of search results, which is empty before a search has been executed. The results list displays a combination of both people and events, with the people items coming before event items. The people items display a person's name and gender. The event items display an event's type, city, country, date, and the associated person's name. Search results must be implemented with a RecyclerView component.

The Search Activity also has an up button, which returns the user to the Main Activity. It has no options menu.

Functionality

The search result list is empty until a search has been executed. When the user executes a search, the search result list is populated with all people and events that match the search string (and are not being filtered). The latest search result remains visible until another search is executed, or the activity is closed.

When the user selects a person from the search results list, a new Person Activity that contains the selected person's information is displayed. When the user selects an event from the search results list, a new Event Activity is displayed in which the event is automatically selected and the map is centered on the event. The up button returns the user to the Main Activity.

Additional Information

Your Map Fragment, Person Activity, Event Activity, and Search Activity need to support (be able to display) information for a spouse of the logged-in user but not children of the logged-in user.

Extra Credit

In some cases, the project specification does not require functionality that would be required of a production-ready Android app. We allow this missing functionality as a way to make the project easier to complete in the time available within a single semester or term. The subsections below describe extra credit opportunities for adding this missing functionality.

Properly Handle Device Rotation (1% extra credit)

Some of your activities will lose their current state when you rotate your Android device. This would not be acceptable for a production-ready Android app. You may make all of your activities remember (or re-create) their state across a device rotation for an extra 1% which will be added to your client pass-off score. This will be an additional 1% of the total points available for pass-off, not 1% of the points you earned.

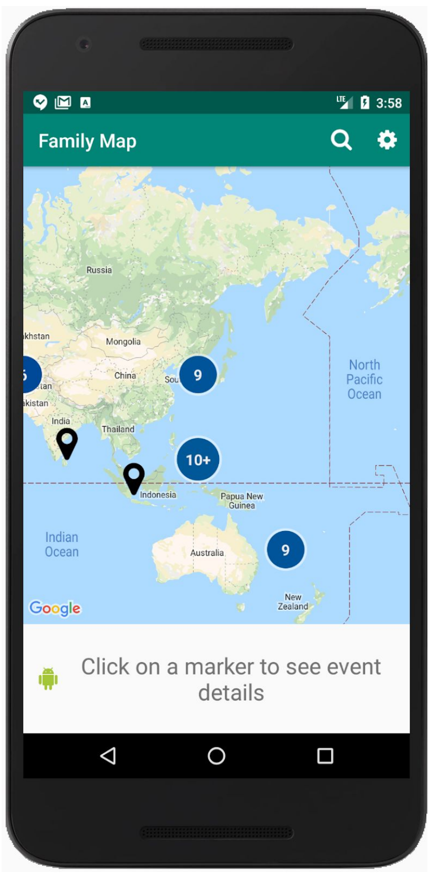
Make Settings Persistent Across an App Restart (1% extra credit)

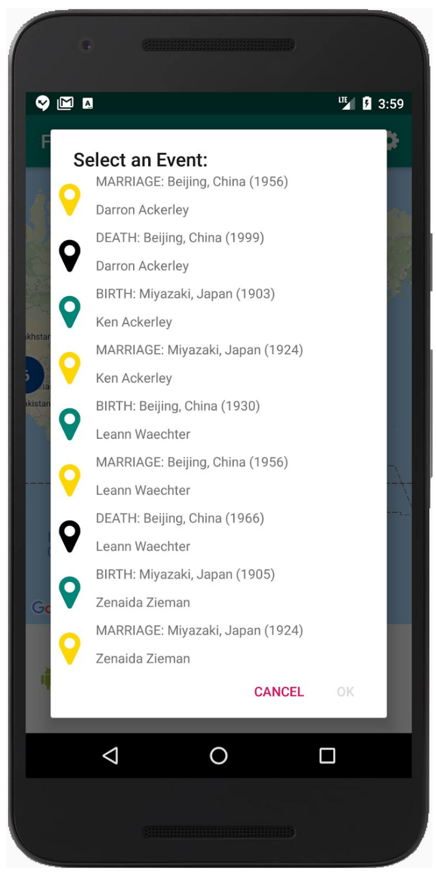
In a production-ready Android app, the settings the user specifies in the settings activity for whether the lines are on or off and the filters should be remembered across a restart of the app. This would mean that whatever settings a user specifies should still be set to those values if the app is stopped and restarted, or even re-deployed from Android Studio. Make these settings persistent for an extra 1% on your pass-off score.

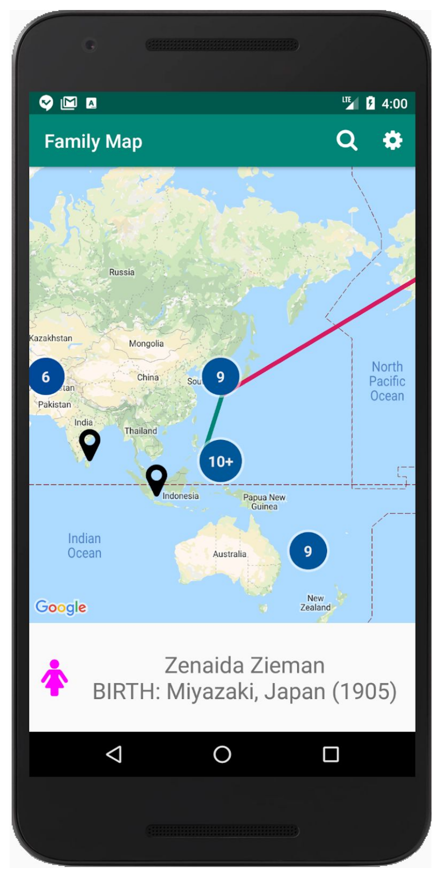
Hint: Consider using the Android SharedPreferences class.

Properly Handle Multiple Markers at the Same Location (3% extra credit)

It is possible to have multiple events occur at the same map location. By default, the last marker added to a location will cover all the others, making only the last marker at that location visible and selectable. For 3% extra credit, place a different kind of map marker at locations that have multiple events at the same location. This marker should indicate how many events are at that location. When a marker indicating multiple events at a location is clicked, display a dialog with a scrollable list of events at that location. After the user selects an event in the dialog and closes the dialog, the selected event should be selected on the map fragment (with its information displayed in the selected event area of your map fragment). This is illustrated in the following three images.







Hint: Consider using the Google Maps Android Marker Clustering Utility.

Resources

The following resources will be helpful in developing your Family Map Android application.

Family Map Server

You will need to use the Family Map server that you created in the previous project. The following information will help you connect your client and server.

IMPORTANT POINTS FOR CONNECTING YOUR CLIENT AND SERVER

1. The Android device and the machine running the server should be on the same WiFi network. (We have seen in the Talmage building there tends be to an issue connecting while using BYUSecure and BYUGuest, but only in the Talmage Building. When in the Talmage building we recommend using CSDept).

2. Make sure to add the following permissions to the AndroidManifest.xml file to allow the Android device to access the network.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

   These should be added just before the <application> entry in AndroidManifest.xml.

3. When testing your app, you will normally run both your client and your server on the same machine, using an Android emulator for your client. In this case, your emulator will function as a separate machine and as a result, using a hostname of “localhost” or an ip address of 127.0.0.1 from your Android login fragment will refer to the emulator--not the host machine that will also contain your running Family Map Server. To access a server running on the same machine as an Android emulator, use the ip address 10.0.2.2 instead of localhost or 127.0.0.1.

Map Libraries for Android

To help build the Map Fragment, you will use the Google Maps for Android library. This library provides an Android UI fragment that can display maps retrieved from Google Maps. The Google map fragment allows you to draw markers and lines on a map using latitude and longitude coordinates, and handles panning and zooming of maps, etc. Information on how to use the Google map fragment can be found here:
https://developers.google.com/maps/documentation/android-api/start

If you are developing for an Amazon Kindle device, instead of using the Google Maps for Android Library, you will use the Amazon Maps for Android (version 2) library. The Amazon Maps library is patterned after the Google Maps library, so the two are very similar. Information on how to use the Amazon Maps library can be found here:

https://developer.amazon.com/public/apis/experience/maps/docs-v2/configuring-your-project-to-use-the-amazon-maps-api-v2

Note: Google Play Services must be installed on an Android device in order to use the Google Maps library. Google Play Services is not installed on Amazon Kindle devices, and therefore you must use the Amazon Maps library on Kindle. However, with some effort it is possible to install Google Play Services on Kindle, thus allowing use of the Google Maps library on Kindle. If you want to do this, Google for details on how to do so.

Drawing Icons

Android Iconify is a library that makes it easy to create and draw dynamically-customizable icons in an Android application. This library is useful for drawing the following icons required by the Family Map application:

- Male and female gender icons
- Event icon
- Search and Settings
- Other icons you wish to incorporate in your UI

The web page for the Android Iconify library can be found here:
https://github.com/JoanZapata/android-iconify

To use Android Iconify, you should include a dependency similar to the following in the **build.gradle** file in your project's **app** folder:

```
dependencies {
   …
   implementation 'com.joanzapata.iconify:android-iconify:2.2.2'
   implementation 'com.joanzapata.iconify:android-iconify-fontawesome:2.2.2'
   …
}
```

In your MainActivity's onCreate method, initialize the Iconify library as follows:

```
import com.joanzapata.iconify.Iconify;
import com.joanzapata.iconify.fonts.FontAwesomeModule;

onCreate() {
      …
      Iconify.with(new FontAwesomeModule());
      …
}
```

You can easily create an Android drawable for an icon with code similar to the following:

```
import com.joanzapata.iconify.IconDrawable;
import com.joanzapata.iconify.fonts.FontAwesomeIcons;
…
Drawable genderIcon = new IconDrawable(getActivity(), FontAwesomeIcons.fa_male).
                                        colorRes(R.color.male_icon).sizeDp(40);
genderImageView.setImageDrawable(genderIcon);
```

The constant values for the necessary icons are:

- fa_male (male icon)
- fa_female (female icon)
- fa_map_marker (event icon)
- fa_search (search icon)
- fa_gear (settings icon)

JSON Parsing

You may use any library you wish for parsing and generating JSON data. If you choose to use the GSON library, you should include a dependency similar to the following in the **build.gradle** file in your project's **app** folder:

```
dependencies {
   …
   implementation 'com.google.code.gson:gson:2.8.0'
   …
}
```

Automated Tests

Write JUnit tests to verify that your Server Proxy class works correctly. Specifically, you should test your Server Proxy methods for logging in, registering a new user, retrieving people, and retrieving events. For each method, you should test both success and failure cases.

Write JUnit tests to verify that your model correctly does the following:

- Calculates family relationships (i.e., spouses, parents, children)
- Filters events according to the current filter settings
- Chronologically sorts a person's individual events (birth first, death last, etc.)
- Correctly searches for people and events

In situations where “failure” cases do not seem to apply, think of normal versus abnormal cases and have at least one normal case test and one abnormal case test. For example, when testing the search functionality, have at least one test where a search returns results and one where a search returns no results.

Source Code/Test Case Evaluation

After you pass off your project with a TA, you should immediately submit your project source code for grading. Your grade on the project will be determined by the date you submitted your source code, not the date that you passed off. If we never receive your source code, you will not receive credit for the assignment. Here are the instructions for submitting your project source code:

1. In Android Studio, execute “Build -> Clean Project” (if this option is not available, you may skip this step).
2. Create a ZIP file containing ALL of your project’s files (not just the Java files)
3. The name of the ZIP file should be your NetID. For example, if your NetID is “bob123”, the name of your zip file should be “bob123.zip”.
4. Submit your ZIP file through Learning Suite

To demonstrate that your test cases execute successfully, you should run your unit tests inside Android Studio, and make a screenshot of the successful test results displayed by Android Studio. Submit your screen shot through Learning Suite. This screenshot is submitted separately from your code ZIP file (i.e., they are different assignments in Learning Suite).

The following criteria will be used to evaluate your source code:

- (25%) Effective class, method, and variable names
- (30%) Effective decomposition of classes and methods
- (30%) Code layout is readable and consistent
- (15%) Effective organization of classes into Java packages
- You are required to use a jar or a shared module for your shared classes (domain, request and result classes)

The TAs will also do a code review of your test cases in order to assess their quality.

Grading

Your grade on this project will consist of the following components:

1. Functionality (how well your app works)
2. Test Case Quality (quality of your test cases)
3. Source Code Quality (quality of your source code)

Note: The grade for your source code and tests is capped at the score you receive for your pass-off. For example, if you received 82% of the available points during your client pass-off (not including extra credit), your source code and test case scores will not exceed 82%.

See the course policies page on the CS 240 website for details on how much each of these components counts toward your overall course grade.