FamilyMap Specification

Contents

Acknowledgements

The FamilyMap project was created by Jordan Wild. Thanks to Jordan for this significant contribution.

Introduction

FamilyMap 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. FamilyMap 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
- Restful Web Services
- Using API's from third party services
- Unit Testing

FamilyMap: A Quick Overview

The FamilyMap Android application consists of six main views:

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

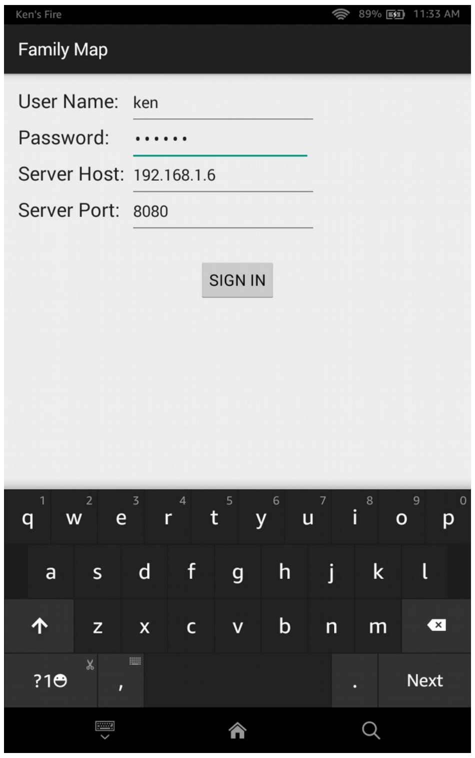
The FamilyMap Application uses two third party services:

- FamilyMap Service - Used for user management and requesting data (See FamilyMap Service under Resources for more information)
- Amazon Maps v2 for Kindle (similar to Google Maps v2 for Android) - Used for displaying maps

Activities

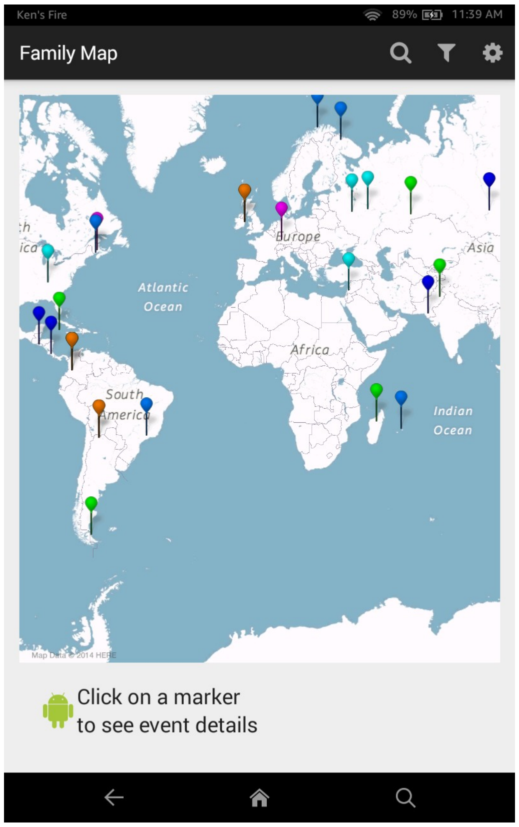
Main Activity

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 using credentials previously set up on the FamilyMap Service.



User Name: ken
Password: ••••••
Server Host: 192.168.1.6
Server Port: 8080

After the user logs in, the Main Activity displays a Map Fragment, which displays markers for the events in the user's family history, as shown below.



Login Fragment

Layout

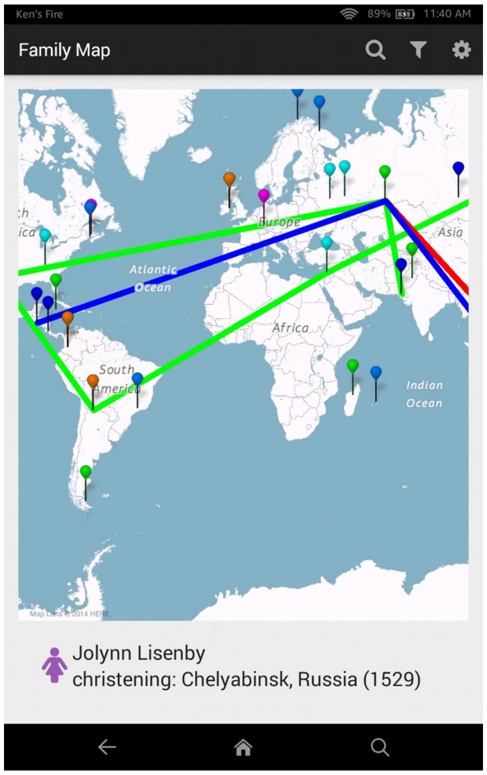
On the Login Fragment's layout, there are four editable text fields where users can enter their user name, password, the FamilyMap Service host name / IP address, and the FamilyMap Service port number. There is also a button allowing the user to submit their credentials to the FamilyMap Service. The Login Fragment does not display an up button or an options menu.

Functionality

The Login Fragment handles logging the user in to the FamilyMap Service. Upon logging the user in, the Login Fragment synchronizes the application with data from the FamilyMap Service. Synchronization pulls the data from the FamilyMap Service and stores it so that the application can access it later. If login or synchronization fails, the Login Fragment displays an error message (i.e., Android "toast"), and allows the user to re-attempt login. If login and synchronization succeed, the Main Activity displays a Map Fragment (described next).

Map Fragment

The Map Fragment takes data from the FamilyMap Service and displays it on an interactive map. Data is displayed according to the application's current settings and filters (see the Settings and Filter activities 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, Filter, and Settings icons/options.

Functionality

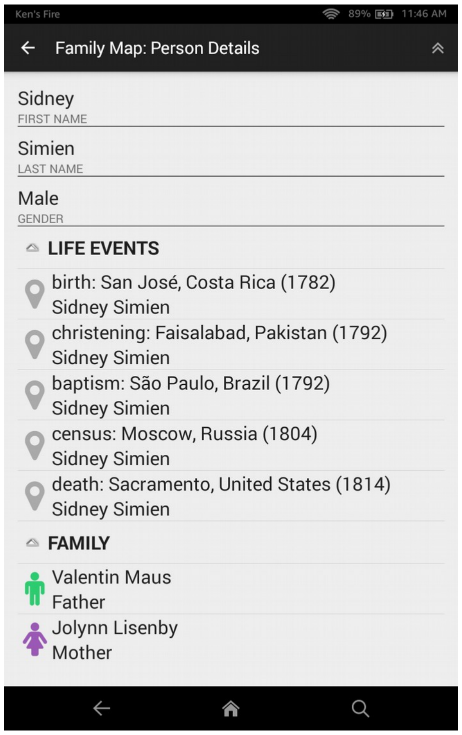
The options menu items allow the user to navigate to the Search Activity, Filter Activity, and Settings Activity, respectively. The interactive map allows the user to zoom in and out, select events from their family history, and view information about those events. Events are displayed on the map by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

The markers are color coordinated based on event type. The markers are clickable, updating the event information display when selected. 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 Filter Activity. See the sections on the Settings and Filter activities for more details.

The event information display shows information about the selected event including the person's gender, the person's name, the 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 in the user's family. The Person Activity also includes information about the person's relationships and life events.



Ken's Fire 89% 11:46 AM

Family Map: Person Details

Sidney
FIRST NAME

Simien
LAST NAME

Male
GENDER

LIFE EVENTS

birth: San José, Costa Rica (1782)
Sidney Simien

christening: Faisalabad, Pakistan (1792)
Sidney Simien

baptism: São Paulo, Brazil (1792)
Sidney Simien

census: Moscow, Russia (1804)
Sidney Simien

death: Sacramento, United States (1814)
Sidney Simien

FAMILY

Valentin Maus
Father

Jolynn Lisenby
Mother

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 an options menu that contains a "Go to Top" icon/option.

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 with years, sorted primarily by year, and secondarily by description normalized to lower-case
3. Events without years sorted by description normalized to lower-case
4. Death events, if present, are always last (whether they have a year or not)

Functionality

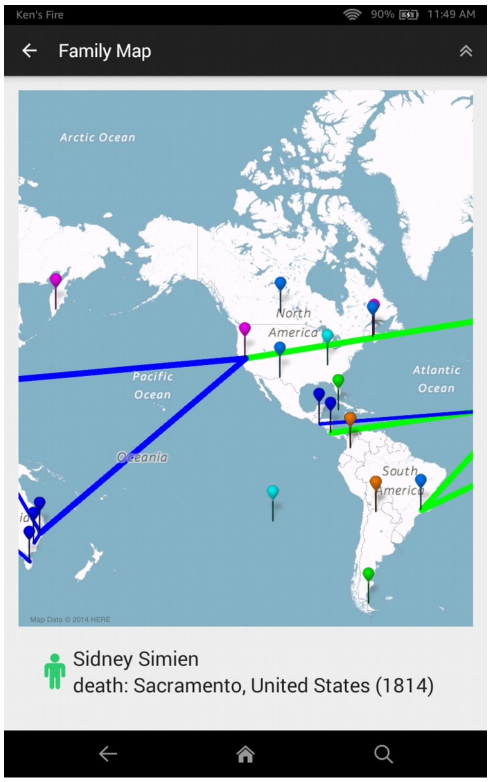
Clicking on a person in the relationship list triggers the creation of a new Person Activity with the information of the selected person. Clicking on an event triggers the creation of the Map Activity, with the map zoomed in and centered on the location of the clicked-on event (described next).

Clicking the up button returns the user to the parent activity (i.e., the activity that opened the current Person Activity).

Selecting the "Go to Top" menu option returns the user back to the top-level Main Activity. (Note, it 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.)

Map Activity

When the user clicks on an event in the Person or Search activities, a Map Activity is created to display information about the selected event. The Map Activity uses the same Map Fragment as the Main Activity to display an interactive map of events. However, when a Map 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.



Sidney Simien
death: Sacramento, United States (1814)

Layout

The Map 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 Map Activity, the Map Fragment displays an up button, and an options menu containing a "Go to Top" icon/option.

Functionality

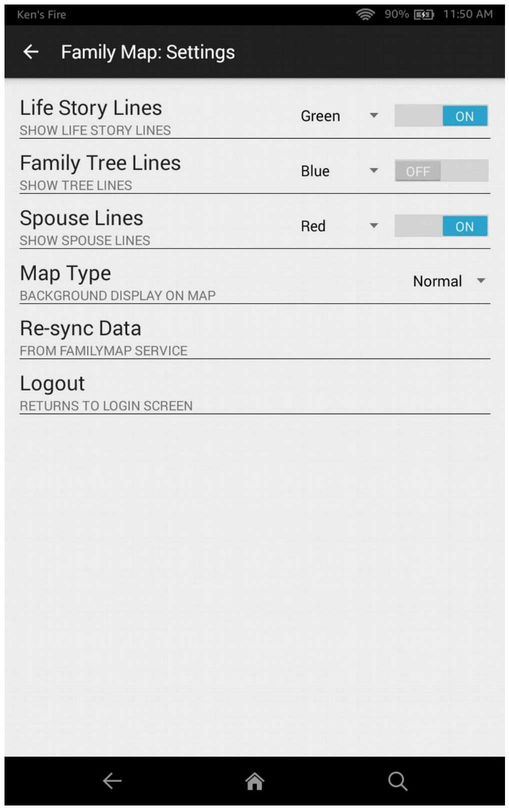
The functionality of Map Activity is the same as that of Map Fragment described previously, with the added requirement that the event being viewed is automatically selected and centered on the map.

Clicking the up button returns the user to the parent activity (i.e., the activity that opened the current Map Activity).

Selecting the “Go to Top” menu option returns the user back to the top-level Main Activity. (Note, it 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.



Life Story Lines
SHOW LIFE STORY LINES
Green
ON

Family Tree Lines
SHOW TREE LINES
Blue
OFF

Spouse Lines
SHOW SPOUSE LINES
Red
ON

Map Type
BACKGROUND DISPLAY ON MAP
Normal

Re-sync Data
FROM FAMILYMAP SERVICE

Logout
RETURNS TO LOGIN SCREEN

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.

Re-Sync

Re-sync with the FamilyMap Service. This destroys any locally cached family data and re-loads all required data from the FamilyMap Service. This is done without changing any of the application settings.

If the re-sync fails, an error message (i.e., Android “toast”) is displayed, and the Settings Activity remains open, thus allowing the user to try again.

If the re-sync succeeds, the user is returned to the Main Activity. When it is re-displayed, the Main Activity presents the Map Fragment populated with the new data and no event selected.

Map Type

Change the map type to one of the following:

- Normal (Default), Hybrid, Satellite, or Terrain

Enable/Disable Relationship Lines

Enable and disable the types of relationship lines to be drawn on the map. Each line has a color chosen by the user. By default, each line should have 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. <u>As lines are drawn recursively up the family tree, they should become progressively and noticeably thinner</u>.

Life Story Line

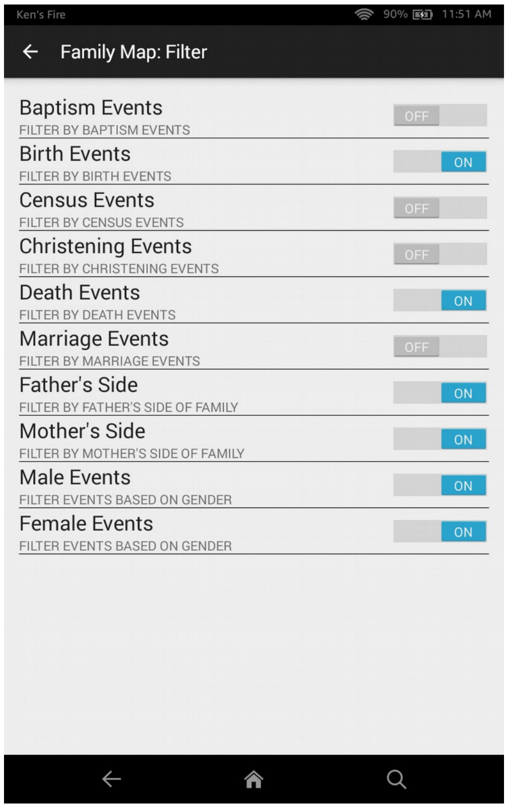
Lines are drawn connecting each event in a person's life story (i.e., the person associated with the selected event), ordered chronologically. (See the Person Activity section for information on how events are ordered chronologically.)

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.

Filter Activity

The Filter Activity allows 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 the search results (search activity) and shown in the person's details (person activity).



Ken's Fire 90% 11:51 AM

Family Map: Filter

Baptism Events FILTER BY BAPTISM EVENTS	OFF
Birth Events FILTER BY BIRTH EVENTS	ON
Census Events FILTER BY CENSUS EVENTS	OFF
Christening Events FILTER BY CHRISTENING EVENTS	OFF
Death Events FILTER BY DEATH EVENTS	ON
Marriage Events FILTER BY MARRIAGE EVENTS	OFF
Father's Side FILTER BY FATHER'S SIDE OF FAMILY	ON
Mother's Side FILTER BY MOTHER'S SIDE OF FAMILY	ON
Male Events FILTER EVENTS BASED ON GENDER	ON
Female Events FILTER EVENTS BASED ON GENDER	ON

Filter Types

The following filters are supported:

Filter by Event Type

These filters allow the user to select one or more event types that will be displayed on the map. By default, all event types are selected. <u>The event type list is built dynamically based on event descriptions that appear in the data</u>. The types of events can be random (e.g.: “Won the lottery”, or “Built a boat”). The list is case insensitive, so an event with the description “birth” is in the same type-group as an event with the description of “BIRTH”. The map updates to only show events with the specified types.

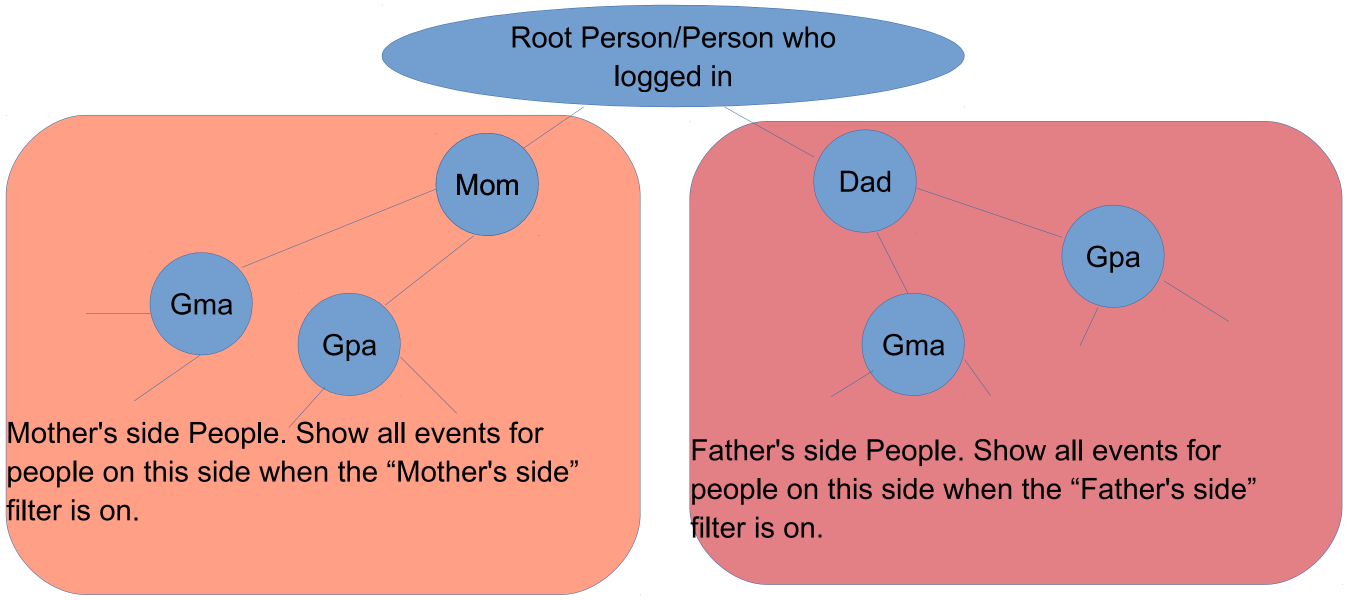
Filters should be calculated with an “and” type of statement. That is to say that all filters that apply to an event have to be turned on for the event to be shown. (e.g.: An event that is for a female, on the father's side, and is a birth event would require all those filters to be turned on for the event to be shown. If any one (or more) of those filters is turned off, that event is no longer shown).

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" you 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 gets classified as "Mother's side". The same applies for the "Father's side". When someone logs in one of the items returned in the JSON is the person's 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.

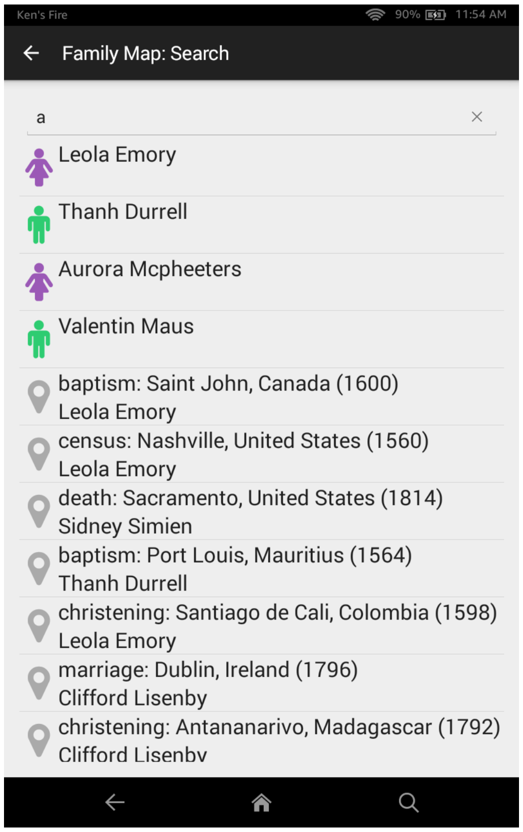

Root Person/Person who logged in
Mom
Gma
Gpa
Mother's side People. Show all events for people on this side when the "Mother's side" filter is on.
Dad
Gpa
Gma
Father's side People. Show all events for people on this side when the "Father's side" filter is on.

Up Button / Options Menu

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

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 event's countries, cities, descriptions, and years. Case is ignored. All matching people and events (that aren't filtered) are listed in the search result, as shown below.



Leola Emory

Thanh Durrell

Aurora Mcpheeters

Valentin Maus

baptism: Saint John, Canada (1600)
Leola Emory

census: Nashville, United States (1560)
Leola Emory

death: Sacramento, United States (1814)
Sidney Simien

baptism: Port Louis, Mauritius (1564)
Thanh Durrell

christening: Santiago de Cali, Colombia (1598)
Leola Emory

marriage: Dublin, Ireland (1796)
Clifford Lisenby

christening: Antananarivo, Madagascar (1792)
Clifford Lisenby

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.

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 Map 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.

Resources

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

FamilyMap Service

To help narrow the scope of this project, a web service has been provided that allows you to create user accounts, generate fake family data, login and retrieve family data from your Android application. The provided zip file includes the full source code for the FamilyMap Service. You may open this project in Android Studio, thus allowing you to run and debug the service. Alternatively, you can run the service using the provided JAR file as follows:

```
java –jar FamilyMapServer.jar <port-number> {max-generations}
```

The main method for the service is on the Server.MainServer class. This method accepts two command-line arguments, as follows:

1. Port Number (required): The port number on which the service should accept client connections (e.g., 8080).

2. Max Generations (optional): The maximum number of generations allowed by the server in **/fill/<username>?generations=N** requests (e.g., N = 4). If not specified, this value defaults to 5.

Once you've got the service running, **you can view its home page for documentation on the available service API calls. You can also make calls interactively through this page to get a better feel for the inputs and outputs of each call.** The home page can be opened in a web browser using the URL **http://host:port/** where host is the name of the machine running the service, and port is the port number the service is listening for connections on. For example, **http://localhost:8080/** All API calls are made using HTTP POST requests.

Initially, you will need to create a user with the **/user/register** call. This will register the user AND fill the database with fake data.

With the **/fill/<username>** call you can create a new set of fake data for a particular user.

Both the **/register** call and the **/fill** call should be performed through the service home page rather than from your Android app.

Once you have a user account and some generated data, you should do the following from your Android app:
Use the **/user/login** call to login the user. The HTTP response for this call includes an authorization token that you should include in all subsequent service calls in an HTTP header with the key "Authorization". It also returns the personId of the person who has just logged in.
Use the **/person** and **/event** calls to retrieve family data from the service.

Here is the JSON model:

```
person:{
        "descendant": String,
        "personID": String,
        "firstName": String,
        "lastName": String,
        "gender": String (m or f),
        "father": String (optional),
        "mother": String (optional),
        "spouse": String
}

event:{
        "eventID": String,
        "personID": String,
        "latitude": Double,
        "longitude": Double,
        "country": String,
        "city": String,
        "description": String,
        "year": String,
        "descendant": String
}
```

<u>IMPORTANT POINTS TO READ TO AVOID MUCH PAIN AND FRUSTRATION</u>

- **The server homepage will give you an opportunity to try out the different inputs and outputs of the various server API calls, thus allowing you to get familiar with the JSON formats, error messages, and registering a new user.**

- **Also remember 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).

- **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.

Map Libraries for Android

To help build the Map Fragment, you will use the Amazon Maps for Android (version 2) library. This library provides an Android UI fragment that can display maps retrieved from Amazon's map web service. The Amazon 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.

For more information and documentation on Amazon Maps for Android, see:

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

NOTE: Amazon Maps for Android is patterned after Google Maps for Android. Amazon does not allow Google Play Services to be installed on Kindle devices, so you must use Amazon Maps instead of Google Maps on the Kindle. Typically, on non-Kindle devices Google Maps is the library of choice. If you are interested in learning about Google Maps, see the following link:

https://developers.google.com/maps/documentation/android/

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 FamilyMap application:

- Male and female gender icons
- Event icon
- Search, Filter, Settings, and Go to Top options menu icons
- Other icons you wish to incorporate in your UI

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 {
   …
   compile 'com.joanzapata.android:android-iconify:1.0.9'
   …
}
```

The Java package for the Android Iconify library is **com.joanzapata.android.iconify**

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

```
import com.joanzapata.android.iconify.IconDrawable;
import com.joanzapata.android.iconify.Iconify;
…
Drawable genderIcon = new IconDrawable(getActivity(), Iconify.IconValue.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_filter (filter icon)
fa_gear (settings icon)
fa_angle_double_up (“go to top” 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 {
   …
   compile 'com.google.code.gson:gson:2.2.4'
   …
}
```

Unit Testing Requirements

Write JUnit tests to verify that your login and data synchronization functionality is working. That is, write a JUnit test that logs into the service, downloads the user's person and event data from the service, and verifies that the client's model was correctly initialized with the returned data.

Write JUnit tests to verify that your model is correctly:

- calculating family relationships (i.e., spouses, parents, children)
- filtering events according to the current filter settings
- chronologically sorting a person's individual events (birth first, death last, etc.)

Source Code Evaluation

After you pass off your project, your source code will be graded by a TA on how well you followed good programming practices. The following criteria will be used to evaluate your source code:

- (15%) Effective class, method, and variable names
- (20%) Effective decomposition of classes and methods
- (20%) Code layout is readable and consistent
- (15%) Effective organization of classes into Java packages
- (30%) High-quality unit test cases implemented with JUnit