Project 4 Task 1 – Activity Generator App

By Xiawei He

Description:

My application takes a search string of the activity type from the user and uses it to randomly generate an activity of that type from Bored API. If no search string is entered, an activity of random type will be generated for you to kill leisure time.

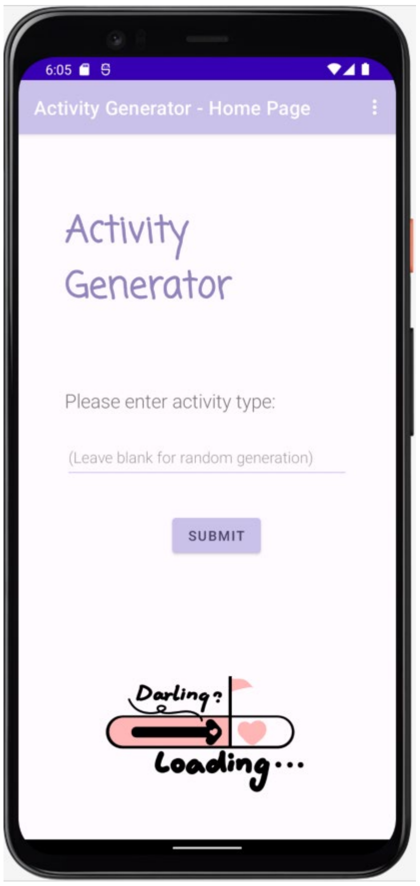
1. Implement a native Android application

The name of my native Android application project in Android Studio is:
Activity Machine

1.1. Has at least two different kinds of views in your Layout (TextView, EditText, ImageView, etc.)

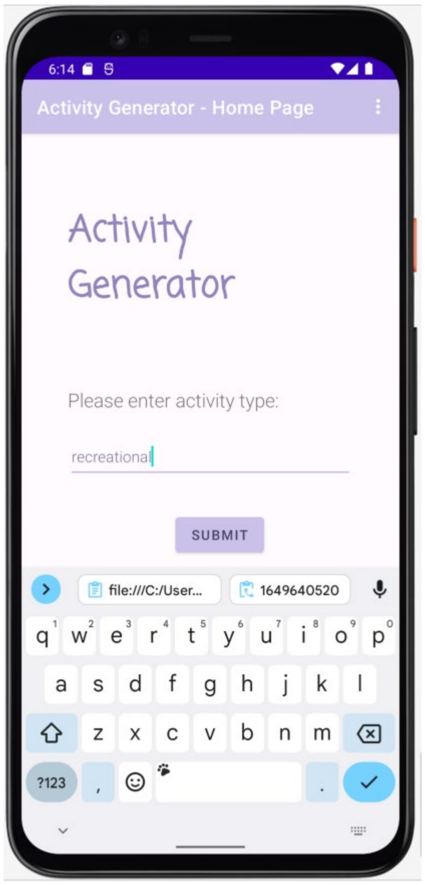
My application uses TextView, EditText, Button and ImageView. See fragment_first.xml and fragment_second.xml for details of how they are incorporated into the ConstraintLayout.

Here is a screenshot of the layout of the home page before the activity is generated.


6:05
Activity Generator - Home Page
Activity Generator
Please enter activity type:
(Leave blank for random generation)
SUBMIT
Darling?
Loading...

1.2. Requires input from the user

Here is a screenshot of the user searching for an activity of type recreational. If no user input is entered, the generator will present an activity with random type.



1.3. Makes an HTTP request (using an appropriate HTTP method) to your web service

My application does an HTTP GET request in getActivity.java. The HTTP request is:

"https://safe-stream-72232.herokuapp.com/getActivity?device=" + getDeviceName()
if no search type is entered by user.

"https://safe-stream-72232.herokuapp.com/getActivity?device=" + getDeviceName() + "&type="
+ searchType
if search type is entered by user.

The search method in getActivity.java makes the request to my web application which returns a JSON file. Then, the stringFormatter method in ActivityMachine.java parses the returned JSON to extract the activity information.

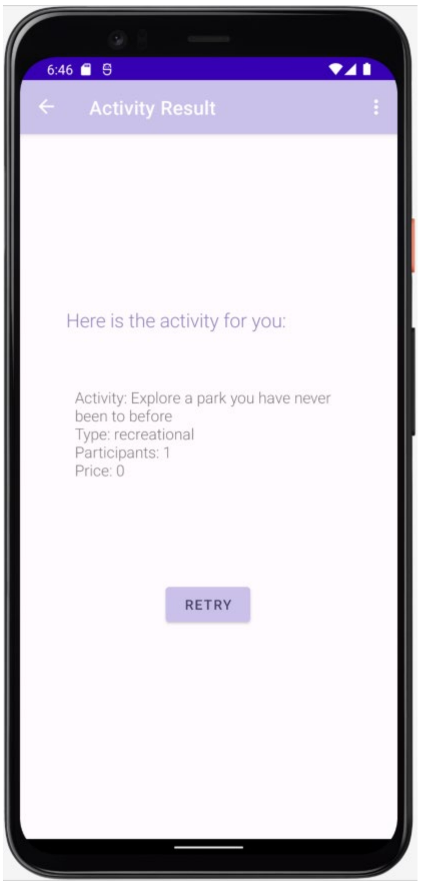
1.4. Receives and parses an XML or JSON formatted reply from the web service

An example of the JSON reply is:

{"activity":"Compliment someone","type":"social","participants":"2","price":"0","link":""}

1.5. Displays new information to the user

Here is the screen shot after the activity has been returned.

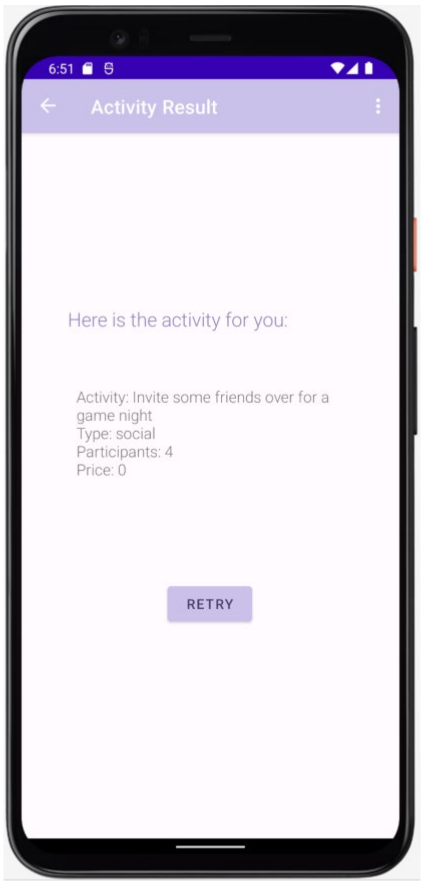


Here is the activity for you:

Activity: Explore a park you have never been to before
Type: recreational
Participants: 1
Price: 0

1.6. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)

The user can click “RETRY” and will be direct back to the home page where they can type in another search type and hit “SUBMIT”. Here is an example of having typed in “social”.



Here is the activity for you:

Activity: Invite some friends over for a game night
Type: social
Participants: 4
Price: 0

2. Implement a web application, deployed to Heroku

The URL of my web service deployed to Heroku is:
https://mighty-escarpment-00833.herokuapp.com/
The project directory name is Project4Task1.

2.1. Using an HttpServlet to implement a simple (can be a single path) API

In my web app project:

Model: the fetch method in the ApiActivity.java

View: there is no web page for task 1

Controller: ApiActivity.java

2.2. Receives an HTTP request from the native Android application

ApiActivity.java receives the HTTP GET request with the argument “type”. It passes this search string on to the model.

2.3. Executes business logic appropriate to your application

ApiActivity.java makes an HTTP request to:

http://www.boredapi.com/api/activity

or http://www.boredapi.com/api/activity?type=recreational for activity type as recreational

It then parses the Json response and extracts the parts it needs to respond to the Android application.

2.4. Replies to the Android application with an XML or JSON formatted response

ApiActivity.java formats the response to the mobile application in a simple Json format without unneeded information:

```
{"activity":"Start a collection","type":"recreational","participants":"1","price":"0","link":""}
```