CS4518 Notebook

Lab 2: User Input Data

Design Overview:

In this lab, I created a application with the following features:

- [x] Two activities that the user is able to switch between
- [x] A napping task created by Async Task that will not be interupted by other processes
- [x] Implicit Intent: the application is able to call the browse and open websites
- [x] Implicit Intent: share with other applications
- [x] Input user data into a database and query the database

Lab 2.1: the TwoActivities project

Definitions

- An `Activity` represents a single screen in your app with which your user can perform a single, focused task such as taking a photo, sending an email, or viewing a map.
- Typically, one activity in an app is specified as the "main" activity (`MainActivity.java`), which is presented to the user when the app is launched.
- Back Stack (last in, first out): Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When the user is done with the current activity and presses the Back button, that activity is popped from the stack and destroyed, and the previous activity resumes.
- An `Intent` is an asynchronous message that you can use in your activity to request an action from another activity, or from some other app component. You use an intent to start one activity from another activity, and to pass data between activities.
- Intent object can pass data to the target activity in two ways:
  - **The data field**: a URI indicating the specific data to be acted on
  - **The intent *extras*** (If the information is not a URI, or more than one piece of information want to send, put that additional information into the *extras*): key/value pairs in a `Bundle` (a collection of data, stored as key/value pairs). Put

keys and values into the intent extra `Bundle` from the sending activity, then get them back out again in the receiving activity.

> (Android fundamentals 02.1: Activities and intents, 2020)

Part 1

1. Start a new Android Studio project called TwoActivities, choose empty activity
2. In **`activity_main.xml`**, drag a button to the lower right corner. Autoconnect to create constraints.
3. Set the **ID** to `button_main`, the **layout_width** and **layout_height** to `wrap_content`, and enter `@string/button_main` for the **Text field**.
4. In **`strings.xml`** add `<string name="button_main">SEND</string>`
5. In the **Text** tab, add an onclick event `android:onClick="launchSecondActivity"`. **`Alt`** + **`Enter`** to create a method in **`MainActivity.java`**. This method is used to send log output to logcat when the `button_main` is clicked.
   1. Define `LOG_TAG`:

      ```
      private static final String LOG_TAG =
          MainActivity.class.getSimpleName();
      ```

   2. In the onClick handler, write:

      ```
      Log.d(LOG_TAG, "Button clicked!");
      ```

   3. API for sending log output: `android.util.Log`
   4. Generally, you should use the `Log.v()` (VERBOSE), `Log.d()` (DEBUG), `Log.i()` (INFO), `Log.w()` (WARN), and `Log.e()` (ERROR) methods to write logs. You can then view the logs in logcat. These different methods can be used to help filter the logs in logcat.

5. More to Refer.

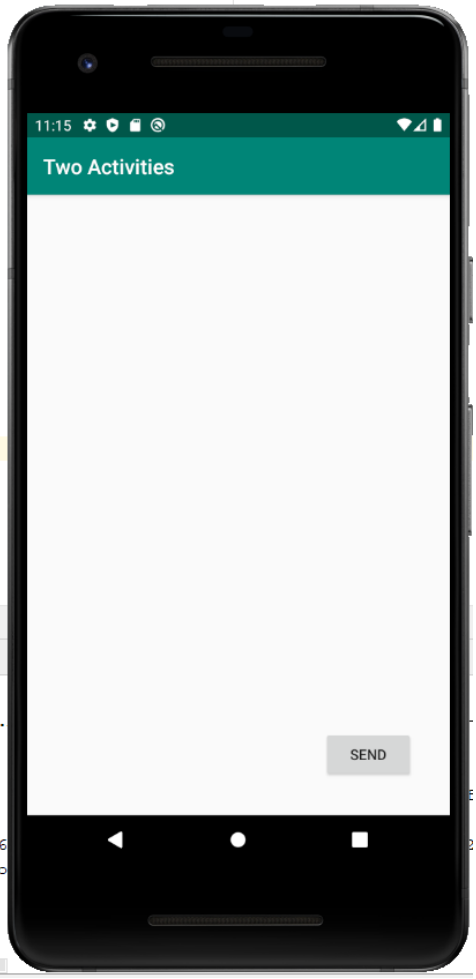
Part 1 Result

```java
package com.example.twoactivities;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.util.Log;
import android.view.View;

public class MainActivity extends AppCompatActivity {
    private static final String LOG_TAG = MainActivity.class.getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void launchSecondActivity(View view) {
        Log.d(LOG_TAG,  msg: "Button clicked!");
    }
}
```

MainActivity › launchSecondActivity()

app

```
02/01 23:14:54: Launching 'app' on Pixel 2 API 29.
$ adb shell am start -n "com.example.twoactivities/com.example.twoactivities.
Waiting for process to come online...
Connected to process 7589 on device 'emulator-5554'.
Capturing and displaying logcat messages from application. This behavior can
D/MainActivity: Button clicked!
I/OpenGLRenderer: Davey! duration=1103ms; Flags=0, IntendedVsync=17782511622
I/Choreographer: Skipped 64 frames!  The application may be doing too much wo
D/MainActivity: Button clicked!
D/MainActivity: Button clicked!
D/MainActivity: Button clicked!
```



Final result: when the left bottom button is clicked, a log message can be viewed in the console

Part 2

1. In `app`, start a new activity (**File > New > Activity > Empty Activity**), name it SecondActivity.
2. What does it do:
    1. add a new `Activity` layout `activity_second.xml`
    2. add a new Java file (`SecondActivity.java`)
    3. update the `AndroidManifest.xml` file to include the new `Activity` .

3. In `AndroidManifest.xml`, modify in the activity block of SecondActivity, add the following code to make mainActivity its parent

```
android:label = "Second Activity"
android:parentActivityName=".MainActivity">
<meta-data
    android:name="android.support.PARENT_ACTIVITY"
    android:value=
        "com.example.twoactivities.MainActivity" />
```

   1. The `parentActivityName` attribute: indicate that the main activity is the parent of the second activity.
   2. The `<meta-data>` element, provide additional arbitrary information about the activity in the form of key-value pairs. In this case the metadata attributes do the same thing as the `android:parentActivityName` attribute
   3. Metadata attributes are required for older versions of Android, because the `android:parentActivityName` attribute is only available for API levels 16 and higher.
4. In the layout `activity_second.xml`, add a textview, with id `text_header` and the following parameters:

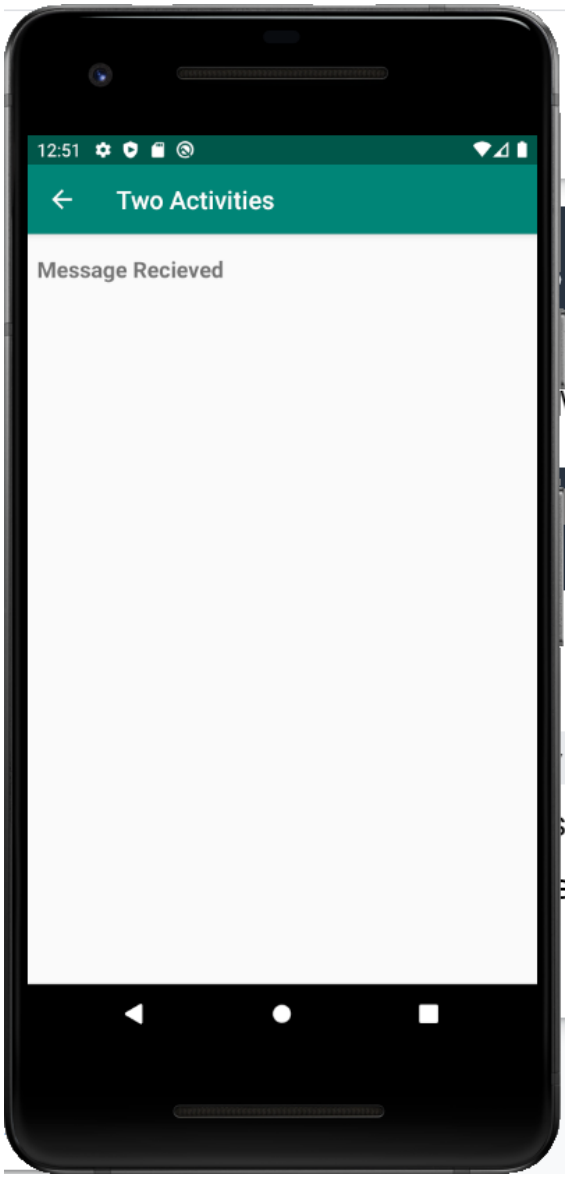
```
android:id="@+id/text_header"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginStart="8dp"
android:layout_marginLeft="8dp"
android:layout_marginTop="16dp"
android:text="@string/text_header"    //Message Received
android:textAppearance="@style/TextAppearance.AppCompat.Medium"
android:textStyle="bold"
```

5. Add an explicit `Intent` to the main `Activity` to activate the second `Activity` when the **Send** button is clicked.
   1. Create a new `intent` `launchSecondActivity()`
   2. call startActivity

```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```

Part 2 result

When click the send button, a second screen appears. Click the upper left arrow to go back to the main activity.



Second Activity

Part 3

1. Add an `EditText` to **`acitivity_main.xml`**:

```
<EditText
android:id="@+id/editText_main"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_marginStart="24dp"
android:layout_marginLeft="24dp"
android:layout_marginEnd="8dp"
android:layout_marginRight="8dp"
android:layout_marginBottom="2dp"
android:ems="10"
android:hint="@string/editText_main"
android:inputType="textLongMessage"
app:layout_constraintBottom_toBottomOf="@+id/button_main"
app:layout_constraintEnd_toStartOf="@+id/button_main"
app:layout_constraintStart_toStartOf="parent" />
```

2. Define the intend and an EditText value, assign it with `findViewById(R.id.editText_main)` in onCreate method

```
public static final String EXTRA_MESSAGE =
        "com.example.twoactivities.extra.MESSAGE";
private EditText mMessageEditText;
```

3. In the onClick event `launchSecondActivity()`, get message from `mMessageEditText`. Remember to turn it into String. Make this message an *Extra* of this Intent

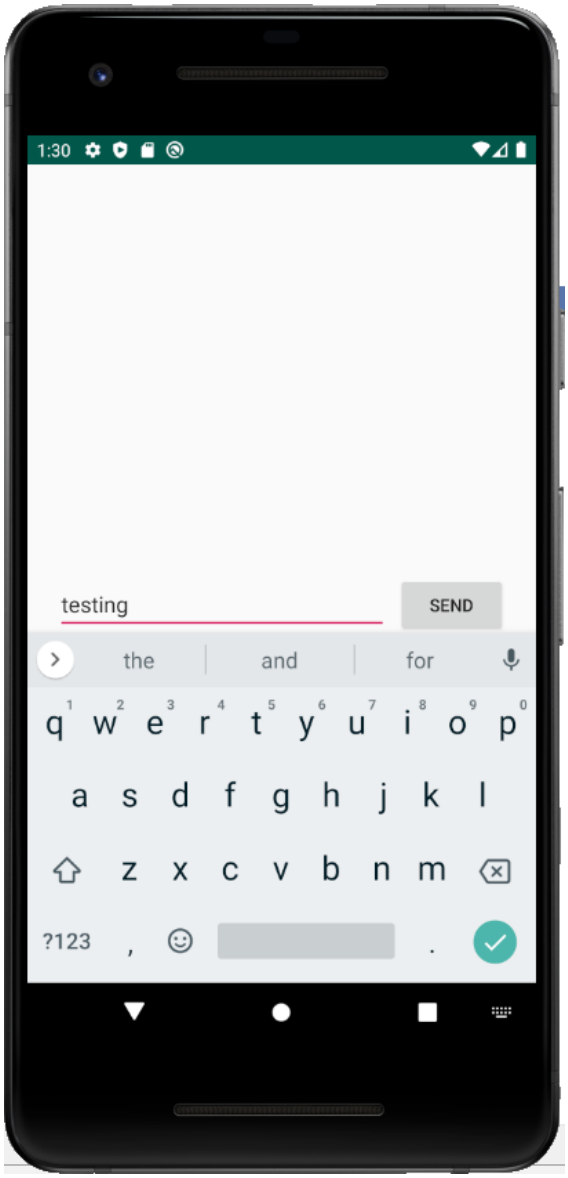
```
Intent intent = new Intent(this, SecondActivity.class);
String message = mMessageEditText.getText().toString();
intent.putExtra(EXTRA_MESSAGE, message);
```

4. In **`activity_second.xml`**, add an empty `TextView` under `text_header`, give it id `text_message`.

5. In the onCreate method of `SecondActivity.java`, use `getIntent()` to get Intent. Use `getStringExtra` on the intent to get the *Extra*. Create a `TextView`, associate with `R.id.text_message` by `findViewById`. Set the text of textView by message.

```
Intent intent = getIntent();
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
TextView textView = findViewById(R.id.text_message);
textView.setText(message);
```

Part 3 result



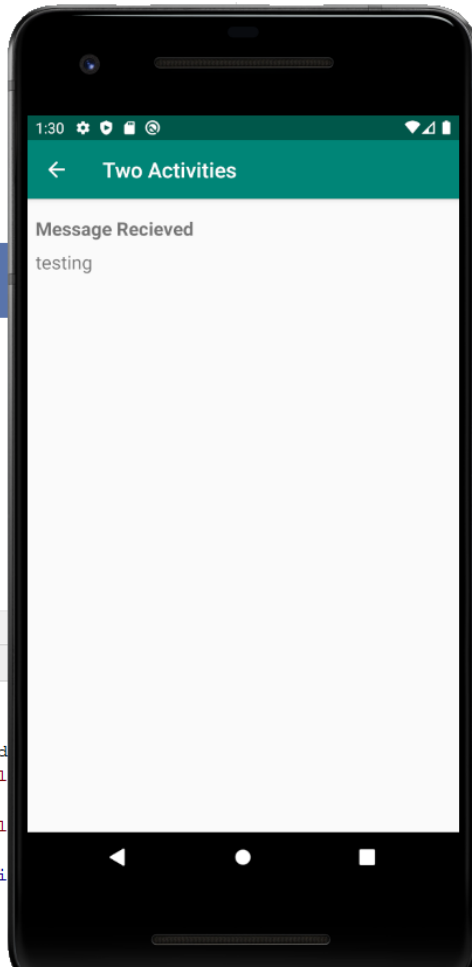
Activity 1

```java
package com.example.twoactivities;

import ...

public class SecondActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        Intent intent = getIntent();
        String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
        TextView textView = findViewById(R.id.text_message);
        textView.setText(message);
    }
}
```

SecondActivity › onCreate()

app

```
D/EGL_emulation: eglMakeCurrent: 0xe401a420: ver 3 0 (tinfo 0xe400f830)
D/eglCodecCommon: setVertexArrayObject: set vao to 0 (0) 0 0
I/AssistStructure: Flattened final assist data: 1476 bytes, containing 1 wind
E/SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero l
I/chatty: uid=10135(com.example.twoactivities) identical 2 lines
E/SpannableStringBuilder: SPAN_EXCLUSIVE_EXCLUSIVE spans cannot have a zero l
D/MainActivity: Button clicked!
W/ActivityThread: handleWindowVisibility: no activity for token android.os.Bi
D/EGL_emulation: eglMakeCurrent: 0xe401a420: ver 3 0 (tinfo 0xe400f830)
D/EGL_emulation: eglMakeCurrent: 0xe401a420: ver 3 0 (tinfo 0xe400f830)
D/EGL_emulation: eglMakeCurrent: 0xe401a420: ver 3 0 (tinfo 0xe400f830)
```



Activity 2 (message received)

Part 4

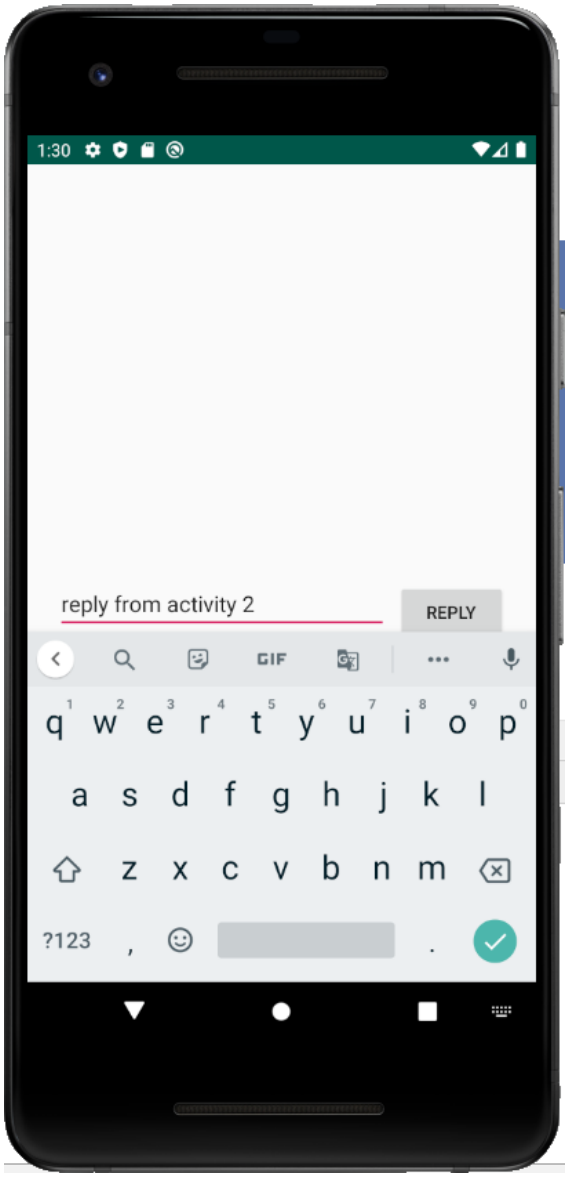
1. Copy the **EditText** `editText_main` and **Button** `button_main` from **`activity_main.xml`** to **`activity_second.xml`** and make changes to create a reply window.
2. Create a onClick event for `button_second` in **`SecondActivity.java`** called `returnReply()`

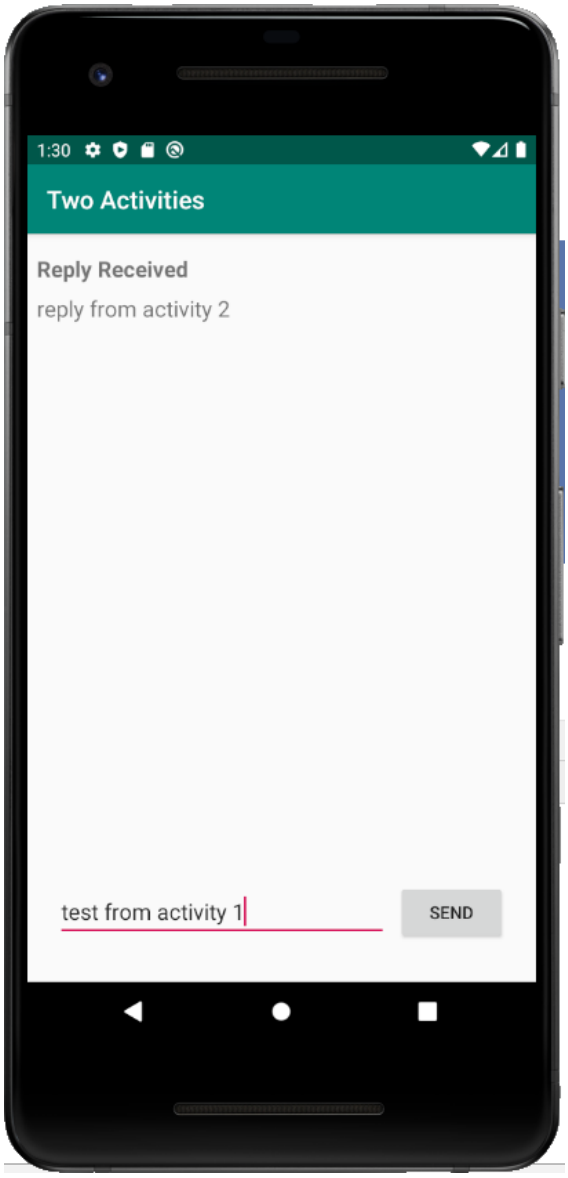
```java
String reply = mReply.getText().toString();
Intent replyIntent = new Intent();
replyIntent.putExtra(EXTRA_REPLY, reply);
setResult(RESULT_OK, replyIntent);
finish();
```

3. In `MainActivity` and `activity_main.xml`, create a textbox to display the reply. Copy past from the upper left cornor from `activity_second.xml`. Change to make them fit. Make these 2 elements initially invisible.
4. In `MainActivity` change `startActivity( )` to be `startActivityForResult()`, and include the `TEXT_REQUEST` key (define it as 1) as the second argument.
5. Override `onActivityResult(int requestCode, int resultCode, Intent data)`

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent d
    super.onActivityResult(requestCode,resultCode,data);
    if (requestCode == TEXT_REQUEST) {
        if (resultCode == RESULT_OK) {
            String reply = data.getStringExtra(SecondActivity.EXTRA_RE
            mReplyHeadTextView.setVisibility(View.VISIBLE);
            mReplyTextView.setText(reply);
            mReplyTextView.setVisibility(View.VISIBLE);
        }
    }
}
```

Part 4 result


1:30
reply from activity 2
REPLY
GIF
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123



Lab 2.2: AsyncTask

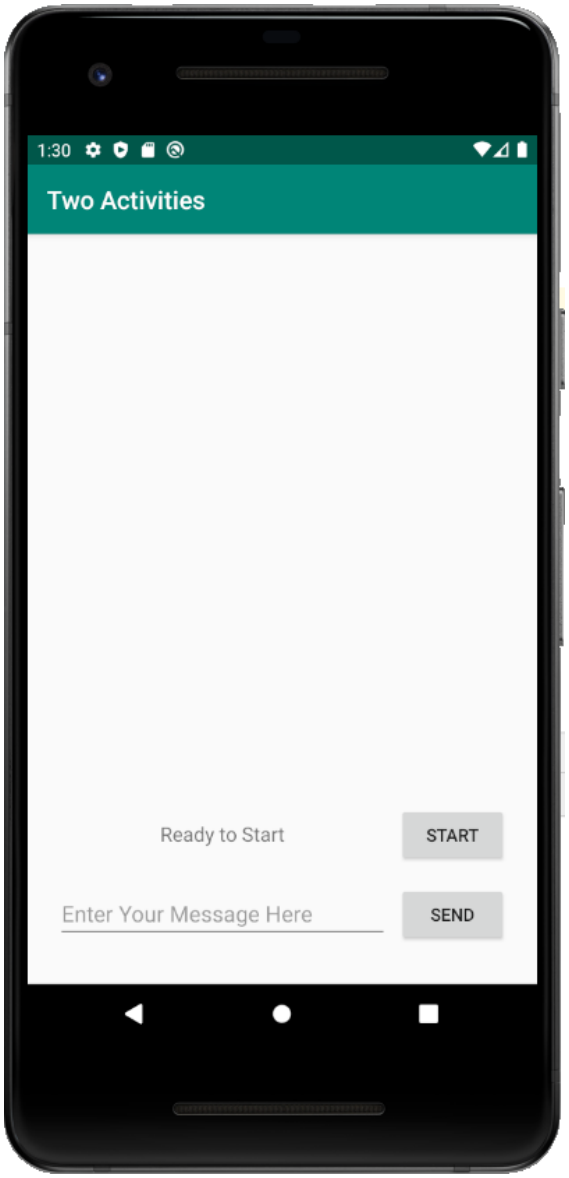
Definitions

- To keep the user experience (UX) running smoothly, the Android framework provides a helper class called `AsyncTask`, which processes work off of the UI thread. Using `AsyncTask` to move intensive processing onto a separate thread means that the UI thread can stay responsive.

> (Android fundamentals 07.1: AsyncTask, 2020)

Part 1

Create a `TextView` and a `button` in `activity_main` for the task. Create a onClick event called `startTask()` for the button.



Part 2

1. Create a new Java file name `SimpleAsyncTask`, extend `AsyncTask <Void, Void, String>`
2. Create a field and a constructor

```
private WeakReference<TextView> mTextView;
SimpleAsyncTask(TextView tv) {
        mTextView = new WeakReference<>(tv);
}
```

3. Override `doInBackground` to sleep for random seconds.
4. Create a method to execute

```
protected void onPostExecute(String result) {
    mTextView.get().setText(result);
}
```

Part 3

Implement in `MainActivity.java`

1. Define and initialize in `onCreate` : `private TextView mTextViewMainStart`
2. In startTask, execute a `SimpleAsyncTask` with `mTextViewMainStart`
3. Save Instance state
    1. Store current state in `onCreate()`

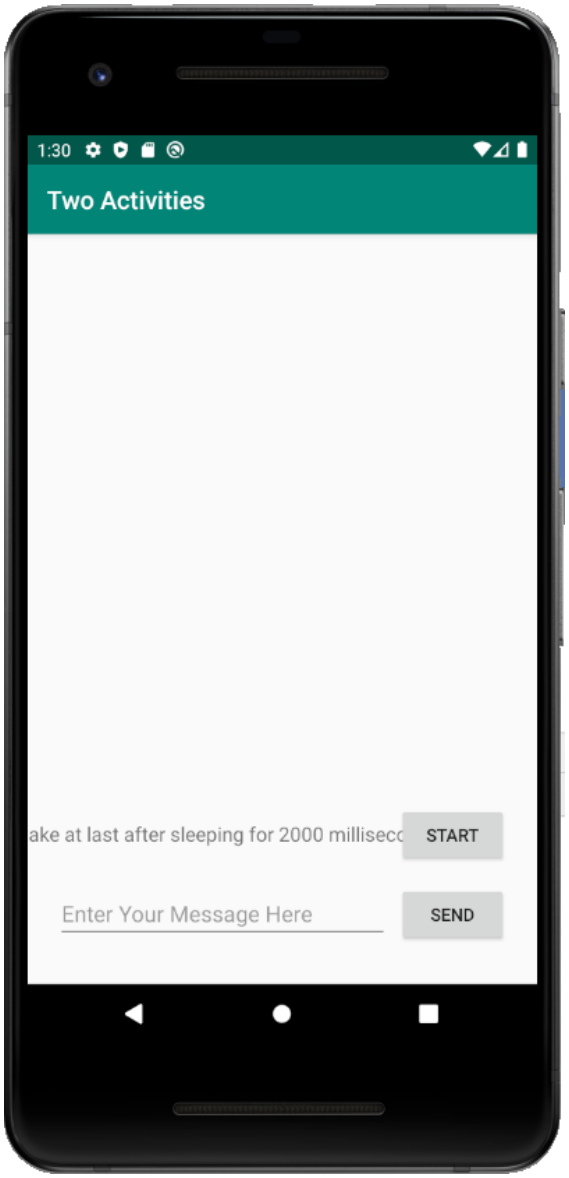
    ```
    if(savedInstanceState!=null){
        mTextViewMainStart.setText(savedInstanceState.getString(TEXT_S
    }
    ```

    2. Override onSaveInstanceState

    ```
    protected void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
    ```

```
    // Save the state of the TextView
    outState.putString(TEXT_STATE,
            mTextViewMainStart.getText().toString());
}
```

Part 3 result

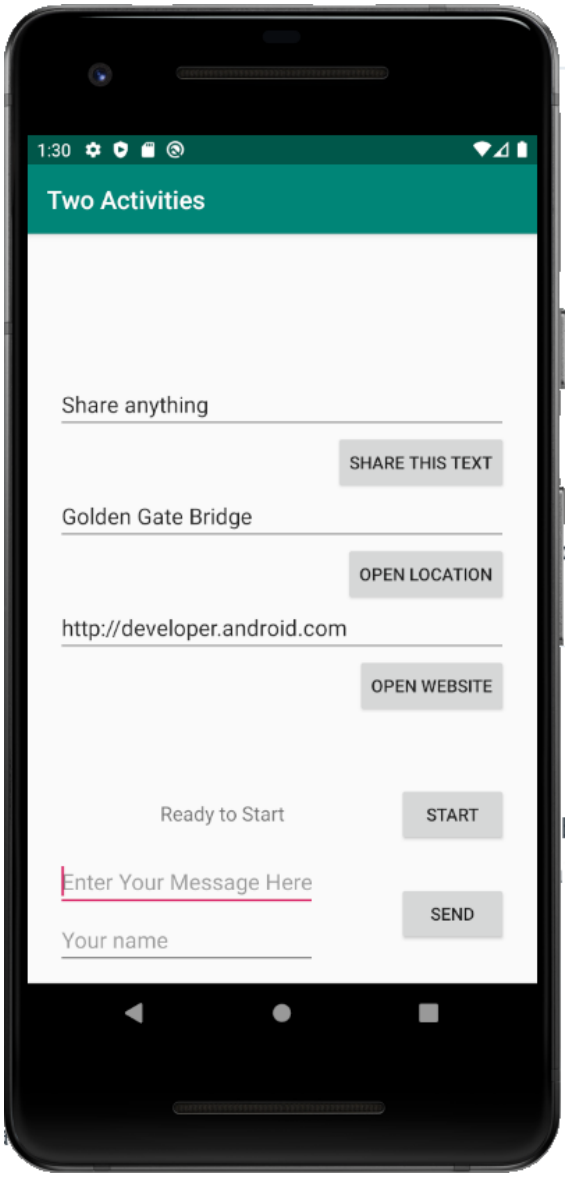


The app randomly sleeped for several seconds. The UI and the Process is not interupted if rotate.

Lab 2.3: Implicit intents

Part 1

Create a layout

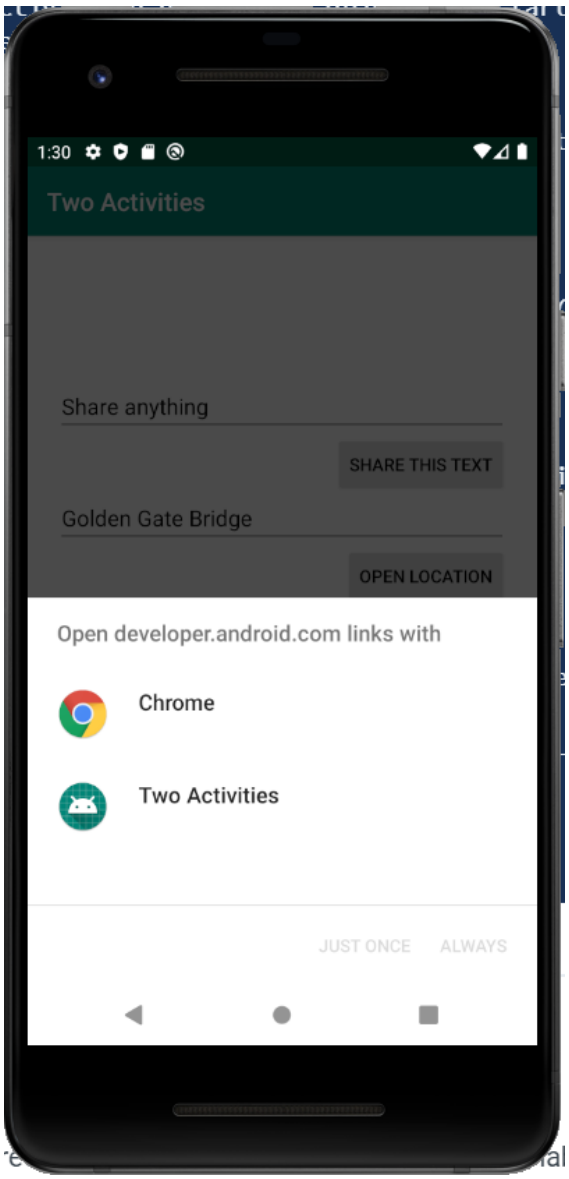


Part 2

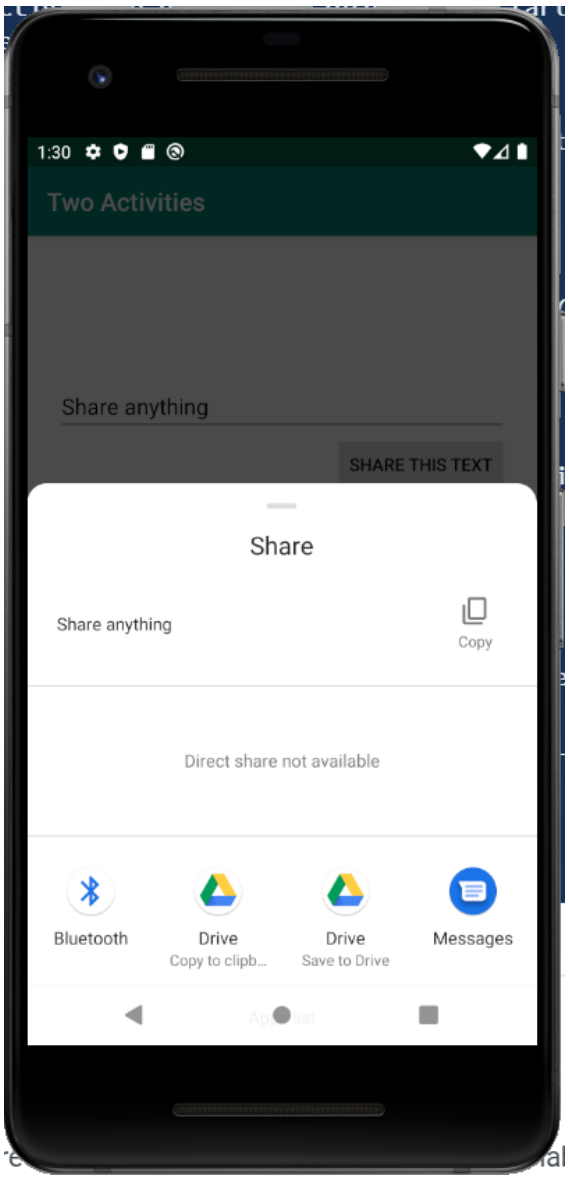
Create onClick buttons to open the browse, google map, and the share pop

```
public void openWebsite(View view) {
    String url = mWebsiteEditText.getText().toString();
```

```
    // Parse the URI and create the intent.
    Uri webpage = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
    // Find an activity to hand the intent and start that activity.
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    } else {
        Log.d("ImplicitIntents", "Can't handle this intent!");
    }
}

public void openLocation(View view) {
    // Get the string indicating a location. Input is not validated; it is
    // passed to the location handler intact.
    String loc = mLocationEditText.getText().toString();
    // Parse the location and create the intent.
    Uri addressUri = Uri.parse("geo:0,0?q=" + loc);
    Intent intent = new Intent(Intent.ACTION_VIEW, addressUri);
    // Find an activity to handle the intent, and start that activity.
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    } else {
        Log.d("ImplicitIntents", "Can't handle this intent!");
    }
}

public void shareText(View view) {
    String txt = mShareTextEditText.getText().toString();
    String mimeType = "text/plain";
    ShareCompat.IntentBuilder
                .from(this)
                .setType(mimeType)
                .setChooserTitle(R.string.share_text_with)
                .setText(txt)
                .startChooser();
    Intent intent = getIntent();
    Uri uri = intent.getData();
    if (uri != null) {
        String uri_string = getString(R.string.uri_label)
                + uri.toString();
        TextView textView = findViewById(R.id.text_uri_message);
        textView.setText(uri_string);
    }
}
```



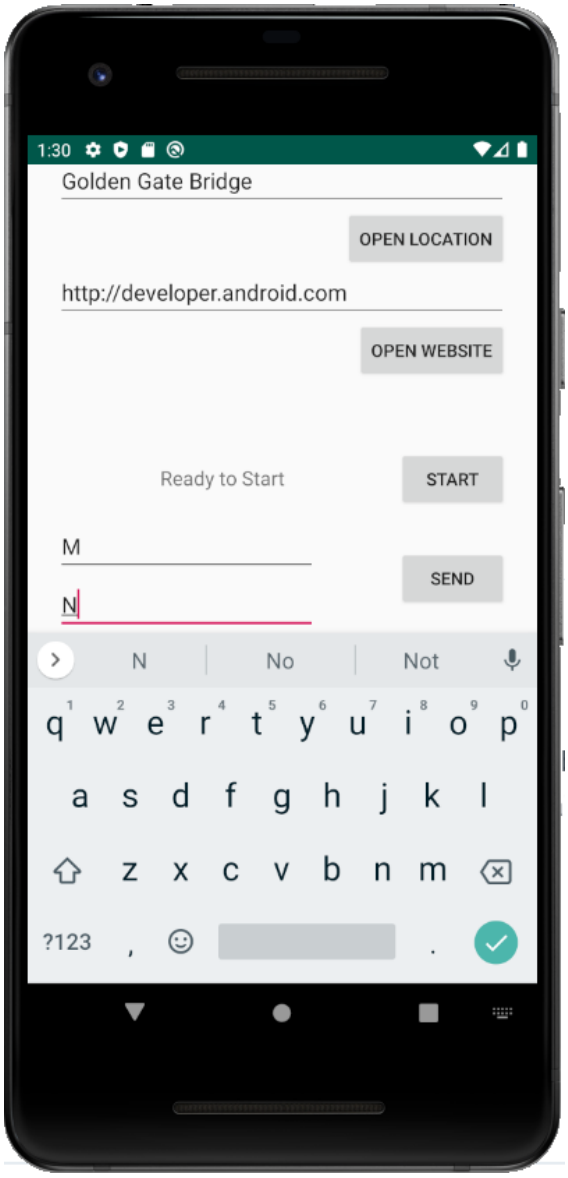
Open in Browse



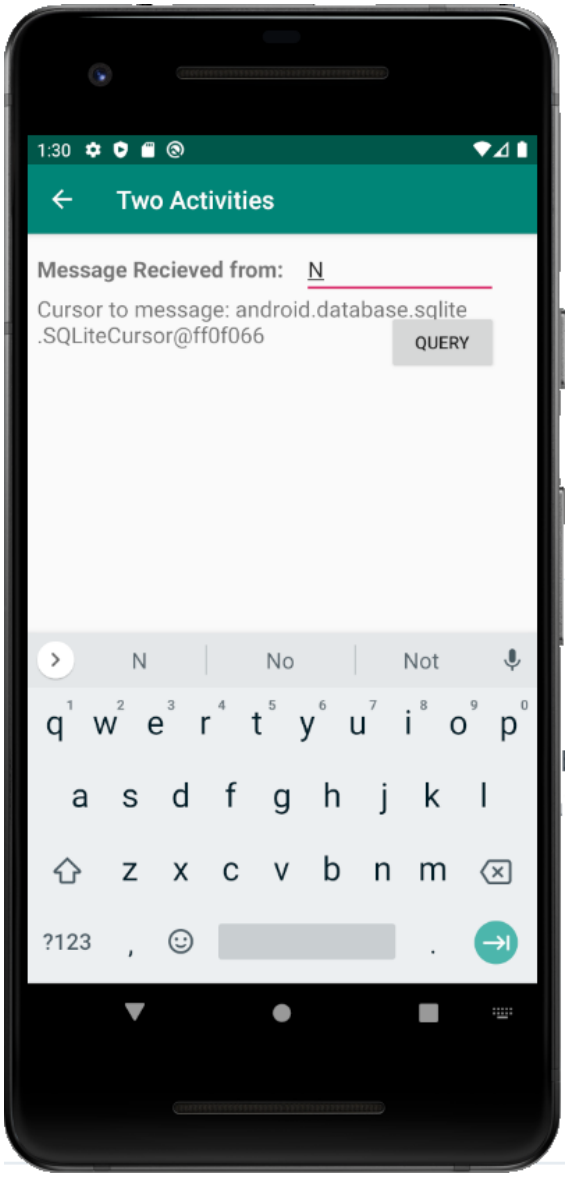
Share Text

Final Project (self designed)

Instead of doing a login screen, I combined this task with the previous labs I did. I replaced the explicit intent. Instead, I store the user input information to a database in the mainActivity, and query to get the message from the secondActivity.



What I still need to improve here is that I did not get the actual message. I only showed the address of this message stored in the database.



References

Android fundamentals 02.3: Implicit intents. (2020). Google.Com. https://codelabs.developers.google.com/codelabs/android-training-activity-with-implicit-intent/index.html?index=..%2F..%2Fandroid-training#0

Android fundamentals 02.1: Activities and intents. (2020). Google.Com. https://codelabs.developers.google.com/codelabs/android-training-create-an-activity/index.html?index=..%2F..android-training#11

Android fundamentals 07.1: AsyncTask. (2020). Google.Com. https://codelabs.developers.google.com/codelabs/android-training-create-

asynctask/index.html?index=..%2F..android-training#0

RajaVamsi11. (2019, March 31). RajaVamsi11/android-login-and-registration. GitHub. https://github.com/RajaVamsi11/android-login-and-registration