CST2335 – Graphical Interface Programming

Lab 3

Introduction:

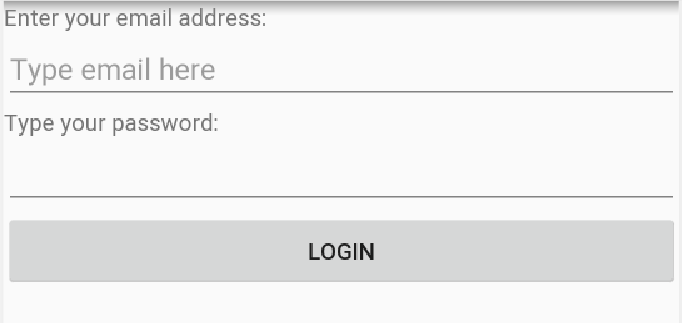
The goal of this lab is become familiar with the Activity lifecycle. Various callback functions are called throughout the stages of starting and stopping an Activity. Also, you will learn how to launch an Activity from another activity.

References:

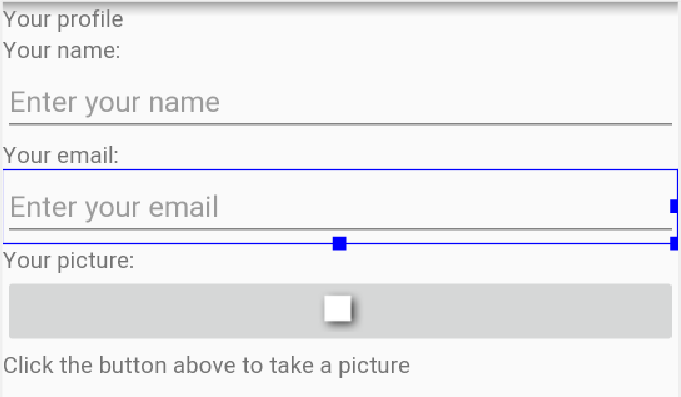
1. <https://developer.android.com/training/basics/data-storage/shared-preferences.html>
2. <https://developer.android.com/guide/components/activities.html>
3. <https://developer.android.com/reference/android/util/Log.html>
4. <https://developer.android.com/training/basics/intents/result.html>

Steps:

1. Create a branch of your software from Lab 2. Select the VCS menu in Android Studio and click “Git” -> “branches”. From the dialog box, select “New Branch” and call it “Lab 3”. You should now have the “master” branch, which is your work from Lab 1, and then another branch called Lab 2.
2. Write a layout file that looks like this:



1. The first and third elements are <TextView> objects. The second and fourth elements are <EditText> widgets. The first will be for entering someone’s email address. You should use the parameter android:inputType=”textEmailAddress”. This will present a keyboard for typing in email addresses. The second EditText should use android:inputType=”textPassword”. This will make the EditText replace the text with “\*” to hide what was typed.
2. Set your MainActivity.java to load this layout file that you created. In the onPause() function, use SharedPreferences to save the user’s email address. This way, when a user presses the Login button, it will save what was typed and reload it the next time the user goes to the page.
3. Somewhere in the onCreate( ) function, load the SharedPreferences and load the user’s email address under the reservation name you used. If nothing is reserved, use the empty string “” as the default value. That way, the EditText hint will show. Debug your program to make sure that you are saving the user’s input correctly and reloading it next time.
4. In AndroidStudio, select the “File” menu, then “New” -> “Activity” -> “Empty Activity”. Give the Activity the name: “ProfileActivity” and make sure “Generate Layout File” is checked. AndroidStudio will then create the Java file, and XML layout file, as well as put a new <Activity> tag in the AndroidManifest for your new Activity. Go to your newly created layout file, and make it look like this:



The text “Your profile”, “Your name”, “Your email”, “Your picture”, and “Click the button above to take a picture” are all <TextView> widgets. The “Enter your name”, and “Enter your email” are <EditText> widgets.

1. Between “Your picture” and “Click the button..” text is an ImageButton. You should set the parameter android:src="@android:drawable/picture\_frame". You should add a click listener to the button so that it creates an Intent to use Android’s built in camera Activity to take a picture:

static final int REQUEST\_IMAGE\_CAPTURE = 1;

private void dispatchTakePictureIntent() {

Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

if (takePictureIntent.resolveActivity(getPackageManager()) != null) {

startActivityForResult(takePictureIntent, REQUEST\_IMAGE\_CAPTURE);

}

}

This will start Android’s Activity that is responsible for the MediaStore.ACTION\_IMAGE\_CAPTURE intent. Since you are using startActivityForResult( ), the activity will call your onActivityResult(int request, int result, Intent data ) when finished. If you clicked the checkbox button on the camera activity, then result will be Activity.RESULT\_OK, and the data parameter will be an intent that has the picture saved under the name “data”:

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

if (requestCode == REQUEST\_IMAGE\_CAPTURE && resultCode == RESULT\_OK) {

Bundle extras = data.getExtras();

Bitmap imageBitmap = (Bitmap) extras.get("data");

mImageButton.setImageBitmap(imageBitmap);

}

}

Copy and paste this code into your ProfileActivity.java. You will have to create a class variable called mImageButton. In your onCreate, initialize this to be the ImageButton you created in your XML file, using (ImageButton)findViewById(). Note here that the resultCode == RESULT\_OK is checking that you pressed the “Accept” button, or checkmark. If you hit the back button at the bottom of the phone, then the resultCode would be RESULT\_CANCELED and the if {} block would not be run.

1. Since you are using the phone’s camera, you will first have to ask for permission in the manifest. Add this to the AndroidManifest, but put it before of the <Application > tag:

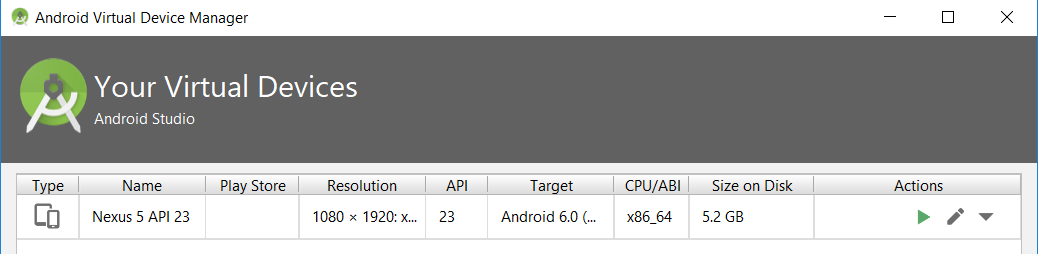
<uses-feature android:name="android.hardware.camera" android:required="true" />

1. In the ProfileActivity.java file, add logging information calls to all of the functions onCreate, onStart, onResume, onPause, onStop, onDestroy, onActivityResult:

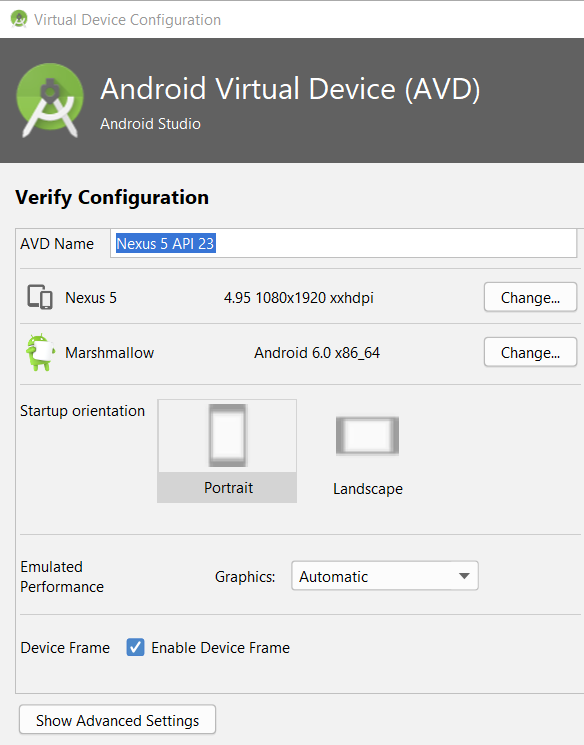
Log.e(ACTIVITY\_NAME, “In function:” + /\* replace with function name \*/);

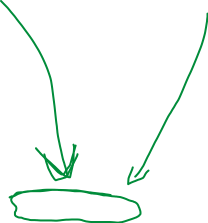
//You will have to create a static String variable called ACTIVITY\_NAME, and set it equal to the Activity’s name. In this case:

public static final String ACTIVITY\_NAME = "PROFILE\_ACTIVITY";

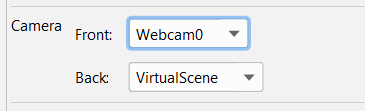
1. You can use your laptop’s camera as a replacement for your emulator’s camera. Click on the AVD manager button in AndroidStudio. From the virtual devices list, click on the pencil icon to edit your virtual device: 

Then click on “Show Advanced Settings”:





Find the Camera setting. Set ***both the Front and Back*** camera to “Webcam0”:



Now when you use the Image Capture activity on your emulator, you should be able to take a real picture. This depends on your computer and the webcam device drivers. If you run into problems and you can’t use it, just set the camera to “VirtualScene” and it will take a fake photo of a virtual room. You could also use an Android phone or tablet to take your pictures.

1. Lastly, add an onClickListener() function to the Login button in MainActivity.java. Whenever you click on the button, you should make an Intent to go to the ProfileActivity. In this Intent, pass along email address that was typed in the email field to the ProfileActivity:

Intent goToProfile = new Intent(MainActivity.this, ProfileActivity.class);

goToProfile.putExtra(“EMAIL”, … what is written in the edit text);

Then in the onCreate of ProfileActivity, get the string from the intent:

Intent fromMain = getIntent();

fromMain.getStringExtra(“EMAIL”);

and put that string in the email field of ProfileActivity:

emailEditText.setText( … the string from Intent extras …);

1. Commit your work with Git, using the commit message “Finished lab 3”, and push the code to the github repository.
2. Demonstrate your work to the lab professor showing the following parts of your lab work:
   1. Set breakpoints on all of the Log.e() messages in ProfileActivity.java and use Debug to show your application. When showing your application, show to the professor that you can step over a line of code, find the value of your variables in the Debug window, and resume the program (+3)
   2. The email address for login is saved in sharedPreferences (whatever you typed last time shows up the next time you start your application).
   3. Use the AndroidStudio terminal window to log into the phone using “adb shell”. From the command line, type “run-as package-name”. Then navigate to “/data/data/package-name/shared\_prefs”, and display the shared preferences using “more filename.xml”
   4. Clicking login starts the ProfileActivity, and you see the ProfileActivity’s onCreate, onStart, and onResume debug message in the LogCat window.
   5. Clicking on the imageButton starts the phone’s camera application. Clicking the back button should go back to the ProfileActivity, and update the image and show the onResume debug message in the LogCat window.
   6. Update your strings.xml file and show that changing Android’s language to your second language displays the translated messages.

Note: This lab uses the following patterns:

* 1. Memento pattern for saving Shared Preferences.
  2. Façade pattern for saving Shared Preferences (one simplified interface to writing to file, database, network connection, etc).
  3. Command Pattern – An Intent object represents the command. The Intent is then resolved to an Activity which can return a value upon completion of the Activity.

Marks: (10 total)

1. Student can set breakpoints, step over a line of code, read the value of variables in the Debug window, and resume the program +3
2. Email address gets stored in sharedPreferences +1
3. Student can navigate to the shared preferences file on the phone and display contents using ***cat*** or ***more*** command. +1
4. LoginActivity transitions to ProfileActivity +1
5. ProfileActivity transitions to photo activity, which updates the button image +1
6. The new buttons all support multiple languages +1
7. If the code was not uploaded to github (-8)