Developing Android Applications

SEG2105 - Introduction to Software Engineering – Fall 2019

Lab Exercise: Profile Manager

- **Objective:**
  1. Build a Sports-centric profile manager

- **Requirements:**
  1. Application should have multiple activities
     1. Each activity should have a specific purpose
  2. You should be able to set/change the Profile Name
  3. You should be able to set/change the Profile Logo/Avatar
     1. Separate Activity
  4. You should be able to set the An Address
     1. Separate Activity
  5. You should be able to open a map application in the correct address set in your application

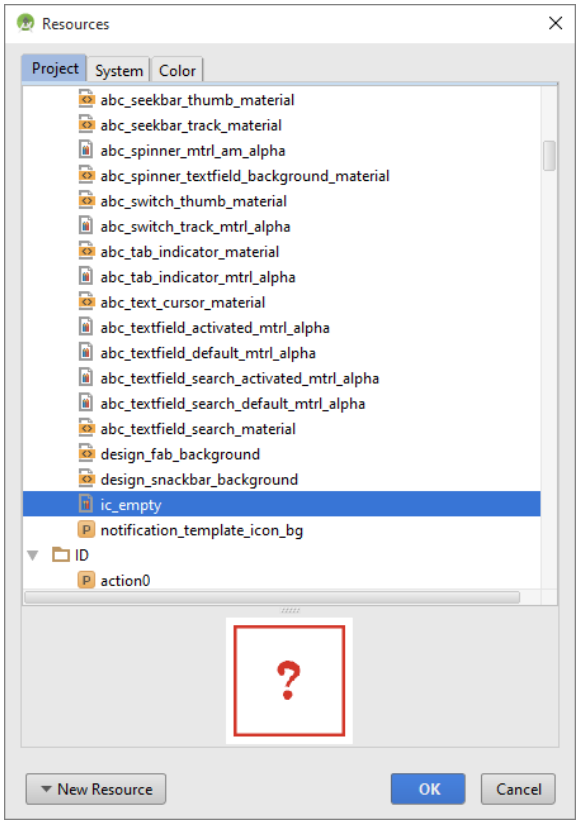
Main Activity: Profile View

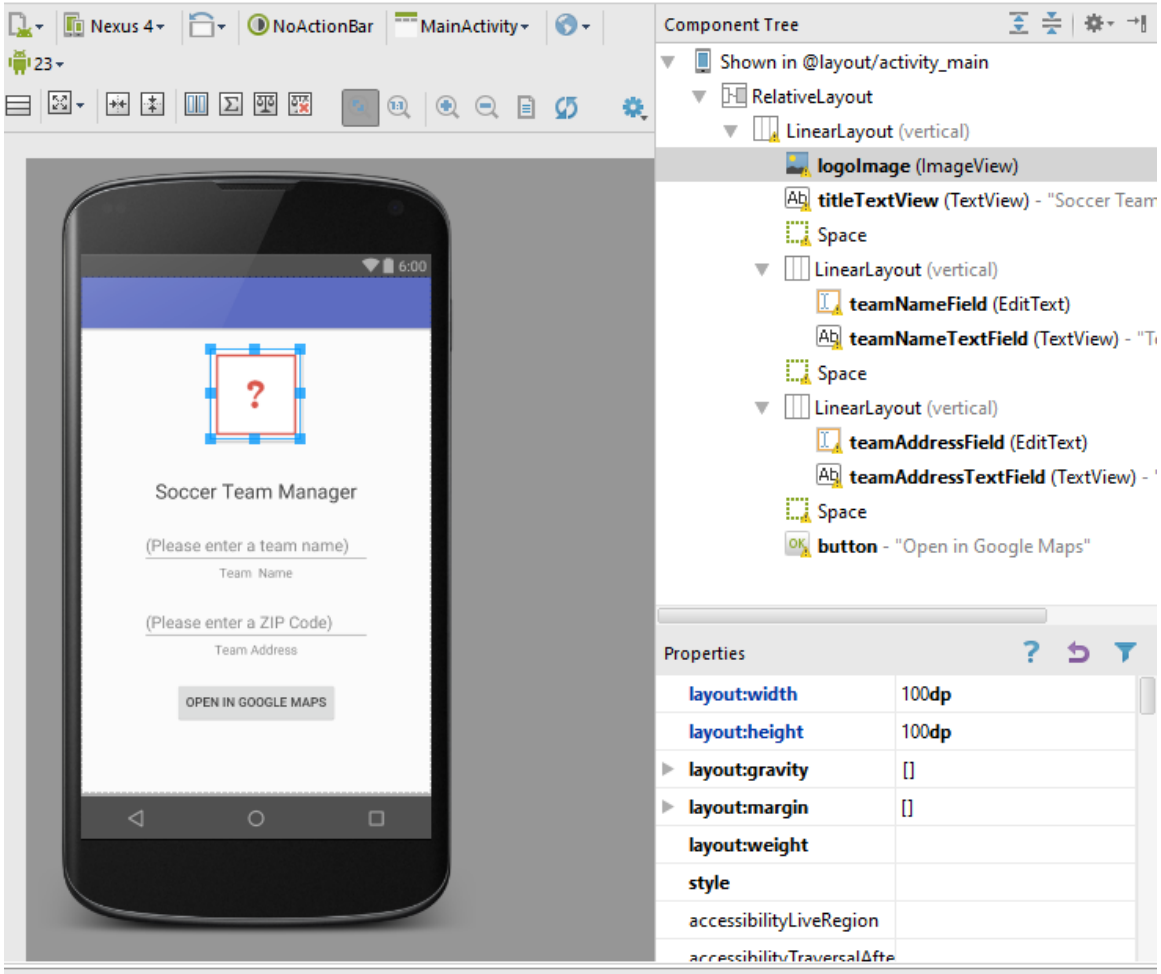
- Create a New Project with an EMPTY ACTIVITY (same as last lab).
- Add a EditText field for the Team/Person name and an accompanying TextView that reads "Name:"
  - In the TeamName EdidText field, set the *hint* property to: "(Please enter a name)"
- Repeat the process for the Location (Label, and editable text)
- Add a **Image View** to represent the Logo / Avatar.
  - The image will initially appear invisible, this is because no image is assigned, nor are size properties.
  - Sthe imageView's width and height parameter to 100dp and you will see the image box.
- Add a button in the bottom of the screen to open a Map
  - The OnClick Button will be presented later (**OnOpenInGoogleMaps)**

Changing Graphics

- To add an art asset (image, sound, etc) to your project, all you have to do is copy it to the "drawable" project folder.
  - app >> res >> drawable.
  - You can right-click on the drawable folder and select "show in explorer" to find it more easily.

- Switch the standard image for an empty alternative
  - Required art assets are available at (pick 1 set):
    - Avatar:
      - http://www.modesto.io/static/UOT/avatars.zip
    - Logo:
      - http://www.modesto.io/static/UOT/logos.zip




Nexus 4
NoActionBar
MainActivity
23
6:00
Soccer Team Manager
(Please enter a team name)
Team Name
(Please enter a ZIP Code)
Team Address
OPEN IN GOOGLE MAPS
Component Tree
Shown in @layout/activity_main
RelativeLayout
LinearLayout (vertical)
logoImage (ImageView)
titleTextView (TextView) - "Soccer Team
Space
LinearLayout (vertical)
teamNameField (EditText)
teamNameTextField (TextView)
Space
LinearLayout (vertical)
teamAddressField (EditText)
teamAddressTextField (TextView)
Space
button - "Open in Google Maps"
Properties
layout:width
100dp
layout:height
100dp
layout:gravity
[]
layout:margin
[]
layout:weight
style
accessibilityLiveRegion

Explicit Intent to External Activity

- Add this Method to your Main Activity Java File
  - Set the OnClick Function of your "Open Map" button
  - It will open your location on Google Maps
  - This function Explicitly requests the google maps package in the intent.

```
public void OnOpenInGoogleMaps (View view) {

    EditText teamAddres = (EditText) findViewById(R.id.teamAddressField);

    // Create a Uri from an intent string. Use the result to create an Intent.
    Uri gmmIntentUri = Uri.parse("http://maps.google.co.in/maps?q="+teamAddres.getText());

    // Create an Intent from gmmIntentUri. Set the action to ACTION_VIEW
    Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);

    // Make the Intent explicit by setting the Google Maps package
    mapIntent.setPackage("com.google.android.apps.maps");

    // Attempt to start an activity that can handle the Intent
    startActivity(mapIntent);
}
```

Note: For this to work, you need Google APIs in your Virtual Emulator, most current Android Images have it included.

Explicit Intent to Internal Activity

- Add the function below to your Main Activity Java File
  - Set the ***OnClick*** Function of your ***ImageView***
  - It will open another activity where we will choose the new avatar for the team

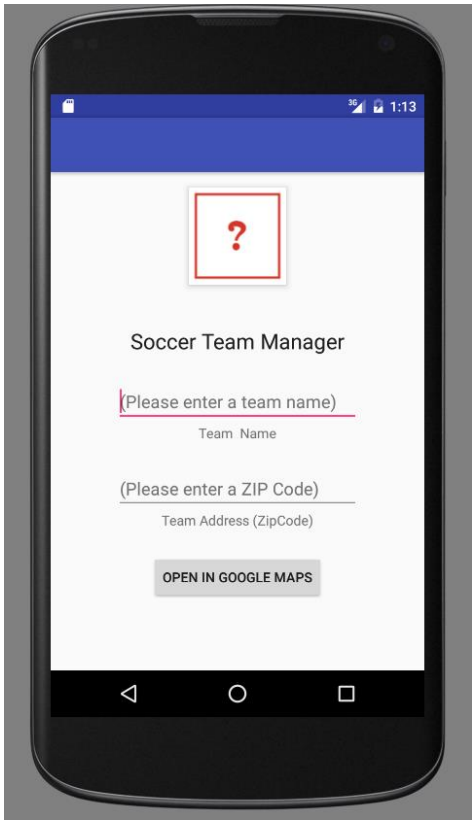
```
public void OnSetAvatarButton(View view) {
    //Application Context and Activity
    Intent intent = new Intent(getApplicationContext(), ProfileActivity.class);
    startActivityForResult (intent,0);
}
```

*Note: **ProfileActivity** is the name we gave to our second activity in the example. If you gave your second activity a different name, you will need to provide the corresponding name to the Intent.*

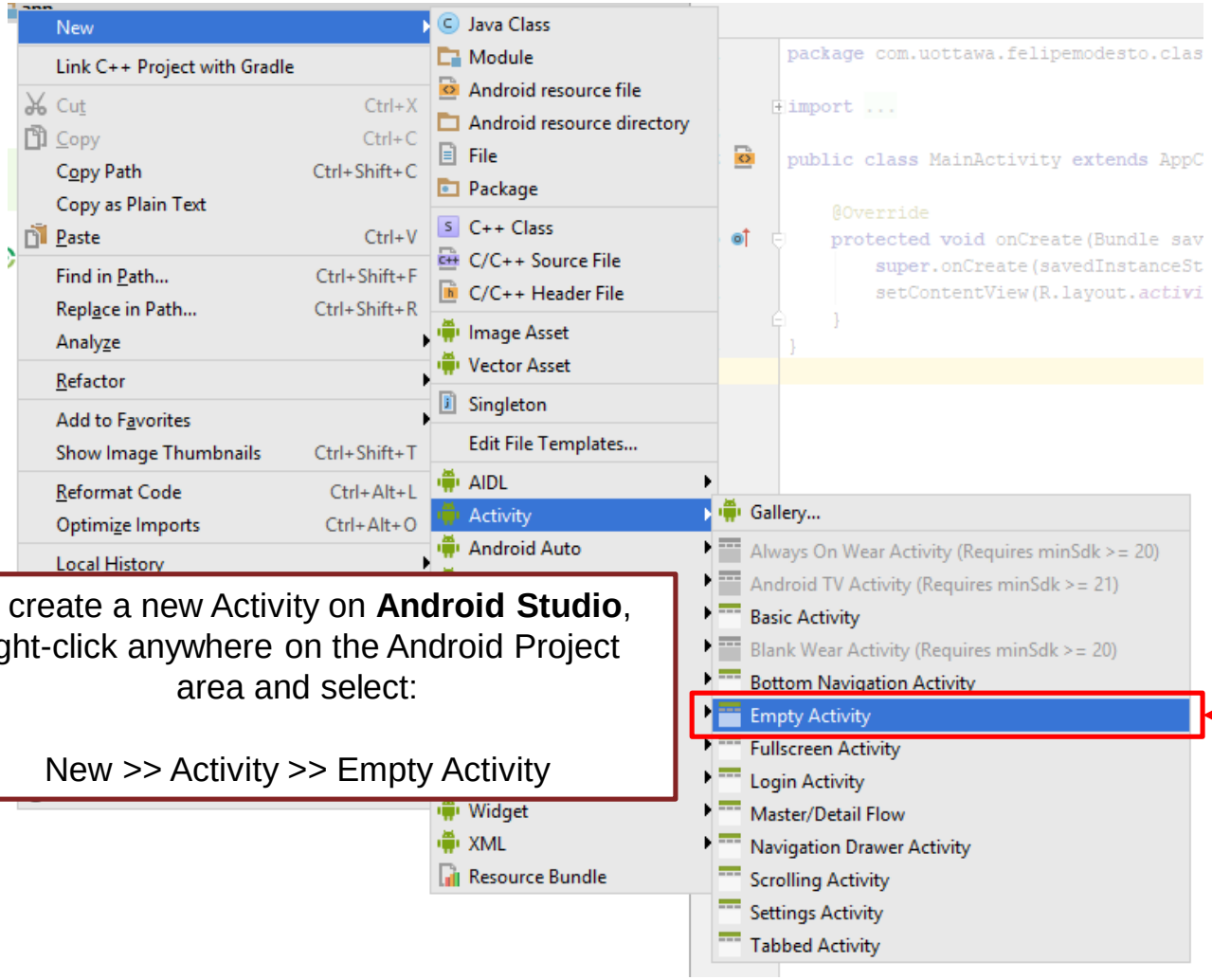
- The Intent object constructor has two parameters
  1. Application Context : Where the application derives from.
     1. If when creating a new Activity, we set the hierarchal parent to the main activity, we pass "this" as the context, otherwise we need to get the application's context using the **getApplicationContext()** function.
  2. Intent Class: Class the intent is requesting

Current Status

- Your emulated application should look vaguely like the one to the side.
- Add an ***OnClick*** function to the Button in the bottom matching the function on a previous slide.
  - This function will open our location on a map application.
- Add an ***OnClick*** function to the ***ViewImage*** matching a function on a previous slide***.***
  - Yes, you can add OnClicks to other objects!
  - This function will open the team logo avatar selector.



Creating a new Activity



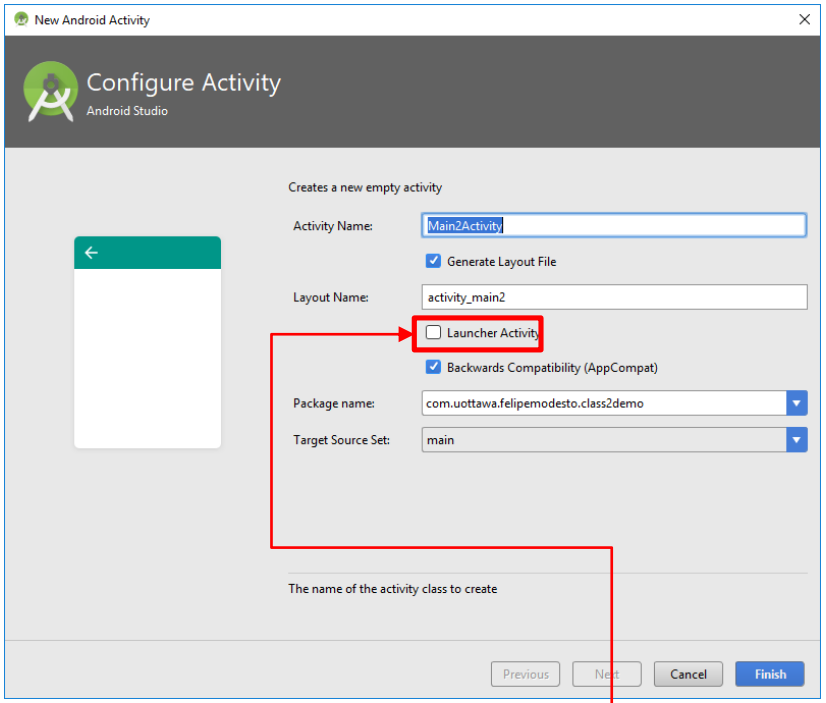
To create a new Activity on **Android Studio**, right-click anywhere on the Android Project area and select:

New >> Activity >> Empty Activity

Select Empty Activity

Creating a new Activity

- Fill the information according to your needs and "Finish" creating the second activity.
- *Note: Creating a new Activity will generate a new .java file for the second activity as well as a new xml layout.*



If selected, this activity will replace the current activity as the one launched when the app is opened.

Second Activity

- Create a second activity that will be used to list the pictures available to the user when changing their profile.
- Remove all items in the hierarchy (e.g.: hello world text) and add a ***VerticalLayout.***
- Add a ***GridLayout*** *to the* ***VerticalLayout*** and create six ***ImageViews*** inside it.
  - Set the Gridlayout's Grid to 2x3:
    - Set "***rowCount***" to 2, and "***columnCount***" to 3
  - Insert and format images as in the previous screen to make the 2 by 3 grid.
    - Indexes in a grid start with the value 0. The top left item should have values: *row 0, Column 0*
    - Position the items in the grid by setting their **"layout_row"** and **"layout_column"** values corresponding to where they are in the grid.
    - Give all image items in the grid the value "1" for the ***"rowspan"*** and **"columnspan"** properties.
- Add button to the ***VerticalLayout*** bellow the ***GridLayout***

Note: *We use the* ***GRIDLAYOUT*** *in this activity, NOT the GRIDVIEW.*

While on the Second Activity

- How do I get back to my main activity?
- Activities are Stacked when new activities are launched, so when the current activity is "done" it will be removed from the stack and the main activity will be called back. All your data will still be in the app, ***textfields*** will still contain the data you may have typed on them.
- Once your activity is pulled from the stack, it's deleted and opening that activity again will instance a new/fresh instance of it.
  - To manage how data is presented you should use controllers, which should inform the application flow.
- You do not need to place Back explicit buttons on android application menus, as Android always has a back button.
  - Pressing it, removes the current activity from the stack and returns to the main activity.

Returning to Main Activity

- Add the code below to your Second Activity Java File.
  - This should be the ***OnClick*** on your Images!
  - This code forwards the ID of the image that was pressed to the Main Activity.

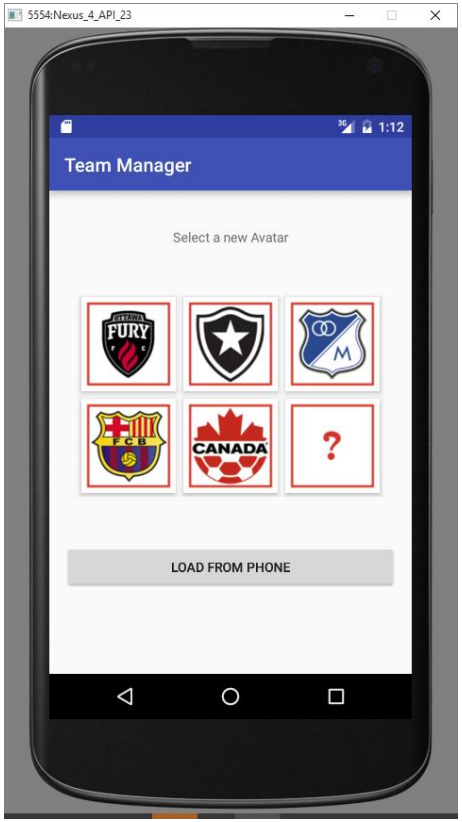
```
public void SetTeamIcon(View view) {
    //Creating a Return intent to pass to the Main Activity
    Intent returnIntent = new Intent();

    //Figuring out which image was clicked
    ImageView selectedImage = (ImageView) view;

    //Adding stuff to the return intent
    returnIntent.putExtra("imageID", selectedImage.getId());
    setResult(RESULT_OK, returnIntent);

    //Finishing Activity and return to main screen!
    finish();
}
```

These lines are comments, please pay attention when copying the code.

Current Status

- Your emulated application should look vaguely like the one on the side.

- Add the SetTeamIcon ***OnClick*** function to each of the Images. All images can map to the same function, available in the next slide.



Processing the result

- Add the function below to your Main Activity Java File.
  - This will process the info we provided in the Return intent.
- The information we passed in the return intent is interpreted and used to set the new image.
- If your IDs are different, update the code to reflect the changes!

Note: Names of Images in the code may differ to the ones you set in your xml file. Check your IDs.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_CANCELED) return;

    //Getting the Avatar Image we show to our users
    ImageView avatarImage = (ImageView) findViewById(R.id.avatarImage);

    //Figuring out the correct image
    String drawableName = "ic_logo_00";
    switch (data.getIntExtra("imageID",R.id.teamid00)) {
        case R.id.teamid00:
            drawableName = "ic_logo_00";
            break;
        case R.id.teamid01:
            drawableName = "ic_logo_01";
            break;
        case R.id.teamid02:
            drawableName = "ic_logo_02";
            break;
        case R.id.teamid03:
            drawableName = "ic_logo_03";
            break;
        case R.id.teamid04:
            drawableName = "ic_logo_04";
            break;
        case R.id.teamid05:
            drawableName = "ic_logo_05";
            break;
        default:
            drawableName = "ic_logo_00";
            break;
    }
    int resID = getResources().getIdentifier(drawableName, "drawable",
getPackageName());
    avatarImage.setImageResource(resID);
}
```

Current Status

- At this point, your application should:
  - Display and update a teams name and address
  - Show the teams Address on Google Maps
  - Update and Show the teams Logo from a predefined list of images.

- Your application currently does not:
  - Load Images from the Device

Loading Images from the Device requires your device to have images loaded to it or a camera accessible, which imposes additional constraints, therefore, the final step is optional and is presented with an educational purpose.

*Loading Images also requires additional **Permissions**, which are generally managed automatically by Gradle, but may need to be added to the **manifest** file.*

Implicit Intents

- Implicit Intent to Camera Feature:
  - ***Intent*** *intent = new* ***Intent****(MediaStore.ACTION_IMAGE_CAPTURE);*
  - *Can add extra content regarding media output, save location, format, etc.*
  - ***onActivityResult*** will receive an image path and will have to decode/handle the related image file.
  - ***(See Next Slide for Code)***

OnActivityResult: Opening a Camera

```
File file = new File(Environment.getExternalStorageDirectory() + "/DCIM/", "image" + new Date().getTime() + ".png");
Uri imgUri = Uri.fromFile(file);
String imgPath = file.getAbsolutePath();
final Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
intent.putExtra(MediaStore.EXTRA_OUTPUT, setImageUri());
startActivityForResult(intent, CAPTURE_IMAGE);
```

<There will Likely be some code in between the variables above and the code below>

```
@Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode != Activity.RESULT_CANCELED) {
            if (requestCode == CAPTURE_IMAGE) {
                ImageView imageView = (ImageView) findViewById(R.id.imgView);
                imageView.setImageBitmap(BitmapFactory.decodeFile(imgPath));
}}}
```

Implicit Intents

- Implicit Intent to Storage:
  - ***Intent*** *intent = new* ***Intent(Intent.ACTION_PICK);***
  - *Can add extra content regarding media type, etc.*
  - ***onActivityResult*** will receive an image path and will have to decode the related image file.
  - ***(See Next Slide for Code)***

OnActivityResult: Getting An Image

```
Intent i = new Intent(Intent.ACTION_PICK,android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
startActivityForResult(i, RESULT_LOAD_IMAGE);
```

<There will Likely be some code in between the variable above and the code below>

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == RESULT_LOAD_IMAGE && resultCode == RESULT_OK && null != data) {
        Uri selectedImage = data.getData();
        String[] filePathColumn = { MediaStore.Images.Media.DATA };
        Cursor cursor = getContentResolver().query(selectedImage,filePathColumn, null, null, null);
        cursor.moveToFirst();
        int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
        String picturePath = cursor.getString(columnIndex);
        cursor.close();
        ImageView imageView = (ImageView) findViewById(R.id.imgView);
        imageView.setImageBitmap(BitmapFactory.decodeFile(picturePath));
}}
```

Curious to Learn more?

- Follow the Android Training Guide to become an expert!
  - https://developer.android.com/training/index.html