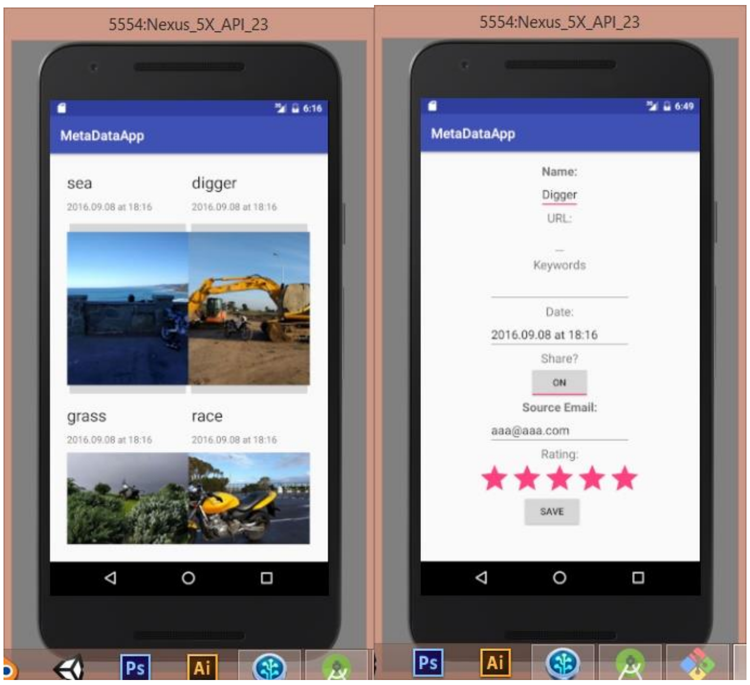
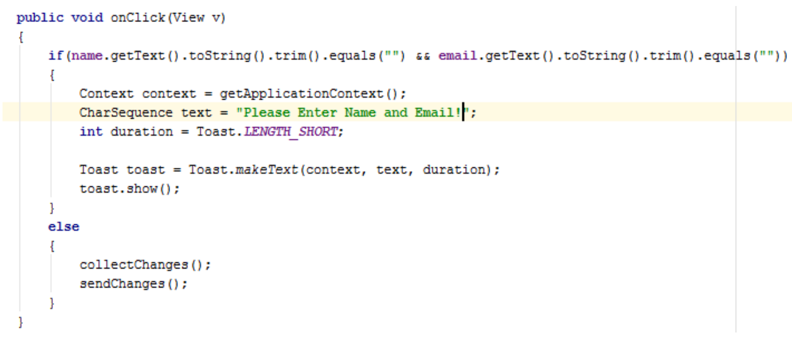
Task 1

- The app contains two activities.

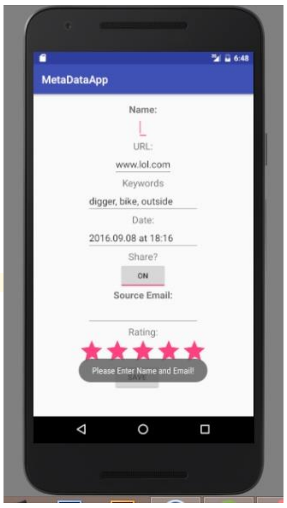


These are the 2 activities for the MetaData app that I have created.

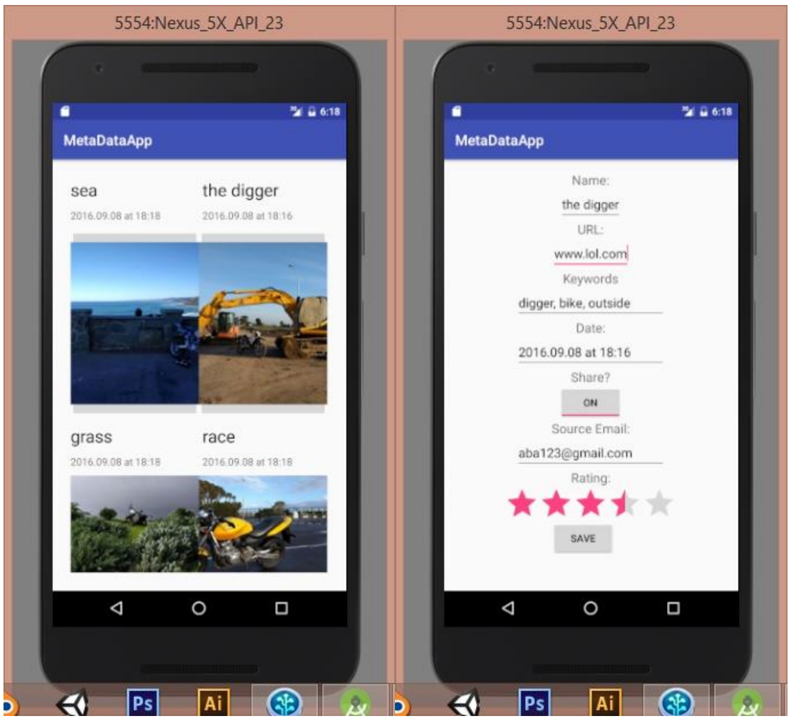
The Necessary field's titles for Name and Email have been bolded to show Importance aswell as having a message displayed if they are not filled out as shown below in code and in app. If no values posted a toast will display an error message and not let the user progress until they fill out the fields.



```
public void onClick(View v)
{
    if(name.getText().toString().trim().equals("") && email.getText().toString().trim().equals(""))
    {
        Context context = getApplicationContext();
        CharSequence text = "Please Enter Name and Email!";
        int duration = Toast.LENGTH_SHORT;

        Toast toast = Toast.makeText(context, text, duration);
        toast.show();
    }
    else
    {
        collectChanges();
        sendChanges();
    }
}
```



- The parcelable protocol has been used and explained (why needed and how it works)



sea 2016.09.08 at 18:18
the digger 2016.09.08 at 18:16
grass 2016.09.08 at 18:18
race 2016.09.08 at 18:18

Name: the digger
URL: www.lol.com
Keywords: digger, bike, outside
Date: 2016.09.08 at 18:16
Share? ON
Source Email: aba123@gmail.com
Rating:
SAVE

Parcelable protocol has been used as it is many times more efficient and quicker than the serializable method for transferring information through intents. The Picture metadata has been stored as a parcel as shown below.

```
@Override
public void writeToParcel(Parcel parcel, int i)
{
    parcel.writeString(Name);
    parcel.writeString(Url);
    parcel.writeString(Keywords);
    parcel.writeString(Date);
    parcel.writeInt(Share ? 1 : 0);
    parcel.writeString(Email);
    parcel.writeInt(Star);
    parcel.writeInt(resID);
}
//constructor
public Picture(String name, String email, int resid)
{
    this.Name = name;
    this.Email = email;
    this.resID = resid;
    this.Share = false;
}
```

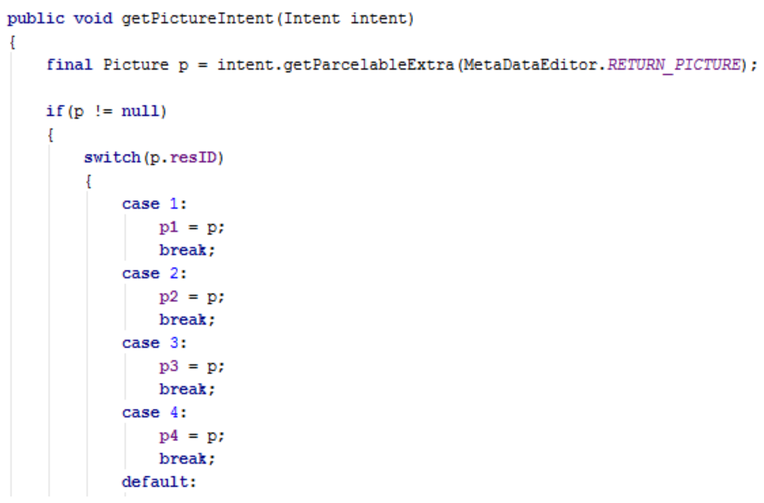
```
//retrieving data from parcel
private Picture(Parcel in)
{
    Name = in.readString();
    Url = in.readString();
    Keywords = in.readString();
    Date = in.readString();
    Share = in.readInt() != 0;
    Email = in.readString();
    Star = in.readInt();
    resID = in.readInt();
}
```

The metadata parcel can easily be transferred through the app with intents as shown below.

This shows the code for sending intent with a parcel and requesting to receive a result with the onActivityResult function.

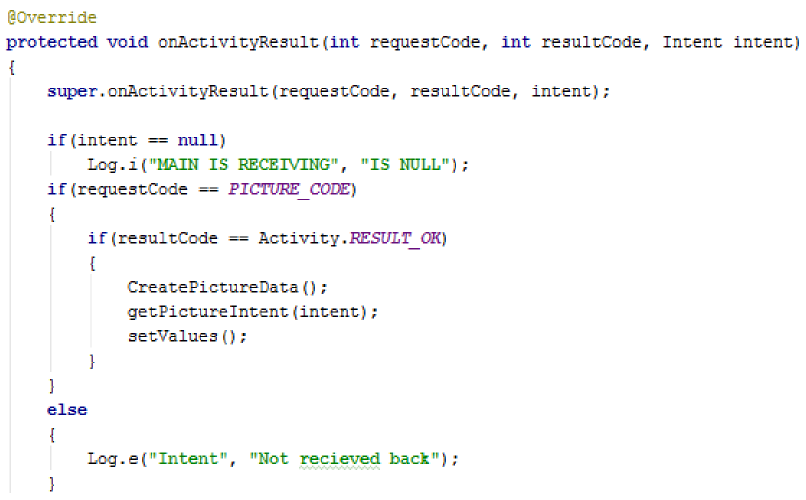
```
public void sendIntent(Picture p)
{
    final Intent intent = new Intent(this, MetaDataEditor.class);
    intent.putExtra(PICTURE, p);

    startActivityForResult(intent, PICTURE_CODE);
}
```

This shows how the main activity will receive the picture parcel, back and decide which picture it is assigned to through the use of an integer resID.



```java
public void getPictureIntent(Intent intent)
{
    final Picture p = intent.getParcelableExtra(MetaDataEditor.RETURN_PICTURE);

    if(p != null)
    {
        switch(p.resID)
        {
            case 1:
                p1 = p;
                break;
            case 2:
                p2 = p;
                break;
            case 3:
                p3 = p;
                break;
            case 4:
                p4 = p;
                break;
            default:
```

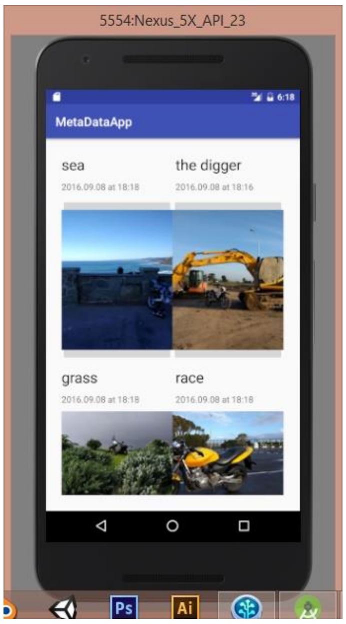
This function checks to make sure that intent is sent back to the main activity successfully before getting the intent and creating errors if no actual intent is sent.



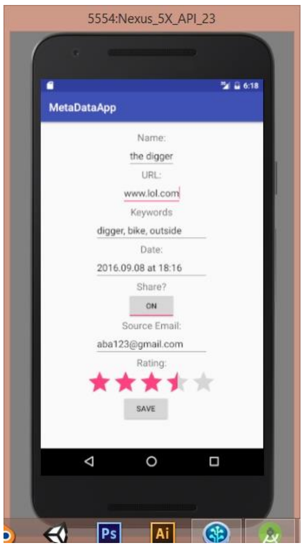
```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent)
{
    super.onActivityResult(requestCode, resultCode, intent);

    if(intent == null)
        Log.i("MAIN IS RECEIVING", "IS NULL");
    if(requestCode == PICTURE_CODE)
    {
        if(resultCode == Activity.RESULT_OK)
        {
            CreatePictureData();
            getPictureIntent(intent);
            setValues();
        }
    }
    else
    {
        Log.e("Intent", "Not recieved back");
    }
```

Styles

The style for the Main activity is designed to be very simplistic with the title of the picture being bold and larger compared to the date which is less significant when a user is reading the information present. The main focus in the scene is the Pictures themselves so they needed to get the attention of the users. In making the Picture's image buttons big makes it much easier for anyone to easily click the correct picture they want to select and easily identify what it is displaying. Putting the information in the grid layout was done to use as much of the provided space as possible without overcrowding it, giving some space between the name, date and pictures. This layout is designed so that the user does not need to scroll the page making it more user friendly.



The style for the Meta data editor activity is more complex than the Main activity. The labels are centre aligned above the input field as labels being above the input field is the recommended method for phones. Having the labels above the input fields allows the user to focus on both the label and field at the same time with rapid processing with least confusion. The central alignment of the labels and input fields is a design choice due to the varying sizes of the different input fields giving the activity some form of conformity, it looks better on the phone compared instead of having everything left aligned and leaving gaps on the right side of the screen. I do not need left aligned labels as there is enough room to fit all the input boxes and labels. Also I don't then need to stretch the different labels to align the input fields properly having the labels on top.


5554:Nexus_5X_API_23

MetaDataApp

Name:
the digger
URL:
www.lol.com
Keywords
digger, bike, outside
Date:
2016.09.08 at 18:16
Share?
ON
Source Email:
aba123@gmail.com
Rating:

SAVE

Task 2

Objective of the test is explained

The Objective is to determine what the users believe to be the most appropriate and readable label and information display sizes and styles. The test is to get Usability Research to find the recommended method for displaying text.

Method used for the test

The test was conducted through a google survey form which showed the users a picture of the phone with the varying font sizes in different layouts and then the questions relating to preference and readability.

How were the font sizes determined & why were these selected?

The font sizes were determined by collecting 3 of the varying sizes of text Appearance resources in the textView style selection.

The font sizes were selected to get a wide range of different sizes ranging from small to large.

What font sizes were shown to the user?

- Small: 14sp
- Medium: 18sp
- Large: 22sp

How was the code modified to show the different font sizes?

The code was modified through the Design Layout changing the style of the labels.

The users were shown three different compiled versions of an app to show the varying font sizes.

Test Participants Smartphone Experience

The first tester was an adult in their early twenties who does not need glasses and has moderate smartphone experience.

The second tester was an adult in their early twenties who does not need glasses and has little to no smartphone experience.

The third tester was a middle aged adult who does need glasses and has moderate smartphone experience.

The last tester was an adult in their early twenties who does need glasses and has high smartphone experience.

Results from the Usability Research are presented in a table

	Can you read text?	Prefer size of labels?	Prefer size of information
Small Font	No, No, No, Yes	No, No, No, No	No, No, No, No
Medium Font	Yes, Yes, Yes, Yes	No, No, No, Yes	Yes, Yes, No, Yes
Large Font	Yes, Yes, Yes, Yes	Yes, Yes, Yes, No	Yes, Yes, Yes, No

Table 1 is showing the results from the user survey test in simplified form.

To coordinate all the feedback provided into a clear format I have summarised the feedback received into whether the testers liked or disliked the certain font sizes. The No's represent that the users were dissatisfied or could not read with the certain size of font. The Yes's in the table represent what size of font the testers liked and could read.

Recommendation is provided for a font size

Labels should be displayed in a large sized font at around 22sp for best visibility.

Information and text can be displayed at either medium(18sp) or large font(22sp) maybe with the option to zoom into the screen.

Nothing should be displayed in small font (14sp) as no one preferred it and most people couldn't read it.

Reflections on method used

This method worked well as users were easily able to see how large or small the text would be due to the screenshots of the emulator taken. It was very easy for me to collect the feedback and compile it into a table since I used google forms which saved all the results from each user and put it all on one page.

Reflections on any surprising/expected or expected findings

I knew that no one would like the small font as it was hard to read.

I expected that Labels should always be large for the user to easily understand what it is that they are entering.

I also expected some people wouldn’t mind the medium or large text which was depending on their eyesight and age.