

CSEE5590/CS490 APS
Programming for Web/Cloud / Mobile Applications

Part - Mobile Development

Lab

ASSIGNMENT - 2

BALACHANDAR KULALA

(Student ID:12543027)

Goal of the Assignment:

To provide good practice on android development using the Services, Storage usage, API usage, Grid layout design

Section-1: Storage usage

Aim:

To get in practice of Storage usage in Android system. Mainly focus on SQLite/ Firebase or Storage usages.

Description of Work:

Here We tried to use the Storage usage by consuming SQLite or Firebase database.

Technical Section:

Majorly used technical skills.

- Android technology
- Core Java
- XML, Firebase, SQLite Databases
- JSON

Development Section

This section contains 2 buttons “FIREBASE DATE” AND “SQLITE DATA”:

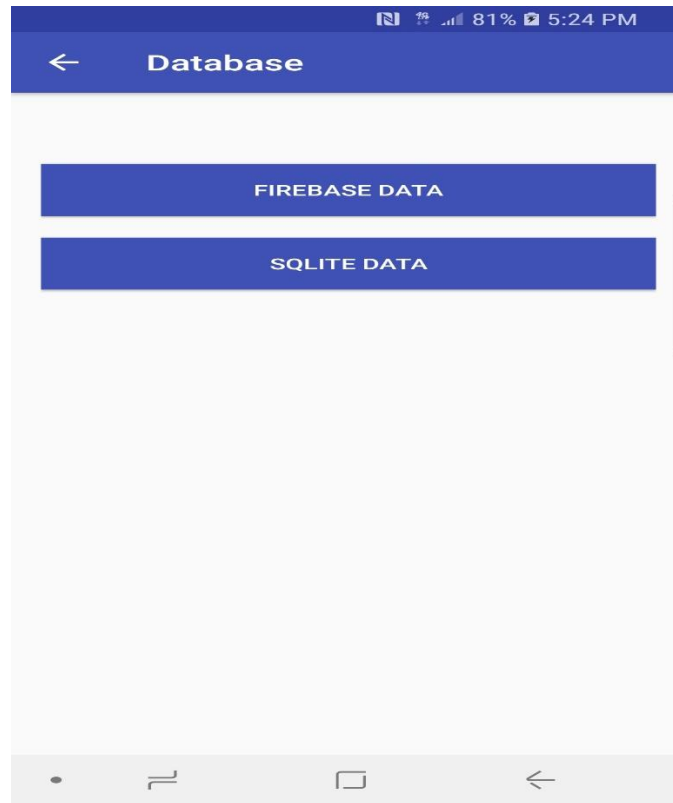
i) Firebase Data:

When the user clicks on this button it displays the data from firebase database.

The input screen is shown below:

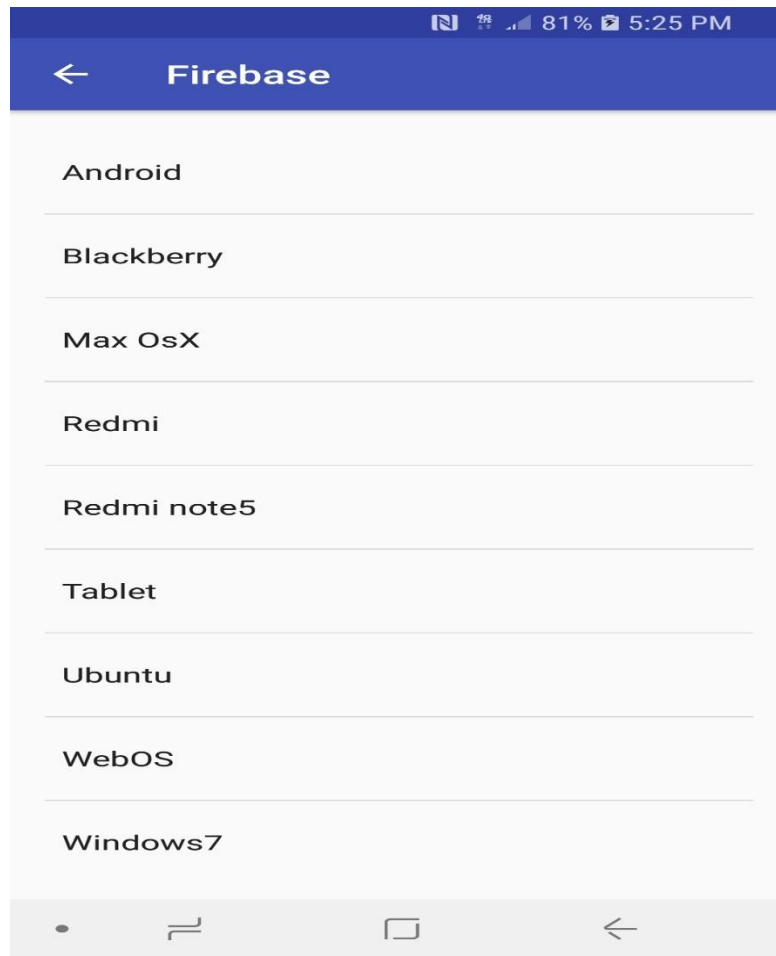
When user clicks on Firebase data button, it grabs the data from firebase database and displays in the new activity.

Screenshot:



After successful click on the “FIREBASE DATA” button, it collects the data from database and displays in next activity. The corresponding output screen shown below.

The corresponding code for this logic is shown in following screen: It authenticates the user credentials, if it exists allows to the home page else denies.



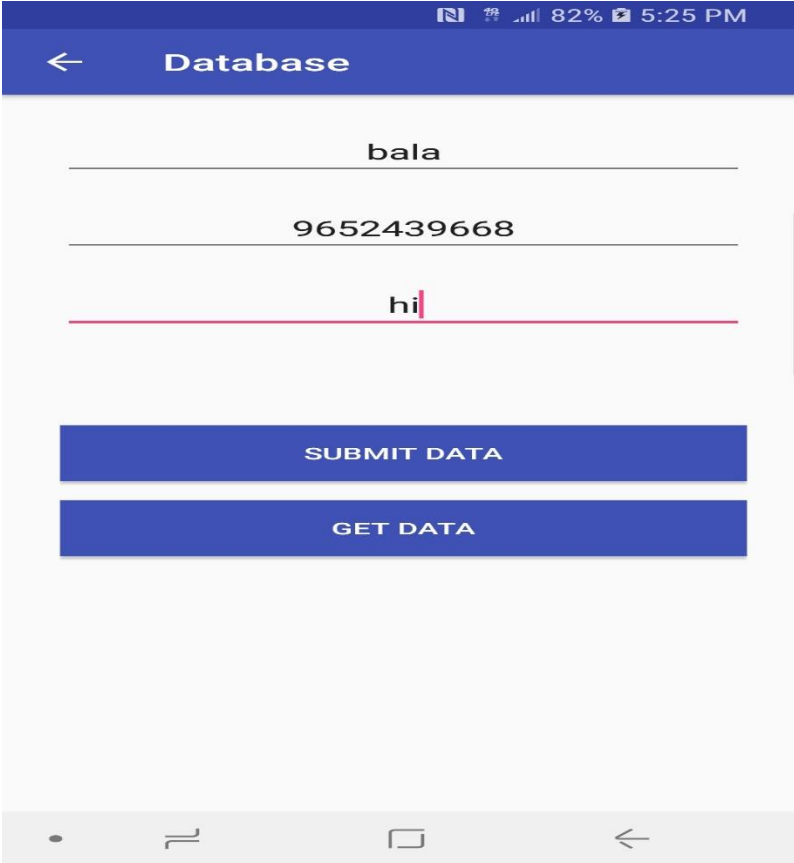
The corresponding code logic is shown below from android studio.

```
public class FirebaseActivity extends AppCompatActivity {
    // DatabaseReference dref = FirebaseDatabase.getInstance().getReference();
    DatabaseReference dref;
    ListView listview;
    ArrayList<String> list=new ArrayList<>();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_firebase);
        android.support.v7.app.ActionBar actionBar = getSupportActionBar();
        actionBar.setHomeButtonEnabled(true);
        actionBar.setDisplayHomeAsUpEnabled(true);
        listview=(ListView)findViewById(R.id.listview);
        final ArrayAdapter<String> adapter=new ArrayAdapter<>(<context> this,android.R.layout.simple_dropdown_item_1line,list);
        listview.setAdapter(adapter);dref=FirebaseDatabase.getInstance().getReference();
        dref.addChildEventListener(new ChildEventListener() {
            @Override
            public void onChildAdded(DataSnapshot dataSnapshot, String s) {
                adapter.add(
                    (String) dataSnapshot.child("description").getValue());
                list.add(dataSnapshot.getKey());
                // list.add(dataSnapshot.getValue(String.class));
                adapter.notifyDataSetChanged();
            }
            @Override
            public void onChildChanged(DataSnapshot dataSnapshot, String s) {
            }
            @Override
            public void onChildRemoved(DataSnapshot dataSnapshot) {
                list.remove(dataSnapshot.getKey());
                adapter.notifyDataSetChanged();
            }
        });
    }
}
```

ii) Open SQLite database:

In this case we have tried to implement Open SQLite database to store the employee data entered and, we display the data from the database in the TextView.

The input screen shown below:



The screenshot shows an Android application interface with a blue header bar containing a back arrow and the title "Database". Below the header, there are three text input fields. The first field contains the text "bala", the second field contains "9652439668", and the third field contains "hi" with a red cursor at the end. Below the input fields, there are two blue buttons with white text: "SUBMIT DATA" and "GET DATA". The bottom of the screen shows the standard Android navigation bar with icons for back, home, and recent apps.

This screen asks user to enter the employee details like, name, phone number and description of work. When we click on Submit button with proper input data. It successfully inserts the data into SQLite database.

This screen also validates the user inputs and emptiness of the inputs and some basic validations. When user clicks on the Get data button it displays the data in TextView.

Output Screen:

Database

Enter Name

Enter Phone Number

Enter comment

SUBMIT DATA

GET DATA

Data Submit Successfully

The corresponding code logic from android studio is displayed below.

```
public void onCreate(SQLiteDatabase database) {  
    String CREATE_TABLE="CREATE TABLE "+TABLE_NAME+" (" +KEY_ID+" INTEGER PRIMARY KEY, "+KEY_Name+" VARCHAR, "+KEY_PhoneNumber+"  
    database.execSQL(CREATE_TABLE);  
}  
  
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
    db.execSQL("DROP TABLE IF EXISTS "+TABLE_NAME);  
    onCreate(db);  
}
```

The above code shows us, how we created the data table with required column parameters.

Bottom block shows, how we drop the table if we wish to or if already exist with wrong data.

The following code snippet explains the insertion query and its results.

```

public void DBCreate(){
    SQLITEDATABASE = openOrCreateDatabase( name: "DemoDataBase", Context.MODE_PRIVATE, factory: null);
    SQLITEDATABASE.execSQL("CREATE TABLE IF NOT EXISTS demoTable(id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, name VARCHAR, phone_number VARCHAR, subject VARCHAR);");
}

public void SubmitData2SQLiteDB(){
    Name = GetName.getText().toString();
    PhoneNumber = GetPhoneNumber.getText().toString();
    Subject = GetSubject.getText().toString();
    CheckEditTextIsEmptyOrNot( Name,PhoneNumber, Subject);
    if(CheckEditTextEmpty == true)
    {
        //SQLITEDATABASE = helper.getWritableDatabase();
        SQLiteQuery = "INSERT INTO demoTable (name,phone_number,subject) VALUES ('"+Name+"', '"+PhoneNumber+"', '"+Subject+"')";
        SQLITEDATABASE.execSQL(SQLiteQuery);
        Toast.makeText( context DatabaseActivity.this, text: "Data Submit Successfully", Toast.LENGTH_LONG).show();
        ClearEditTextAfterDoneTask();
    }
}

```

Section-2: Service usage

Aim:

To get in practice of consuming services given by third party entities or inbuilt/default services.

Description of Work:

Here We tried to implement sensor-based service especially touch sensor of the android device.

Technical Section:

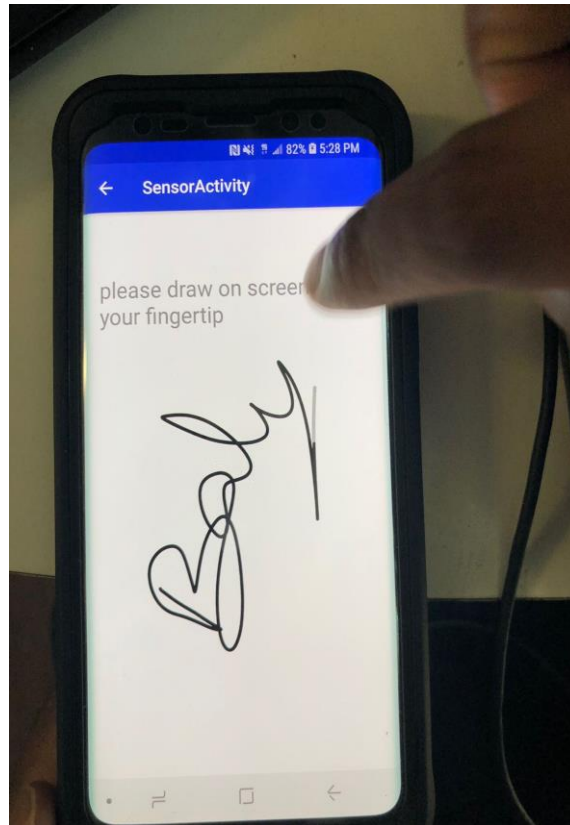
Majorly used technical skills.

- Android technology
- Core Java
- XML
- JSON

Development Section

Here we have developed an activity that uses the service related to touch sensor. Actual implementation shows how the touch sensor works by drawing a line on the screen along with the fingertip.

The corresponding output screen shown below:



The corresponding code snippet from android studio is shown below:

```
public class SensorActivity extends AppCompatActivity implements GestureOverlayView.OnGesturePerformedListener {
    private GestureLibrary gestureLib;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        android.support.v7.app.ActionBar actionBar = getSupportActionBar();
        actionBar.setHomeButtonEnabled(true);
        actionBar.setDisplayHomeAsUpEnabled(true);

        GestureOverlayView gestureOverlayView = new GestureOverlayView( context this);
        View inflate = getLayoutInflater().inflate(R.layout.activity_sensor, null);
        gestureOverlayView.setGestureColor(Color.BLACK);

        gestureOverlayView.addView(inflate);
        gestureOverlayView.addOnGesturePerformedListener(this);
        gestureLib = GestureLibraries.fromRawResource( context this, R.raw.gesture);
        //gestureOverlayView.setGestureColor(Color.parseColor(this,R.raw.gesture));
        if (!gestureLib.load()) {
            finish();
        }
        setContentView(gestureOverlayView);
    }
}
```


Section-3: API usage

Aim:

To get in practice of the Machine Learning/ Social Media related API with android studio.

Description of Work:

Here We tried to implement Machine learning activities using API. This describes the usage of text recognition, facial expression recognition, barcode reader, label detector.

Technical Section:

Majorly used technical skills.

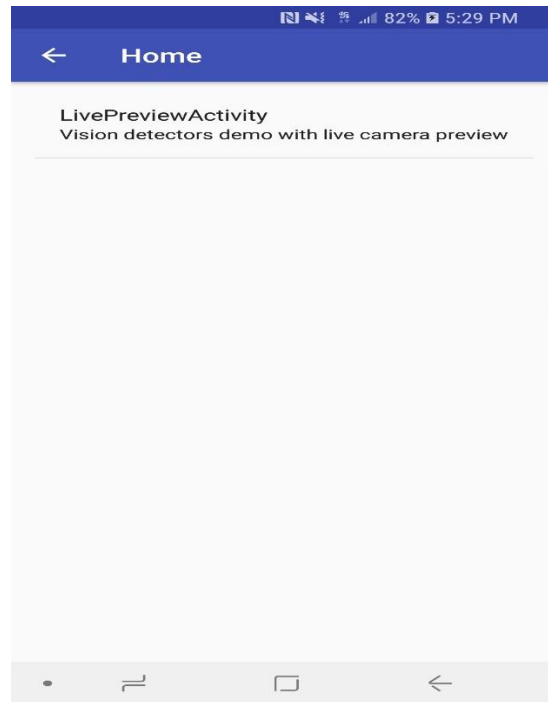
- Android technology
- Core Java
- XML
- Machine Learning API

Development Section:

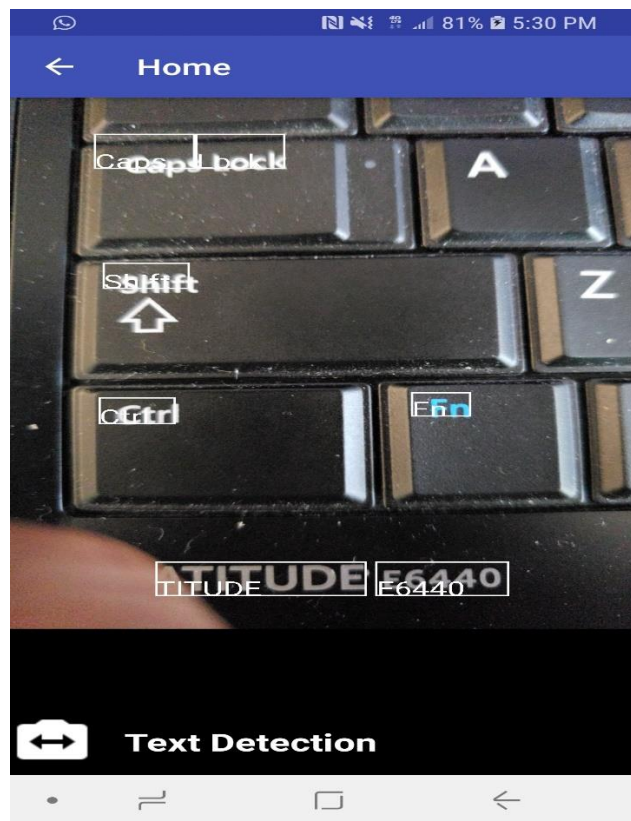
This activity contains a screen which will open the back camera first and based on our selection it performs the actions. If we selected the text recognition it displays the text found in the back camera, similarly if we try to use bar code scanner, it displays the bar code value, if we chosen the facial expression option it displays like smile, anger and more.

The input screen contains the text to touch, when user touches the text, it opens the camera to capture the background. Then at the bottom we can select the option that we need to experience.

The input screen:

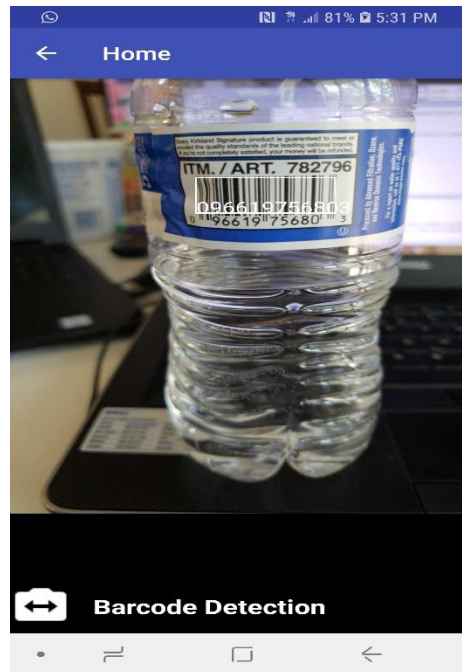


The following are the output screens and corresponding code logic snippet.



This is the text recognition output screen.

The following screen shows the barcode detection output screen:



```
@Override
protected void onSuccess(
    @NonNull List<FirebaseVisionBarcode> barcodes,
    @NonNull FrameMetadata frameMetadata,
    @NonNull GraphicOverlay graphicOverlay) {
    graphicOverlay.clear();
    for (int i = 0; i < barcodes.size(); ++i) {
        FirebaseVisionBarcode barcode = barcodes.get(i);
        BarcodeGraphic barcodeGraphic = new BarcodeGraphic(graphicOverlay, barcode);
        graphicOverlay.add(barcodeGraphic);
    }
}
```

```
public ImageLabelingProcessor() {
    detector = FirebaseVision.getInstance().getVisionLabelDetector();
}
```

```
@Override
protected void onSuccess(
    @NonNull List<FirebaseVisionLabel> labels,
    @NonNull FrameMetadata frameMetadata,
    @NonNull GraphicOverlay graphicOverlay) {
    graphicOverlay.clear();
    LabelGraphic labelGraphic = new LabelGraphic(graphicOverlay, labels);
    graphicOverlay.add(labelGraphic);
}
```



Section-4: Layout usage

Aim:

To get in practice of different layouts usage like implementing Grid layout, relational, inline and web layout.

Description of Work:

Here We tried to implement Grid layout in new activity, it displays the grid with full of default images, when user clicks on the any one of the grid image it displays the image in full screen.

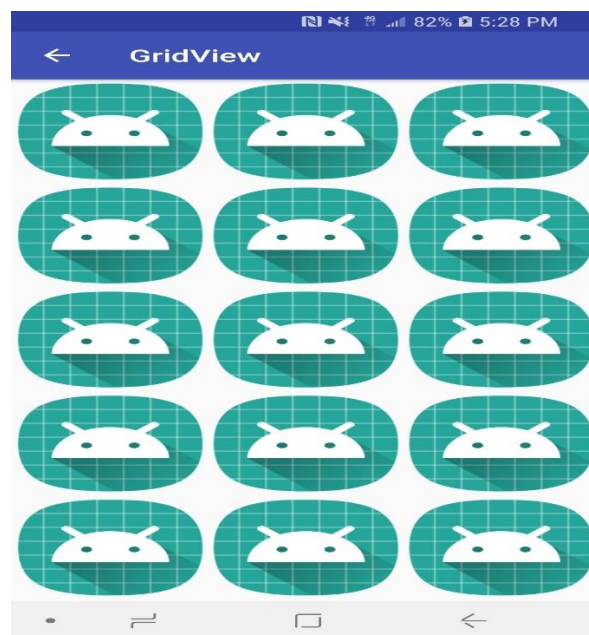
Technical Section:

Majorly used technical skills.

- Android technology
- Core Java
- XML
- GRID layout

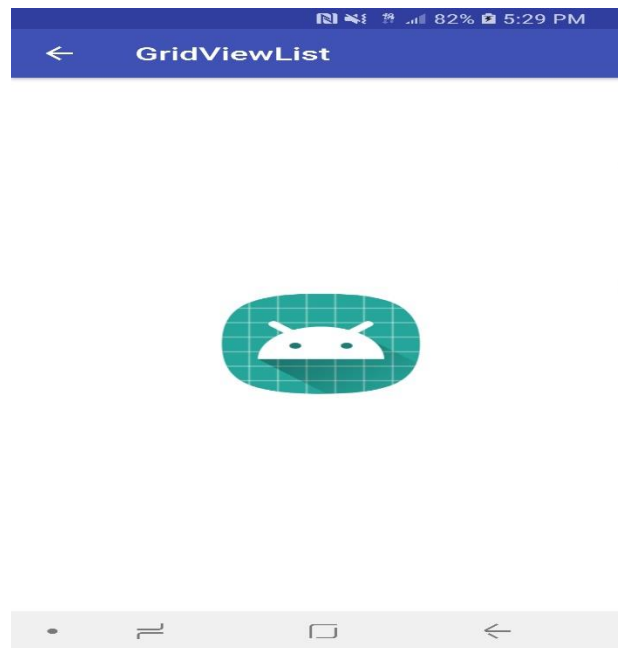
Development Section:

This screen contains grid full of images, when user clicks on the grid image it is displayed in the new window.



The output screen shown below:

When the user clicks on individual image it displays in new activity.



The corresponding code snippet from android studio shown below:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_grid);
    simpleGrid = (GridView) findViewById(R.id.simpleGridView);
    android.support.v7.app.ActionBar actionBar = getSupportActionBar();
    actionBar.setHomeButtonEnabled(true);
    actionBar.setDisplayHomeAsUpEnabled(true);
    // init GridView
    // Create an object of CustomAdapter and set Adapter to GridView
    CustomAdapter customAdapter = new CustomAdapter(getApplicationContext(), logos);
    simpleGrid.setAdapter(customAdapter);
    // implement setOnItemClickListener event on GridView
    simpleGrid.setOnItemClickListener((parent, view, position, id) -> {
        // set an Intent to Another Activity
        Intent intent = new Intent( packageContext: GridActivity.this, SecondActivity.class);
        intent.putExtra( name: "image", logos[position]); // put image data in Intent
        startActivity(intent); // start Intent
    });
}
```

GitHub LINK

<https://github.com/bkkhf/Mobilelab2>

DEMO LINK

<https://youtu.be/xu3iUOdUcy8>

References:

- <https://www.w3schools.com>
 - <https://www.udacity.com/>
 - <http://developer.android.com/guide/index.html>
- <https://www.coursera.org/courses?languages=en&query=android>
- <https://developer.android.com/docs/>