Author: Daniel DANG
School of Computing (SoC), Eastern Institute of Technology (EIT)
Email: daniel.dang.nz@gmail.com

Lab 4: Multiple Activities in Android by using Intent
Create College Program Management app

Objectives

This practical lab shows students how to work with multiple-activity and How to **exchange data** between **those Activities** by using "Intent" object.

In this tutorial, students will develop a mobile application called **College Program Management**.

Upon the completion of the practical lab, students learn:
How to design Graphic User Interface (GUI) in Android:

- Use both RelativeLayout & LinearLayout and learn how to design nested layout;
- Use Visual controls: TextView, ImageView, Button, Radio Buttons, WebView, ListView, Spinner (drop-down menu);
- Learn how to use ListView and Spinner for displaying an array of items;

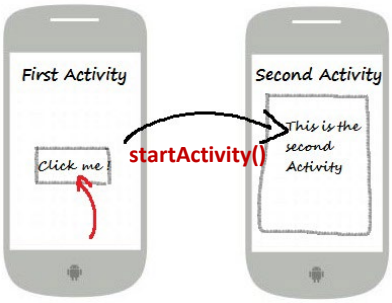
How to code java in Android:

- Learn how to integrate and modify the "Activity lifecycle" in the app: onCreate -> onStart -> onResume -> onPause -> onStop -> onDestroy or onRestart;
- Learn how to create Multiple-Activity app: use "Intent" object to navigate amongst Activities;
- Learn How to use ListView element and Adapter to populate items inside ListView;
- Learn to use XML Parser technique to process data stored in XML file;
- Learn to use SMS API in Android for sending SMS
- Learn to use "implicit Intent" to send email in Android

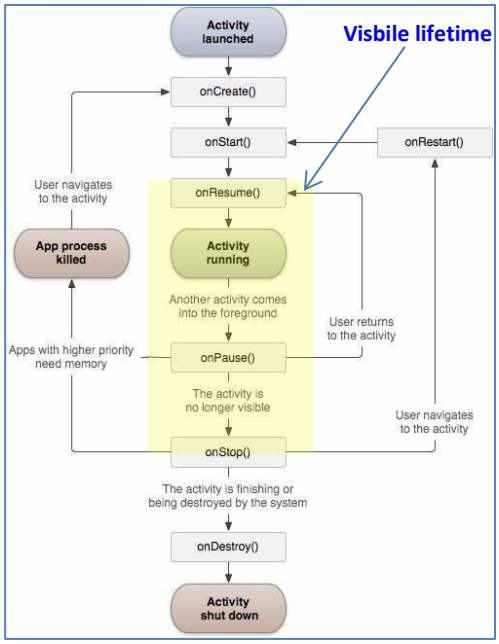
Student will create the app called "ACProgramManager":

- Part 1: Analyse app function and design its main user interface
- Part 2: Add the second Activity: "AboutAMES_Activity"
- Part 3: Add the third Activity: "AboutAC_Activity"
- Part 4: Add the fourth Activity: "ListViewProgrammes_Activity"
- Part 5: Add the fifth Activity: "ProgrammeDetail_Activity"
- Part 6: Add the sixth Activity: "Enrol_Activity"
- Part 7: Add the seventh Activity: "ContactUs_Activity"

Activity



- **An Activity** is an Android component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map. Each activity is given a window in which to draw its user interface.
- **Life cycle: Android system initiates its program** with in an Activity starting with a call on **onCreate() callback** method. There is a sequence of callback methods that start up an activity and a sequence of callback methods that tear down an activity as shown in the below Activity life cycle diagram:



Callback	Description
onCreate()	This is the first callback and called when the activity is first created.
onStart()	This callback is called when the activity becomes visible to the user.
onResume()	This is called when the user starts interacting with the application.
onPause()	The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.
onStop()	This callback is called when the activity is no longer visible.
onDestroy()	This callback is called before the activity is destroyed by the system.
onRestart()	This callback is called when the activity restarts after stopping it.

When we develop an app, we usually deal with several Activities (or associated screens) in Android. In Android, **Intents are the objects** for sending and receiving data between Android Activity.

Intents

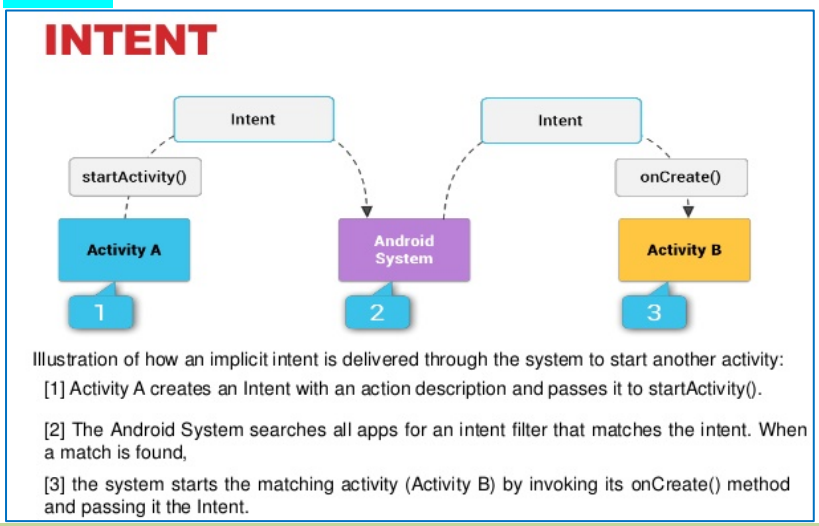
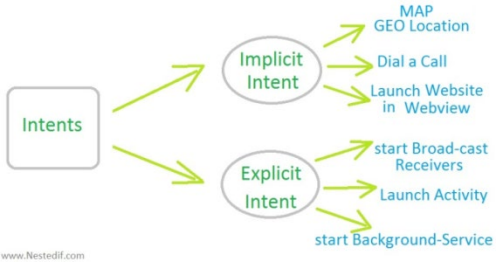


Illustration of how an implicit intent is delivered through the system to start another activity:

[1] Activity A creates an Intent with an action description and passes it to startActivity().

[2] The Android System searches all apps for an intent filter that matches the intent. When a match is found,

[3] the system starts the matching activity (Activity B) by invoking its onCreate() method and passing it the Intent.



An Intent is a **messaging object** you can use to **request an action** from another app component. Although intents facilitate communication between **Android components** in several ways, there are three fundamental use-cases:

To start an activity:

- **An Activity represents a single screen in an app**. You can start a new instance of an Activity by passing an **Intent to startActivity().** The **Intent** describes the activity to start and carries any necessary data.
- If you want to receive a result from the activity when it finishes, call **startActivityForResult().** Your activity receives the result as a separate Intent object in your activity's onActivityResult() callback. For more information, see the Activities guide.

There are two types of intents:

- **Explicit intents** specify the component to start by name (the fully-qualified class name). You'll typically use an explicit intent to start a component in your own app, because you know the class name of the activity or service you want to start. For example, start a new activity in response to a user action or start a service to download a file in the background.
- **Implicit intents** do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it. For example, if you want to show the user a location on a map, you can use an implicit intent to request that another capable app show a specified location on a map.

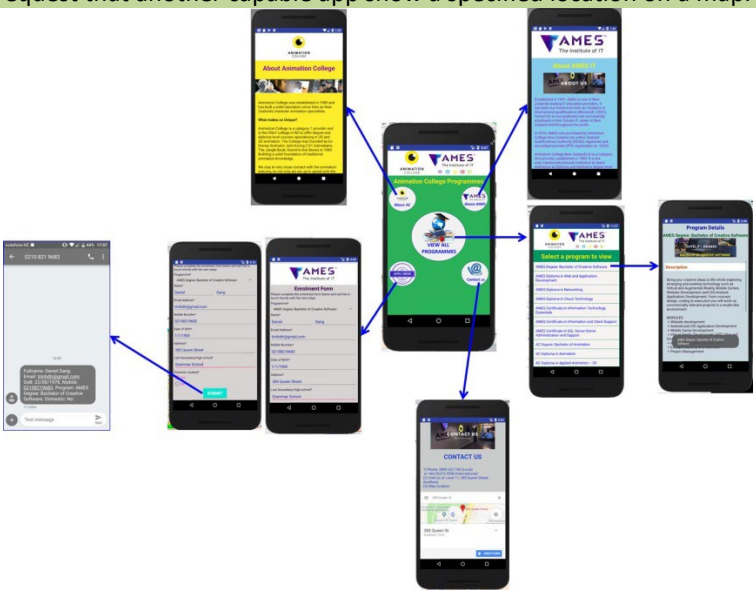


Figure 1: App wireframe

References:

[1]. Intents and Intent Filters: https://developer.android.com/guide/components/intents-filters.html
[2]. Intents and Filters: https://www.tutorialspoint.com/android/android_intents_filters.htm
[3]. Activities: https://developer.android.com/guide/components/activities.html
[4]. Android Activity: https://www.tutorialspoint.com/android/android_acitivities.htm

PART 1: Analyze and design "main layout"

Step 1: Create a new Android Project

- Application name: **ACProgramManager_[yourname]**
- Company Domain: **ac.ames.project.[yourname]**
- Target Android devices: **Phone and Tablet**;
  - Minimum SDK: **API 21: Android 5.0**
- Add an Activity to Mobile: **Empty Activity**;
  - Activity name: **MainActivity**
  - Layout name: **activity_main**

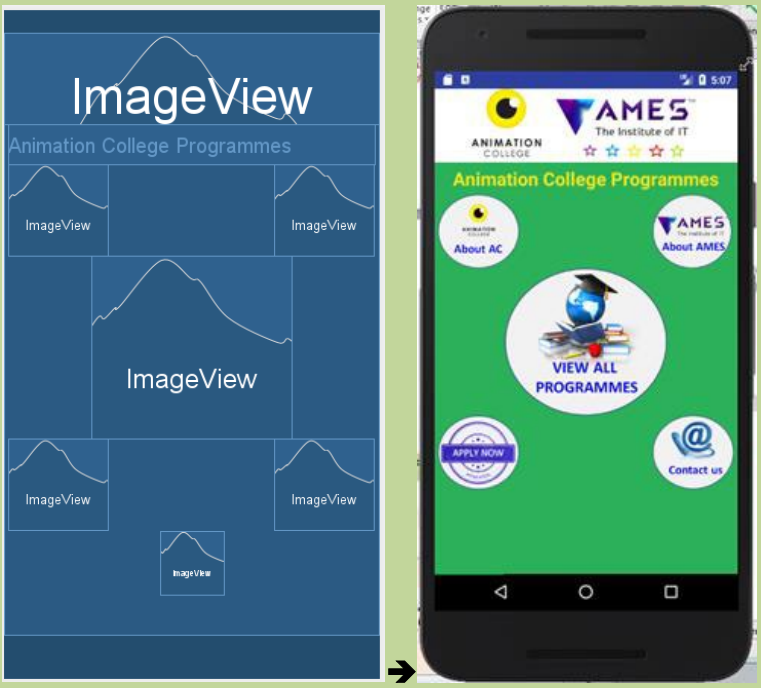
+ Target devices: run on 71.3% devices
_Smartphone & Tablet
_Android KitKat (5.0) & API21 (min)

Step 2: Analyze & design the "main layout": activity_main.xml

+ The main layout (activity_main.xml) is the "main screen" or "home screen" of the app.
When the app is launched, this home screen will be opened up first. On the home screen, there are **few options** for users to naviage to other screens such as:

- **About & AMES:** Show a short description of AMES IT: introduction & partners
- **About Animation College:** Show a short description of Animation College: **:** introduction & partners
- **View All Programmes:** Display all degree, diplomas and certificates offered by both Animation College and AMES IT (12 in totals: 7 AMES programmes & 5 Animation programmes). Each programme contains the information: *(1) programme description & modules, (2) NZQF level, (3) duration, (4) start dates, (5) tuition fee, and (6) career opportunities;*
- **Online Application:** Show enrolment form for users to fill in and send via SMS or via Email to college;
- **College contact:** Display phone number, addres, and our Animation College location on Google Map;
- **App Author:** Display app developer information (headshot, name, email, studentID)

⇨ **The "main layout" structure look like**:



+ In this part, the main layout is quite simple. The main layout contains 2 below visual elements:

- **1 ImageView: displays college logo;**
- **1 TextView: displays title "Animtion College Programmes";**

+ Open and edit the activity_main.xml as below:

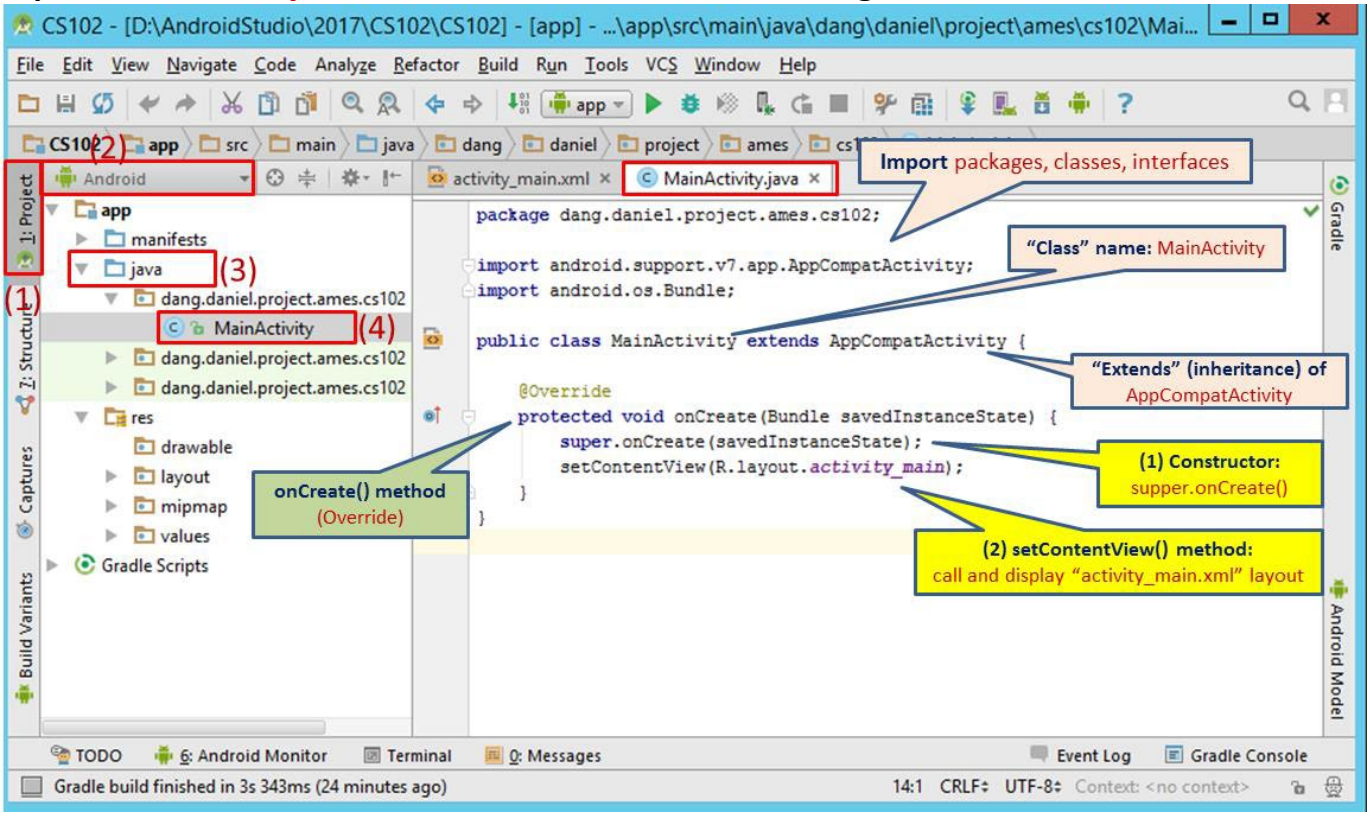
activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#29b15b"
    tools:context=".MainActivity">

    <!--ImageView: display college logo-->
    <ImageView
        android:id="@+id/college_logo"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:background="#FFFFFFFF"
        android:contentDescription="college logo"
        android:gravity="center"
        android:scaleType="fitCenter"
        android:src="@drawable/collegelogo_transparent" />

    <!--TextView: display title-->
    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/college_logo"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:gravity="center"
        android:padding="5dp"
        android:text="Animation College Programmes"
        android:textColor="#f1e61a"
        android:textSize="25sp"
        android:textStyle="bold" />
</RelativeLayout>
```

+ Run your app on AVD (Nexus 5X) to see the result:

Step 3: Main Activity: java coding

+ Open MainActivit.java file and have a look at the auto-generated codes:



+ Open and have a look at AndroidManifest.xml file:

You will see that **there is only one Activity (.MainActivity) in project structure** and this activity is **"launcher" property** or the app will opne this Acitivty first when the app is launched:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="project.ames.ac.nz.lab_multiactivity_daniel">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

PART 2: Add the second Activity: "AboutAMES_Activity"

Step 1: Add a new Activity to the project

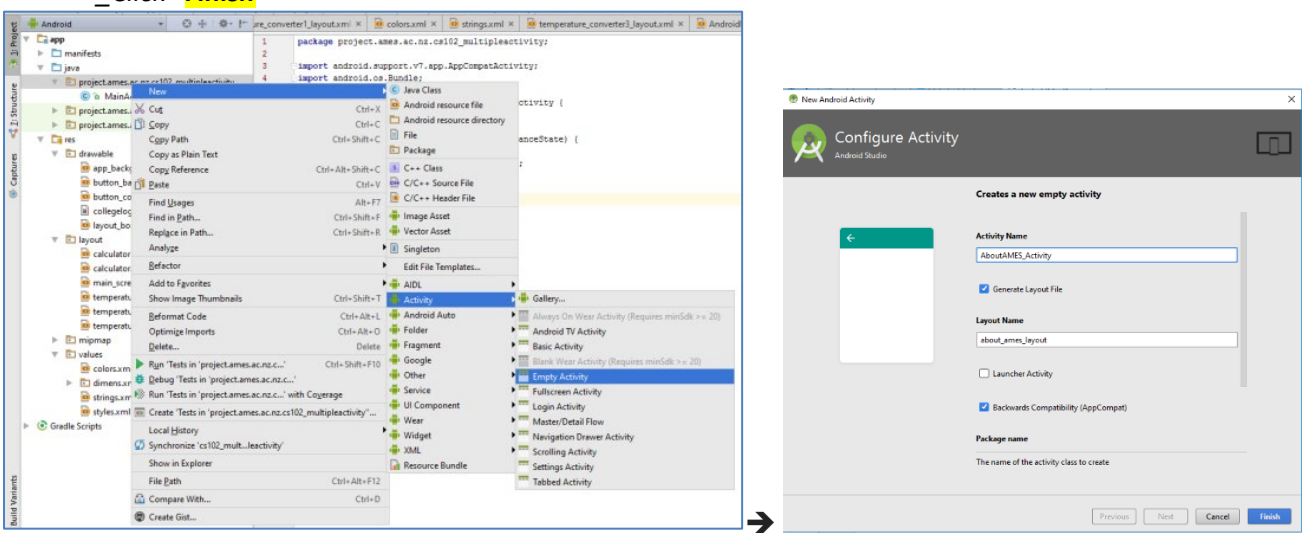
+ Add a new Activity, called "AboutAMES _Activity" and its associated "about_ames_layout"

*_Right click on the folder containing **MainActivity** → "**New**" → "**Activity**" → "**Empty Activity**" → a "Configure Acitivy" window is appeared:*

*_Enter the "Activity Name": **AboutAMES_Activity***

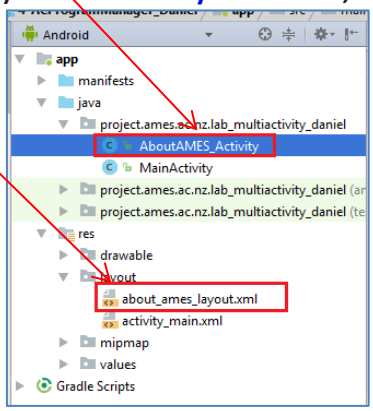
*_Enter the "Layout Name": **about_ames_layout***

*_Click "**Finish**"*



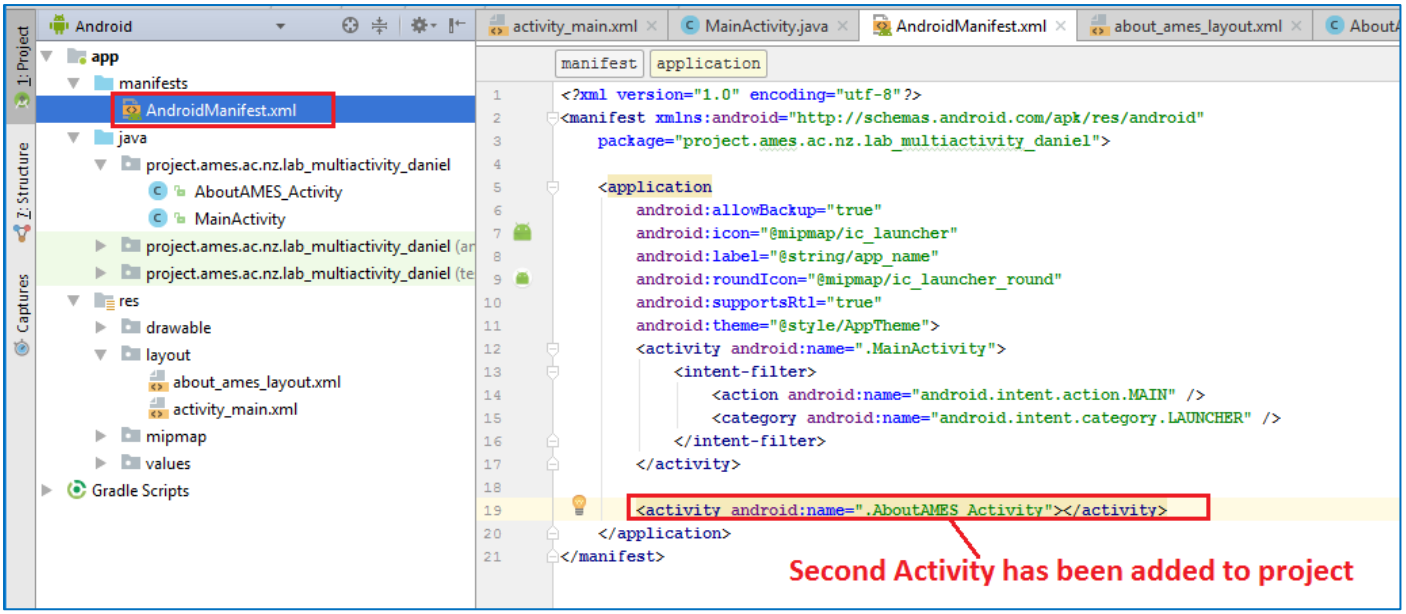
+ Now when you view your Android project, you can see:

- **An Activity (AboutAMES_Activity.java) is added to the same folder containing "MainActivity.java";**
- **a layout (about_ames_layout.xml) is added to "layout" folder;**



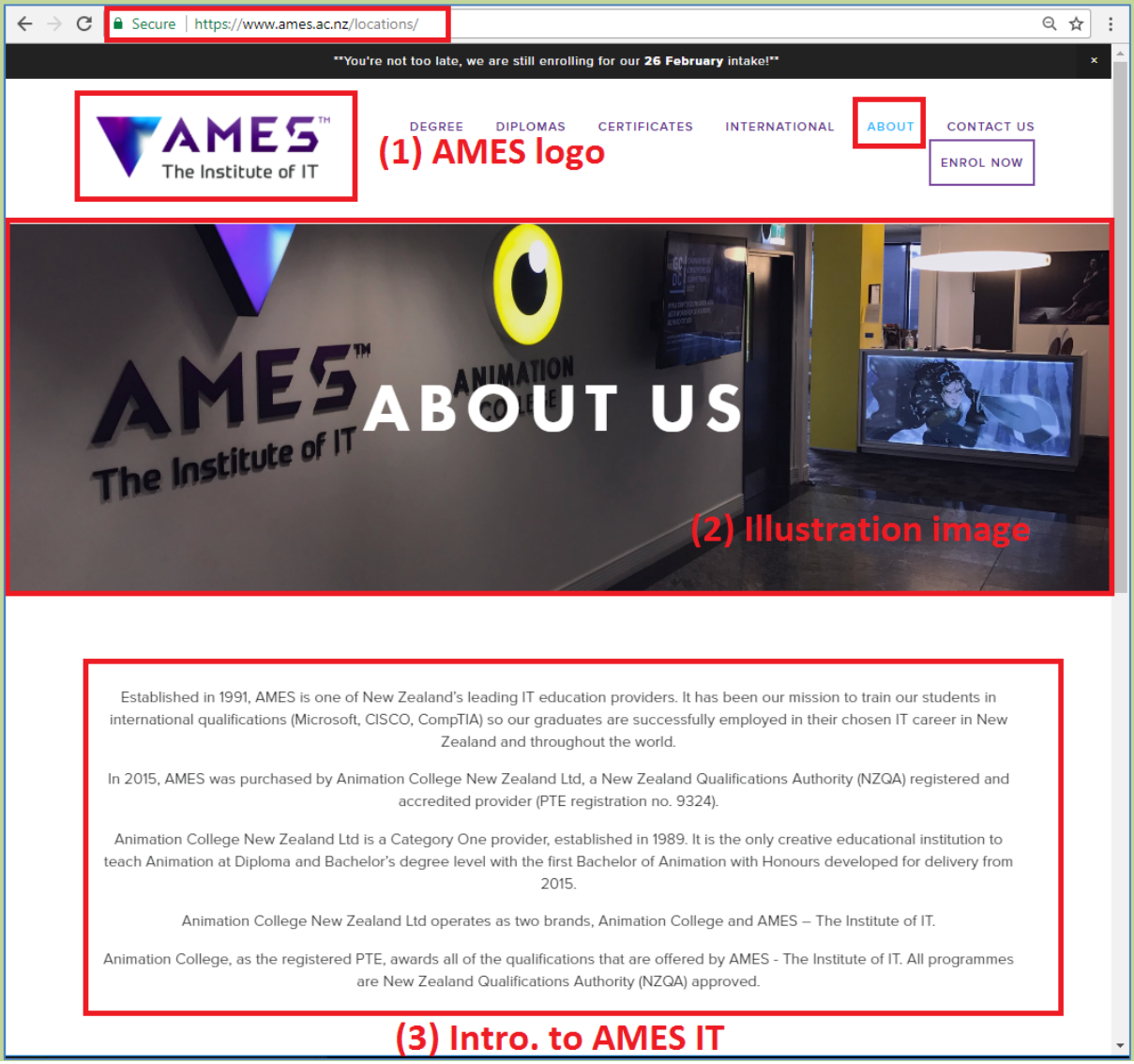
+ Open and have a look at AndroidManifest.xml file:

You will see that **the second Activity (.AboutAMES_Activity) has been auto added to AndroidManifest.xml file**:



Step 2: Analyze & design the about_ames_layout.xml

Open the link https://www.ames.ac.nz/locations/ and have a look at the content of "About AMES IT" page that we have to display:



In the "About AMES " layout, we will show 3 information of AMES IT:

1. **AMES logo:**

2. **Illustration image:**

3. **Intro. to AMES IT:**

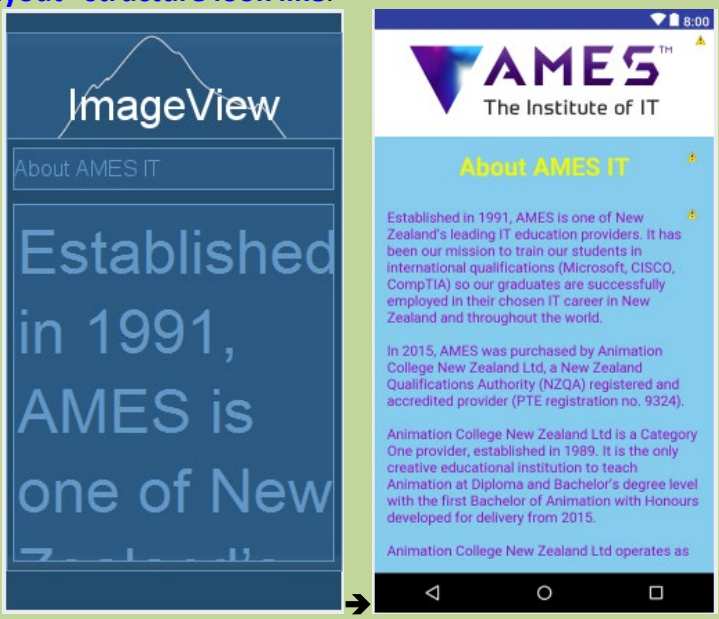
Established in 1991, AMES is one of New Zealand's leading IT education providers. It has been our mission to train our students in international qualifications (Microsoft, CISCO, CompTIA) so our graduates are successfully employed in their chosen IT career in New Zealand and throughout the world.

In 2015, AMES was purchased by Animation College New Zealand Ltd, a New Zealand Qualifications Authority (NZQA) registered and accredited provider (PTE registration no. 9324).

Animation College New Zealand Ltd is a Category One provider, established in 1989. It is the only creative educational institution to teach Animation at Diploma and Bachelor's degree level with the first Bachelor of Animation with Honours developed for delivery from 2015.

Animation College New Zealand Ltd operates as two brands, Animation College and AMES – The Institute of IT.
Animation College, as the registered PTE, awards all of the qualifications that are offered by AMES - The Institute of IT. All programmes are New Zealand Qualifications Authority (NZQA) approved"

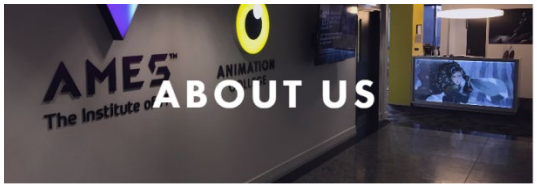
⇨ **The "About AMES layout" structure look like**:



Step 3: Prepare resource (images & texts) for the about_ames_layout.xml:

+ First of all, copy below images into the "drawable" folder:

ames_logo.jpg



illustration_image_ames.png

+ Then we declare 1 String variable ("about_ames") in strings.xml file:

Open strings.xml file in "values" folder and add 1 variables "about_ames":

```
<resources>
    <string name="app_name">Animation College Programmes</string>

    <!--Add "about_ames" variable-->
    <string name="about_ames">Established in 1991, AMES is one of New Zealand\'s leading IT education
providers. It has been our mission to train our students in international qualifications (Microsoft,
CISCO, CompTIA) so our graduates are successfully employed in their chosen IT career in New Zealand
and throughout the world.\n
\nIn 2015, AMES was purchased by Animation College New Zealand Ltd, a New Zealand Qualifications
Authority (NZQA) registered and accredited provider (PTE registration no. 9324).\n
\nAnimation College New Zealand Ltd is a Category One provider, established in 1989. It is the only
creative educational institution to teach Animation at Diploma and Bachelor\'s degree level with the
first Bachelor of Animation with Honours developed for delivery from 2015.\n
\nAnimation College New Zealand Ltd operates as two brands, Animation College and AMES - The
Institute of IT.\n
\nAnimation College, as the registered PTE, awards all of the qualifications that are offered by AMES
- The Institute of IT. All programmes are New Zealand Qualifications Authority (NZQA) approved.
    </string>

</resources>
```

Step 4: Design Graphic User Interface (GUI): about_ames_layout.xml

+ Now design the about_ames_layout.xml layout that contains:

- **1 ImageView: displays ames logo;**
- **1 TextView: displays title "About AMES IT";**
- **1 ImageView: display illustration image of ames;**
- **1 TextView: displays the detail of AMES IT;**

+ Open and edit the about_ames_layout.xml as below:

about_ames_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#85cced"
    android:orientation="vertical"
    tools:context=".AboutAMES_Activity">

    <!--ImageView: display ames logo-->
    <ImageView
        android:id="@+id/college_logo"
        android:layout_width="match_parent"
        android:layout_height="130dp"
        android:layout_gravity="center"
        android:background="#FFFFFFFF"
        android:contentDescription="college logo"
        android:gravity="center"
        android:scaleType="fitCenter"
        android:src="@drawable/ames_logo" />

    <!--TextView: display title-->
    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:gravity="center"
        android:padding="5dp"
        android:text="About AMES IT"
        android:textColor="#eff70c"
        android:textSize="30sp"
        android:textStyle="bold" />

    <!--ImageView: display illustration image-->
    <ImageView
        android:id="@+id/illustration_image"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_gravity="center"
        android:contentDescription="illustration image"
        android:gravity="center"
```

```
        android:scaleType="fitCenter"
        android:src="@drawable/illustration_image_ames" />

    <!--ScrollView: because about AMES information is over one page long, need ScrollView-->
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <!--TextView: display AMES IT detail-->
        <TextView
            android:id="@+id/aboutAMESIT"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:gravity="left"
            android:padding="5dp"
            android:scrollbars="vertical"
            android:text="@string/about_ames"
            android:textColor="#980bbf"
            android:textSize="17sp"
            android:textStyle="normal" />
    </ScrollView>
</LinearLayout>
```

Step 5: Use "Intent" object to jump from MainActivity to AboutAMES_Activity when users touch "About AMES" icon on mains screen:

In this part, we will add an image icon (**about_ames.png**) to **activity_main.xml layout**. When users touchs this icon, the app will open **AboutAMES_Activity** to display its associated **about_ames_layout.xml**.

+ Open activity_main.xml and add "about_ames_icon" icon below "title" TextView as below:

_First, copy the about_ames_icon.png image to "drawable" folder:

Image: about_ames_icon.png

_Then, add an ImageView to main layout (activity_main.xml):

```
<!--ImageView: "About AMES"-->
<ImageView
    android:id="@+id/aboutAMES"
    android:layout_width="110dp"
    android:layout_height="100dp"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/title"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:background="@drawable/about_ames_icon"
    android:contentDescription="ames icon" />
```

_Run your app on AVD (Nexus 5X) to see the result:

At the moment, if you touch "About AMES" icon, nothing happens. We will implement the function for "About AMES" icon touching in the next step in java code:

+ Open MainActivity.java file and add the below codes to implement the function:

when users touch "About AMES" icon, the app will move to AboutAMES_Activity to display the detailed information of AMES College:

Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:

_Declare a variable "aboutAmes" right after the class declaration:

```
//About AMES - 1: Declare "aboutAmes" variable as ImageView type
private ImageView aboutAmes;
```

_Inside onCreate() method, add code to find reference for "aboutAmes":

```
////////////////////////////////////////////////////////////////////////////////////////////////
//About AMES - 2: Find references and do casting for "aboutAmes"
aboutAmes = (ImageView) findViewById(R.id.aboutAMES);
```

_Inside onCreate() method, set click listener for "aboutAmes" imageView. Inside onCick() method of listener, create an Intent object to transition from this activity (MainActivity) to activity (AboutAMES_Activity):

```
////////////////////////////////////////////////////////////////////////////////////////////////
//About AMES - 3: Set click listener for the "aboutAmes" ImageView
aboutAmes.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Create an Intent object to transition from this activity (MainActivity) to
        // another activity (AboutAMES_Activity)
        Intent intent = new Intent(MainActivity.this, AboutAMES_Activity.class);
        startActivity(intent); //Start Activity
    }
});
```

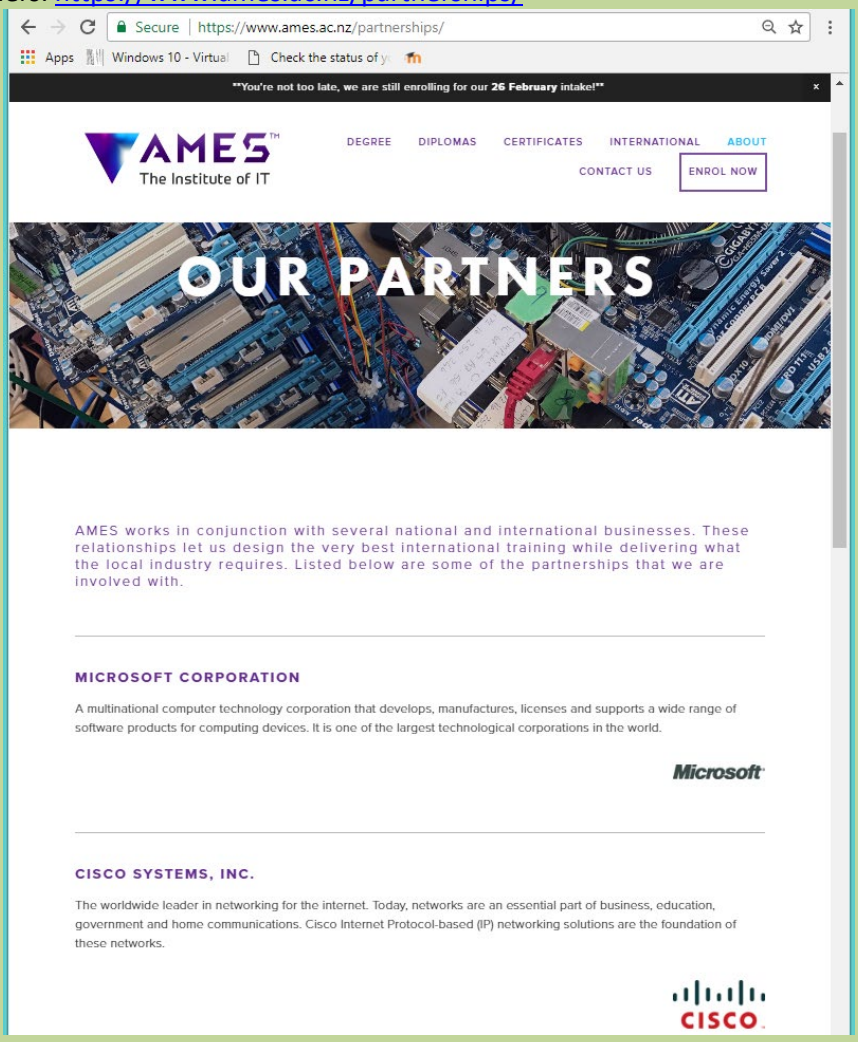
_Run the app on AVD (Nexus 5X) the see the result:

If users touch **"About AMES" icon**, the app will open **AboutAMES_Activity** to display its associated **about_ames_layout.xml**. When click **"back" button**, the app returns to main screen.

About AMES: Show a short description of AMES IT: **introduction** & **partners.**

However, so far the **"About AMES" Activity** only shows introduction of AMES but not "partners". It's your turn to add "partners" part into the app.

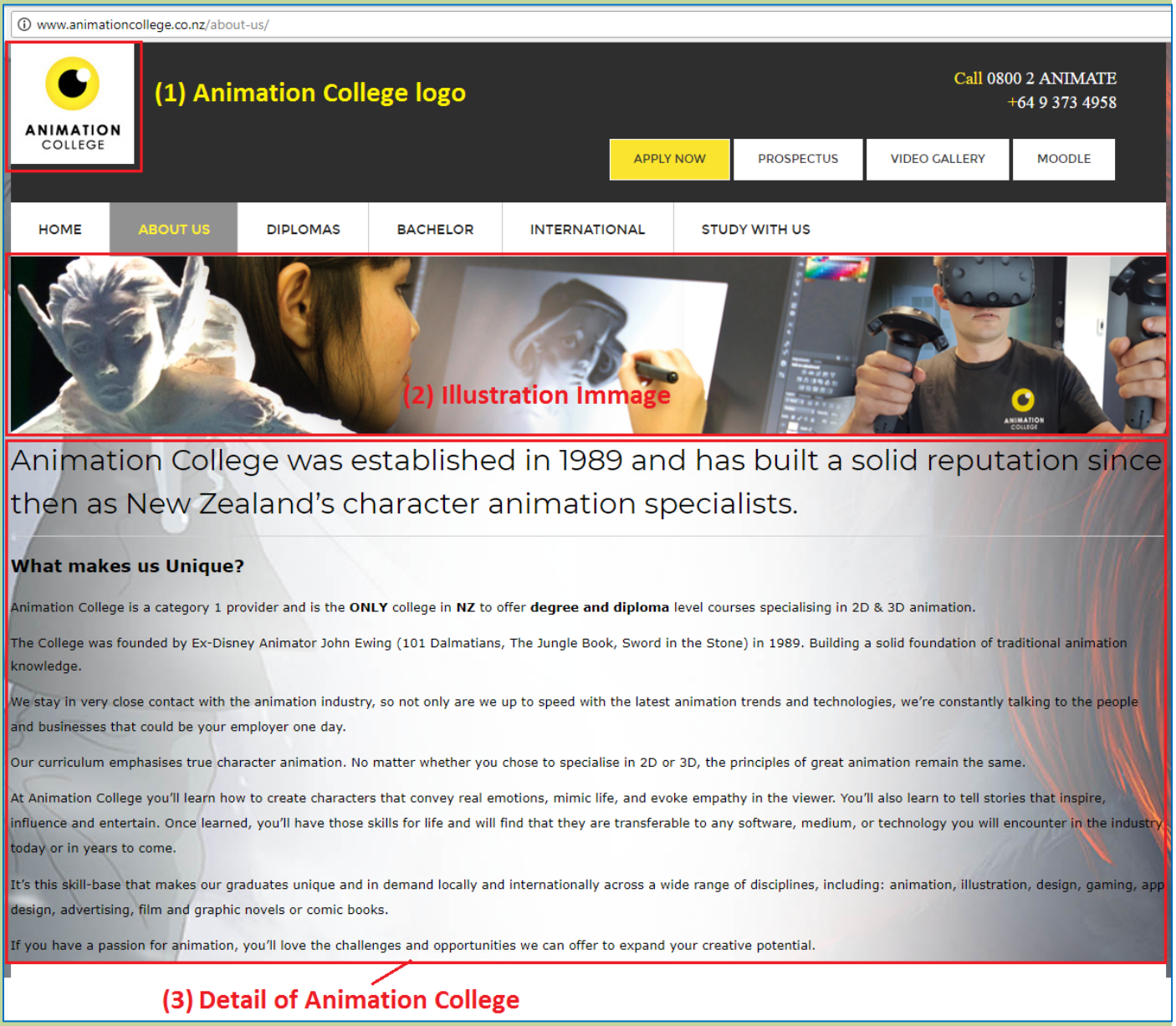
The link to AMES partners: https://www.ames.ac.nz/partnerships/



PART 3: Add the third Activity: "AboutAC_Activity"

It's your turn to add **the third Actiivty (AboutAC_Activity),** similarly to **AboutAMES_Activity**. The **AboutAC_Activity** will display the detail of Animation College as the content at the link: http://www.animationcollege.co.nz/about-us/

Open http://www.animationcollege.co.nz/about-us/ to have a look at "About Animation College":



Step 1: Add a new Activity to the project

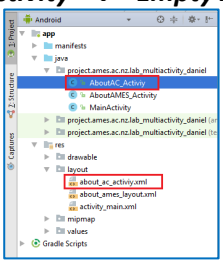
+ Add a new Activity, called "AboutAC_Activity" and its associated "about_ac_layout"

*_Right click on the folder containing **MainActivity** → "**New**" → "**Activity**" → "**Empty Activity**" → a "Configure Acitivy" window is appeared:*

*_Enter the "Activity Name": **AboutAC_Activity***

*_Enter the "Layout Name": **about_ac_layout***

*_Click "**Finish**"*



Step 2: Analyze & design the about_ac_layout.xml

In the "About AC" layout, we will show 3 information of AC:

1. **AC logo:**

2. **Illustration image:**

3. **Intro. to AC:**

Animation College was established in 1989 and has built a solid reputation since then as New Zealand's character animation specialists.

What makes us Unique?

Animation College is a category 1 provider and is the ONLY college in NZ to offer degree and diploma level courses specialising in 2D & 3D animation.

The College was founded by Ex-Disney Animator John Ewing (101 Dalmatians, The Jungle Book, Sword in the Stone) in 1989. Building a solid foundation of traditional animation knowledge.

We stay in very close contact with the animation industry, so not only are we up to speed with the latest animation trends and technologies, we're constantly talking to the people and businesses that could be your employer one day.

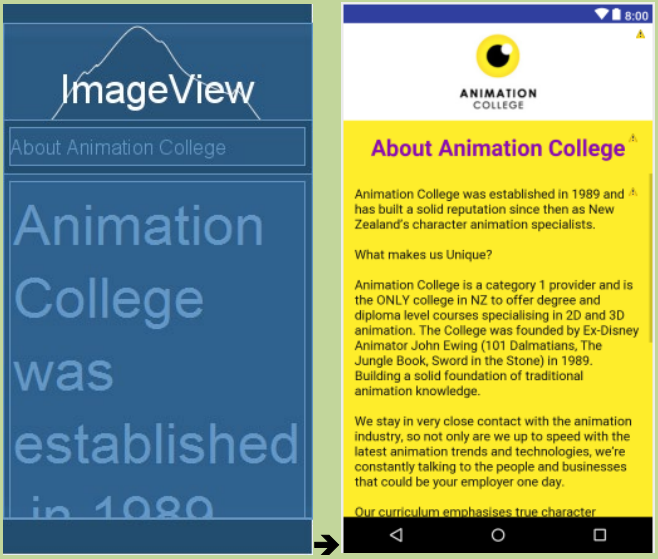
Our curriculum emphasises true character animation. No matter whether you chose to specialise in 2D or 3D, the principles of great animation remain the same.

At Animation College you'll learn how to create characters that convey real emotions, mimic life, and evoke empathy in the viewer. You'll also learn to tell stories that inspire, influence and entertain. Once learned, you'll have those skills for life and will find that they are transferable to any software, medium, or technology you will encounter in the industry today or in years to come.

It's this skill-base that makes our graduates unique and in demand locally and internationally across a wide range of disciplines, including: animation, illustration, design, gaming, app design, advertising, film and graphic novels or comic books.

If you have a passion for animation, you'll love the challenges and opportunities we can offer to expand your creative potential.

⇨ **The "About AC layout" structure look like**:



Step 3: Prepare resources (images & texts) for the about_ac_layout.xml:

+ First of all, copy below images into the "drawable" folder:

ac_logo.jpg

illlustration_image_ac.png

+ Then we declare 1 String variable ("about_ac") in strings.xml file:

Open strings.xml file in "values" folder and add 1 variables "about_ac":

```
<!--Add "about_ac" variable-->
    <string name="about_ac">Animation College was established in 1989 and has built a solid reputation
since then as New Zealand\'s character animation specialists. \n
\n<b>What makes us Unique?</b>\n
\nAnimation College is a category 1 provider and is the ONLY college in NZ to offer degree and diploma
level courses specialising in 2D and 3D animation.
The College was founded by Ex-Disney Animator John Ewing (101 Dalmatians, The Jungle Book, Sword in
the Stone) in 1989. Building a solid foundation of traditional animation knowledge.  \n
\nWe stay in very close contact with the animation industry, so not only are we up to speed with the
latest animation trends and technologies, we\'re constantly talking to the people and businesses that
could be your employer one day.\n
\nOur curriculum emphasises true character animation. No matter whether you chose to specialise in 2D
or 3D, the principles of great animation remain the same. \n
\nAt Animation College you\'ll learn how to create characters that convey real emotions, mimic life,
and evoke empathy in the viewer. You\'ll also learn to tell stories that inspire, influence and
entertain. Once learned, you\'ll have those skills for life and will find that they are transferable
to any software, medium, or technology you will encounter in the industry today or in years to come.\n
\nIt\'s this skill-base that makes our graduates unique and in demand locally and internationally
across a wide range of disciplines, including: animation, illustration, design, gaming, app design,
advertising, film and graphic novels or comic books.\n
\nIf you have a passion for animation, you\'ll love the challenges and opportunities we can offer to
expand your creative potential.
    </string>
```

Step 4: Design Graphic User Interface (GUI): about_ac_layout.xml

+ Now design the about_ac_layout.xml layout that contains:

- **1 ImageView: displays ac logo;**
- **1 TextView: displays title "About AC";**
- **1 ImageView: display illustration image of ac;**
- **1 TextView: displays AC introduction;**

+ Open and edit the about_ac_layout.xml as below:

about_ac_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FEED2B"
    android:orientation="vertical"
    tools:context=".AboutAC_Activiy">

    <!--ImageView: display AC logo-->
    <ImageView
        android:id="@+id/college_logo"
        android:layout_width="match_parent"
        android:layout_height="130dp"
        android:layout_gravity="center"
        android:background="#FFFFFFFF"
        android:contentDescription="ac logo"
        android:gravity="center"
        android:scaleType="fitCenter"
        android:src="@drawable/ac_logo" />

    <!--TextView: display title-->
    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:gravity="center"
        android:padding="5dp"
        android:text="About Animation College"
        android:textColor="#8a0eaf"
        android:textSize="30sp"
        android:textStyle="bold" />

    <!--ImageView: display illustration image-->
    <ImageView
        android:id="@+id/illustration_image"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_gravity="center"
        android:contentDescription="illustration image"
        android:gravity="center"
        android:scaleType="fitCenter"
        android:src="@drawable/illustration_image_ac" />

    <!--ScrollView: because about AC information is over one page long, need ScrollView-->
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <!--TextView: display AC detail-->
        <TextView
            android:id="@+id/aboutAMESIT"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:gravity="left"
            android:padding="5dp"
            android:scrollbars="vertical"
            android:text="@string/about_ac"
            android:textColor="#101010"
            android:textSize="17sp"
            android:textStyle="normal" />
    </ScrollView>
</LinearLayout>
```

Step 5: Use "Intent" object to jump from MainActivity to AboutAC_Activity when users touch "About AC" icon on mains screen:

In this part, we will add an image icon (**about_ac.png**) to **activity_main.xml layout**. When users touchs this icon, the app will open **AboutAC_Activity** to display its associated **about_ac_layout.xml**.

+ Open activity_main.xml and add "about_ac_icon" icon below "title" TextView as below:

_First, copy the about_ames_icon.png image to "drawable" folder:

Image: about_ac_icon.png

_Then, add an ImageView to main layout (activity_main.xml): at the bottom of current codes:

```
<!--ImageView: "About AC"-->
<ImageView
    android:id="@+id/aboutAC"
    android:layout_width="110dp"
    android:layout_height="100dp"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/title"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:background="@drawable/about_ac_icon"
    android:contentDescription="ac icon" />
```

+ Open MainActivit.java file and add the below codes to implement the function:

When users touch "About AC" icon, the app will move to AboutAC_Activity to display the detailed information of Animation College:

Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:

_Declare a variable "aboutAC" right after the class declaration:

```
//About AC - 1: Declare "aboutAC" variable as ImageView type
private ImageView aboutAC;
```

_Inside onCreate() method, add code to find reference for "aboutAC":

```
//////////////////////////////////////////////////////////////////////////////////////////
//About AC - 2: Find references and do casting for "aboutAC"
aboutAC = (ImageView) findViewById(R.id.aboutAC);
```

_Inside onCreate() method, set click listener for "aboutAC" imageView. Inside onCick() method of listener, create an Intent object to transition from this activity (MainActivity) to activity (AboutAC_Activity):

```
//////////////////////////////////////////////////////////////////////////////////////////
//About AC - 3: Set click listener for the "aboutAmes" ImageView
aboutAC.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Create an Intent object to transition from this activity (MainActivity) to
        // another activity (AboutAC_Activity)
        Intent intent = new Intent(MainActivity.this, AboutAC_Activiy.class);
        startActivity(intent);//Start Activity
    }
});
```

_Run the app on AVD (Nexus 5X) the see the result:

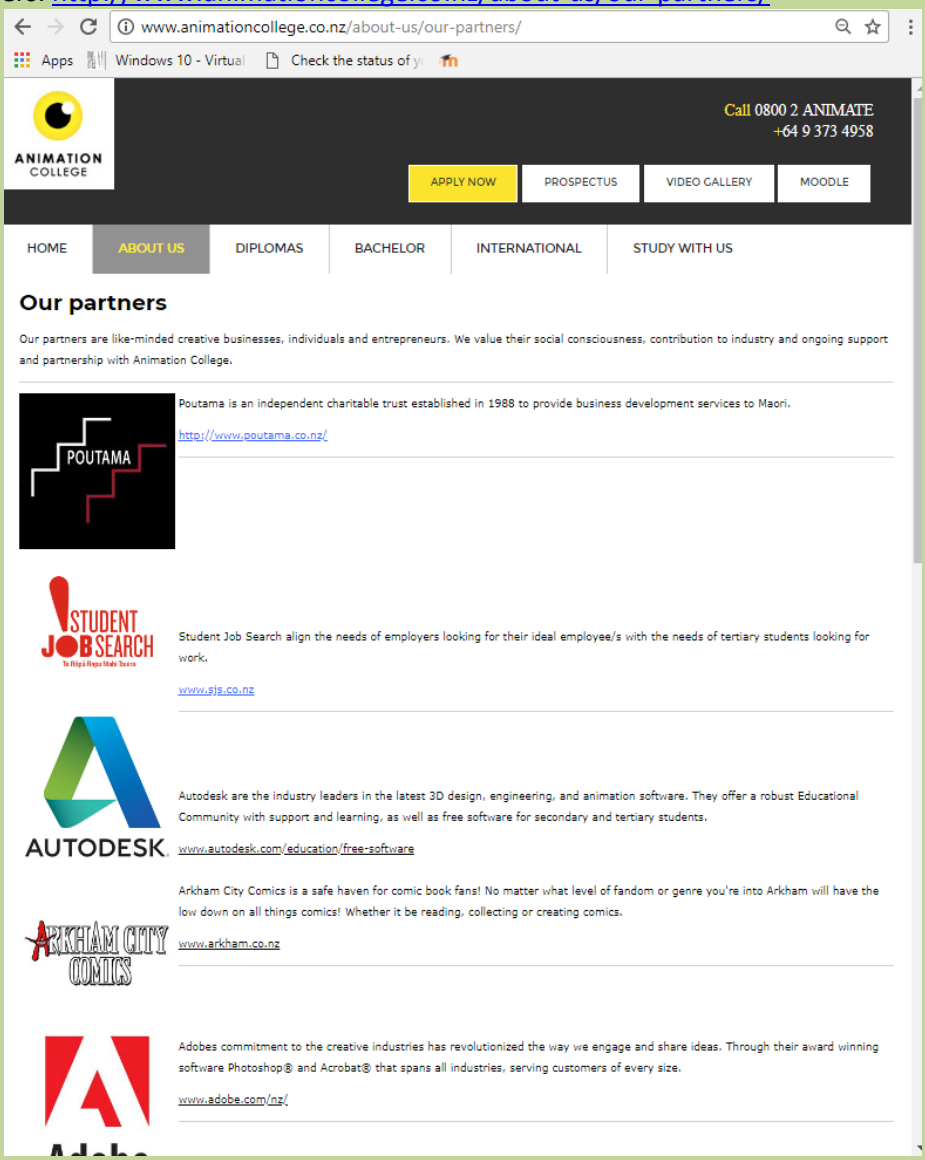
If users touch **"About AC" icon**, the app will open **AboutAC_Activity** to display its associated **about_ac_layout.xml**. When click **"back" button**, the app returns to main screen.

Self-directed learning

About AC: Show a short description of AC: **introduction** & **partners.**

However, so far the **"About AC" Activity** only shows introduction of AC but not "partners". It's your turn to add "partners" part into the app.

The link to AC partners: http://www.animationcollege.co.nz/about-us/our-partners/

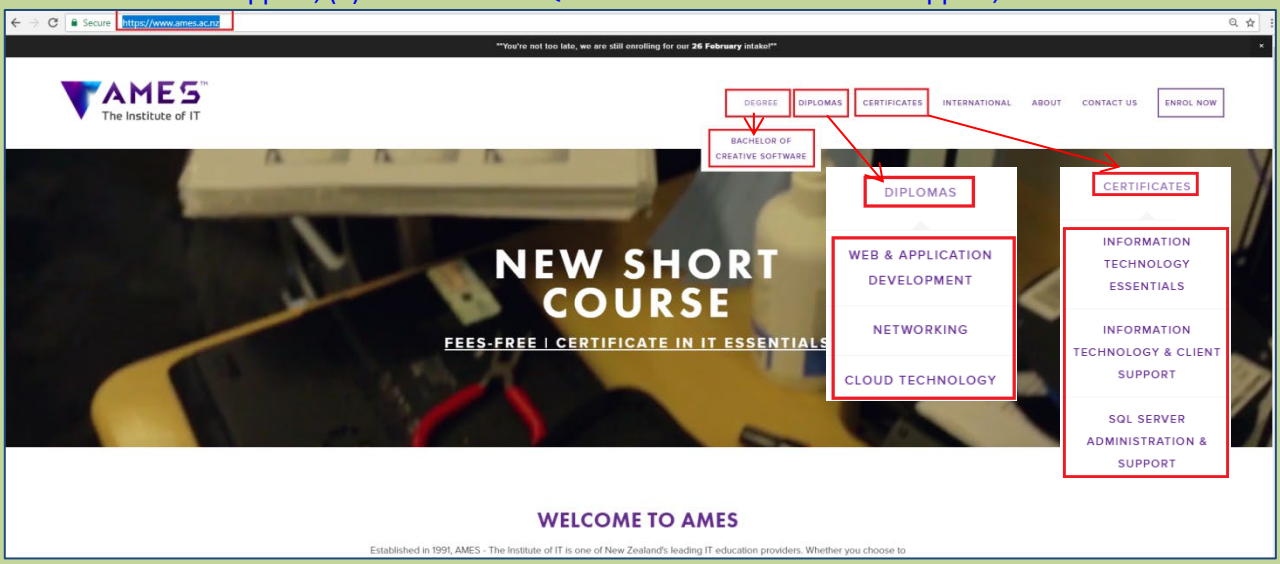


PART 4: Add the fourth Activity:

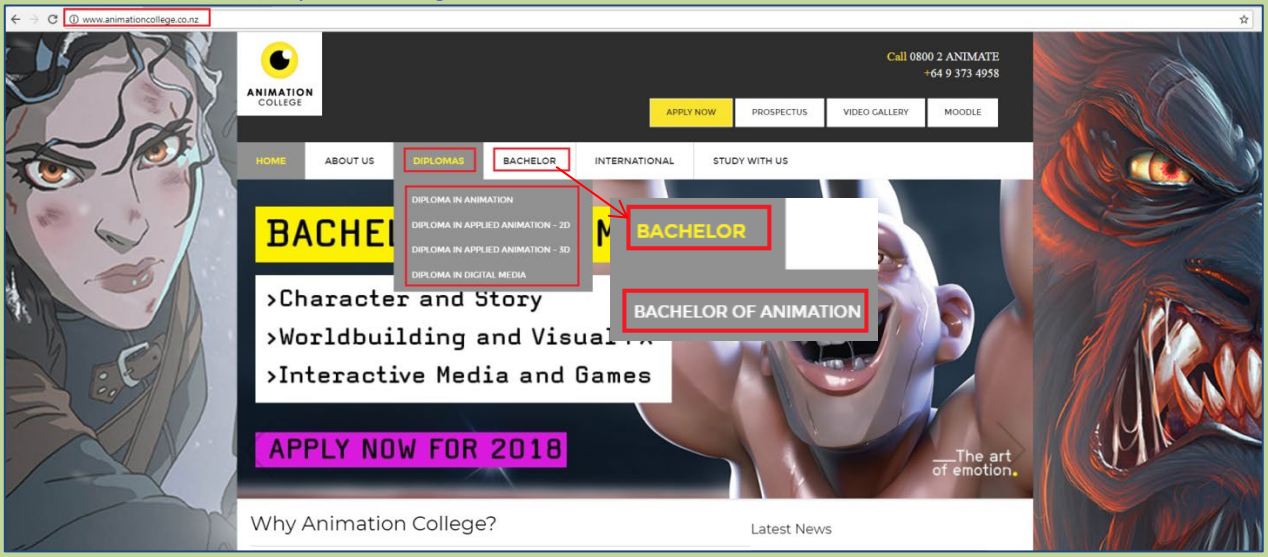
"ListViewProgrammes_Activity"

The "**view all programmes**" feature will display all 7 AMES programmes and 5 Animation programmes offered by both Animation College and AMES. Whenever **a specific programme is clicked**, the app displays its relevant information (***programme description & module, NZQF level, duration, tuition fee, start dates, and career opportunities***);

Here is the link to **7 AMES programmes**: https://www.ames.ac.nz/
(1) Degree: Bachelor of Creative Software; (2) Diploma in Application and Application Development; (3) Diploma in Networking; (4) Diploma in Cloud Technology; (5) Certificate in Information Technology Essentials; (6) Certificate in Information and Client Support; (7) Certificate in SQL Server Administration and Support;



Here is the link to **5 AC programmes**: http://www.animationcollege.co.nz/
(1) Degree - Bachelor of Animation; (2) Diploma in Animation; (3) Diploma in Applied Animation – 2D; (4) Diploma in Applied Animation – 3D; (5) Diploma in Digital Media;



Step 1: Collect and prepare all resources (images & texts) for this feature:

+ Collect titles of 12 programmes from websites: https://www.ames.ac.nz/ & http://www.animationcollege.co.nz/:

All 12 programmes titles

(1) AMES Degree: Bachelor of Creative Software
(2) AMES Diploma in Web and Application Development
(3) AMES Diploma in Networking
(4) AMES Diploma in Cloud Technology
(5) AMES Certificate in Information Technology Essentials
(6) AMES Certificate in Information and Client Support
(7) AMES Certificate in SQL Server Administration and Support
(8) AC Degree: Bachelor of Animation
(9) AC Diploma in Animation
(10) AC Diploma in Applied Animation – 2D
(11) AC Diploma in Applied Animation – 3D
(12) AC Diploma in Digital Media

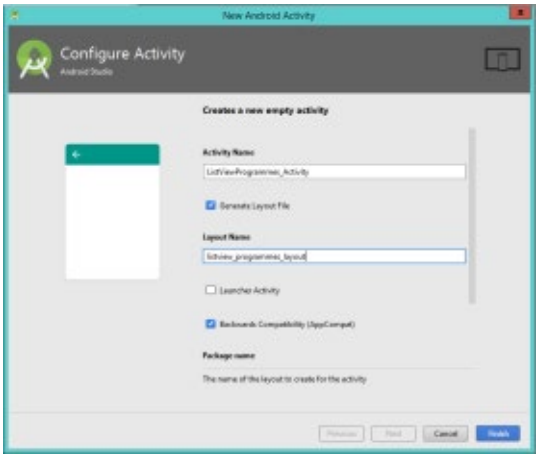
+ Then we declare 1 String variable ("all_programmes_array") in strings.xml file:
Open strings.xml file in "values" folder and add 1 variables "all_programmes_array":

```
<!--Add "all_programmes_array" array variable that contains 12 programmes of AC and AMES-->
<string-array name="all_programmes_array">
    <item>AMES Degree: Bachelor of Creative Software</item>
    <item>AMES Diploma in Web and Application Development</item>
    <item>AMES Diploma in Networking</item>
    <item>AMES Diploma in Cloud Technology</item>
    <item>AMES Certificate in Information Technology Essentials</item>
    <item>AMES Certificate in Information and Client Support</item>
    <item>AMES Certificate in SQL Server Administration and Support</item>
    <item>AC Degree:  Bachelor of Animation</item>
    <item>AC Diploma in Animation</item>
    <item>AC Diploma in Applied Animation - 2D</item>
    <item>AC Diploma in Applied Animation - 3D</item>
    <item>AC Diploma in Digital Media</item>
</string-array>
```

Step 2: Add a new Activity to the project

+ Add a new Activity, called "ListViewProgrammes_Activity" and its associated "listview_programmes_layout"

*_Right click on the folder containing **MainActivity** → "**New**" → "**Activity**" → "**Empty Activity**" → a "Configure Acitivy" window is appeared:*
*_Enter the "Activity Name": **ListViewProgrammes_Activity***
*_Enter the "Layout Name": **listview_programmes_layout***
*_Click "**Finish**"*



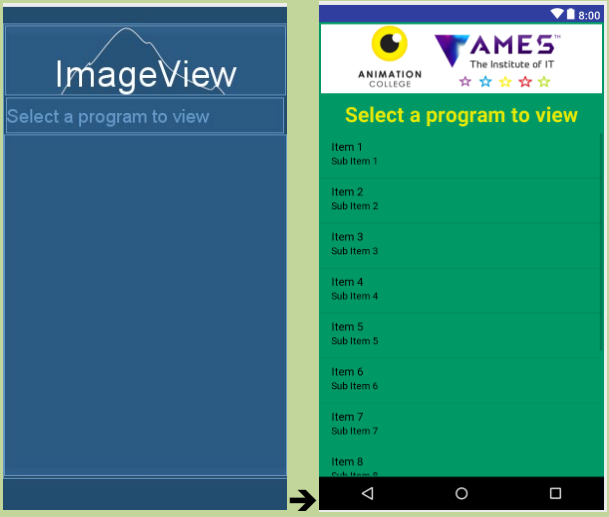
Step 3: Analyze & design the listview_programmes_layout

The listview_programmes_layout simply show a vertical list of 12 programme "titles" and when users click one programme in the list, the app will open a screen displaying all detailed information (extracted from all_programs_detail.xml file in "xml" folder) of that programmes.

+ Analyze the listview_programmes_layout.xml layout that contains:

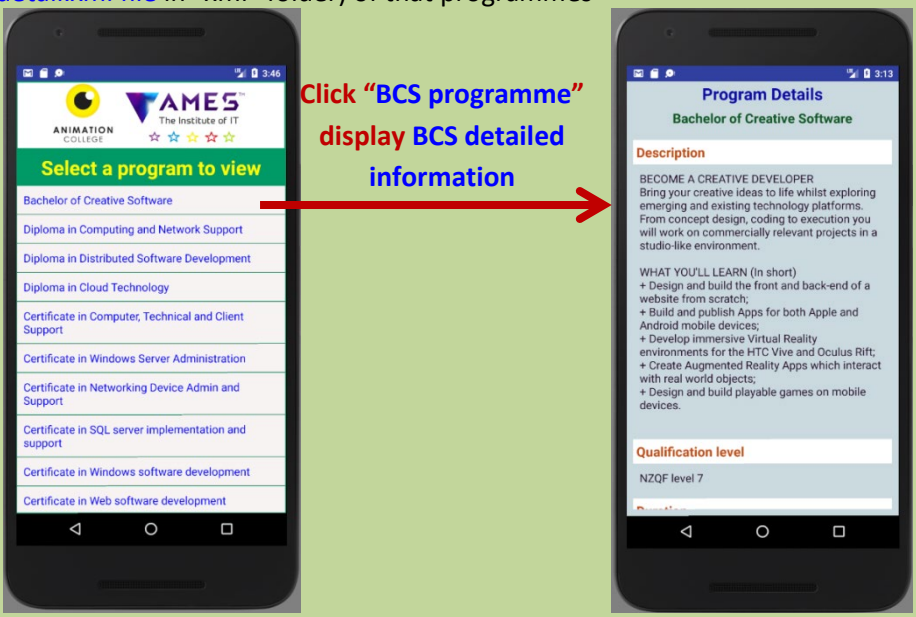
- 1 ImageView: display our college logo;
- 1 TextView: display title "Select a programme to view";
- 1 ListView: display all 12 programmes titles in vertical order;

⇨ **The "ListView Programmes layout" structure look like**:



+ Analyze the function of "view all programmes":

When users click one programme in the list, the app will open a screen displaying all detailed information (extracted from all_programs_detail.xml file in "xml" folder) of that programmes

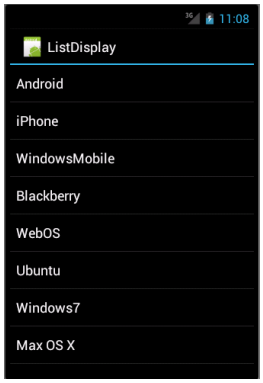


In this part, you will learn how to **implement a ListView to display dynamically the list of programs offered by our college.** When user clicks one program, a new screen appears and shows detailed information of that program.

What is ListView in Android?

Source: https://www.tutorialspoint.com/android/android_list_view.htm

Android **ListView** is a view which groups several items and display them in vertical scrollable list. The list items are automatically inserted to the list using an **Adapter** that pulls content from a source such as an array or database.



LIST VIEW

ListView Attributes

Following are the important attributes specific to GridView –

Attribute	Description
android:id	This is the ID which uniquely identifies the layout.
android:divider	This is drawable or color to draw between list items. .
android:dividerHeight	This specifies height of the divider. This could be in px, dp, sp, in, or mm.
android:entries	Specifies the reference to an array resource that will populate the ListView.
android:footerDividersEnabled	When set to false, the ListView will not draw the divider before each footer view. The default value is true.
android:headerDividersEnabled	When set to false, the ListView will not draw the divider after each header view. The default value is true.

An adapter actually bridges between UI components and the data source that fill data into UI Component. The Adapter holds the data and sends the data to adapter view, the view can take the data from adapter view and shows the data on different views like as spinner, list view, grid view etc.
The **ListView** is subclasses of **AdapterView** and they can be populated by binding them to an **Adapter**, which retrieves data from an external source and creates a View that represents each data entry.

ArrayAdapter

You can use this adapter when your data source is an array. By default, ArrayAdapter creates a view for each array item by calling toString() on each item and placing the contents in a **TextView**. Consider you have an array of strings you want to display in a ListView, initialize a new **ArrayAdapter** using a constructor to specify the layout for each string and the string array:

```
ArrayAdapter adapter = new ArrayAdapter<String>(this,R.layout.ListView,StringArray);
```

Here are arguments for this constructor:

- First argument **this** is the application context. Most of the case, keep it **this**.
- Second argument will be layout defined in XML file and having **TextView** for each string in the array.
- Final argument is an array of strings which will be populated in the text view.

Once you have array adapter created, then simply call **setAdapter()** on your **ListView** object as follows:

```
ListView listView = (ListView) findViewById(R.id.listview);
listView.setAdapter(adapter);
```

You will define your list view under res/layout directory in an XML file.

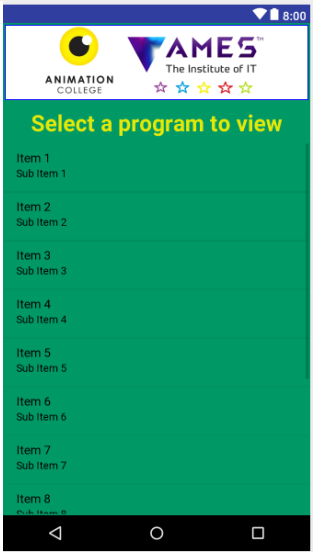
Read more:
[1]. List View: https://developer.android.com/guide/topics/ui/layout/listview.html
[2]. Android List View: https://www.tutorialspoint.com/android/android_list_view.htm

+ Now design the listview_programmes_layout layout that contains:

- 1 ImageView: display our college logo;
- 1 TextView: display title "Select a programme to view";
- 1 ListView: display all 12 programmes titles in vertical order;

Open and edit the listview_programmes_layout and edit it as below:
listview_programmes_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#FF009966"
    android:orientation="vertical"
    android:padding="2dp"
    tools:context=".ListViewProgrammes_Activity">

    <!--ImageView: display the college logo-->
    <ImageView
        android:id="@+id/collegelogo"
        android:layout_width="match_parent"
        android:layout_height="100dp"
        android:layout_centerHorizontal="true"
        android:layout_margin="1dp"
        android:background="#FFFFFFFF"
        android:contentDescription="college logo"
        android:foregroundGravity="center_horizontal"
        android:padding="2dp"
        android:scaleType="fitCenter"
        android:src="@drawable/collegelogo_transparent" />

    <!--TextView: display title "Select a program to view"-->
    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="3dp"
        android:gravity="center"
        android:padding="5dp"
        android:text="Select a program to view"
        android:textColor="#e5e907"
        android:textSize="30sp"
        android:textStyle="bold" />

    <!--ListView: display all 12 offered programmes-->
    <ListView
        android:id="@+id/listview_menu"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center_vertical|center_horizontal" />
</LinearLayout>
```



Step 4: Use "Intent" object to jump from MainActivity to ListViewProgrammes_Activity when users touch "Programmes icon" on mains screen:

In this part, we will add an image icon (**programmes_icon.png**) to **activity_main.xml layout**. When users touchs this icon, the app will open **ListViewProgrammes_Activity** to display its associated **listview_programmes_layout.xml**.

+ Open activity_main.xml and add "programmes_icon" icon below "about_ames" and "about_ac" ImageViews and located in the screen centre:

_First, copy the programmes_icon.png image to "drawable" folder:

Image: programmes_icon.png

_Then, add an ImageView to main layout (activity_main.xml):

```
<!--ImageView: display all programmes offered by AC-->
<ImageView
    android:id="@+id/viewAllProgrammes"
    android:layout_width="220dp"
    android:layout_height="200dp"
    android:layout_below="@+id/aboutAC"
    android:layout_centerInParent="true"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:background="@drawable/programmes_icon"
    android:contentDescription="programme icon" />
```

+ Open MainActivity.java file and add the below codes to implement the function:

When users touch "programmes" icon, the app will move to ListViewProgrammes_Activity to display the detailed information of selected programme.

Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:

_Declare a variable "viewAllProgrammes" right after the class declaration:

```
//View All Programmes - 1: Declare "viewAllProgrammes" variable as ImageView type
private ImageView viewAllProgrammes;
```

_Inside onCreate() method, add code to find reference for "viewAllProgrammes":

```
////////////////////////////////////////////////////////////////////////////////////////
//View All Programmes - 2: Find references and do casting for "viewAllProgrammes"
viewAllProgrammes = (ImageView) findViewById(R.id.viewAllProgrammes);
```

_Inside onCreate() method, set click listener for "viewAllProgrammes" imageView. Inside onCick() method of listener, create an Intent object to transition from this activity (MainActivity) to activity (ListViewProgrammes_Activity):

```
////////////////////////////////////////////////////////////////////////////////////////
//View All Programmes - 3: Set click listener for the "viewAllProgrammes" ImageView
viewAllProgrammes.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Create an Intent object to transition from this activity (MainActivity) to
        // another activity (ListViewProgrammes_Activity)
        Intent intent = new Intent(MainActivity.this, ListViewProgrammes_Activity.class);
        startActivity(intent);//Start Activity
    }
});
```

+ Run the app on AVD (Nexus 5X) the see the result:

If users touch **"view all programmes" icon**, the app will open **ListViewProgrammes _Activity** to display its associated **about_ac_layout.xml**. So far the list view of all programmes is empty. We'll populate this list view in java code later.

Step 5: Add java code to ListViewProgrammes_Activity so that the ListView will be populated with all program titles:

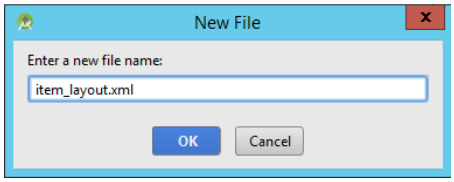
+ Add an item_layout.xml file to display items in ListView:

Add a **new xml file to "layout" folder** and name it as **item_layout.xml**:

_Right click on ***"layout" folder -> New -> File***

_In the window "***New File***", enter the new file name: ***item_layout.xml***

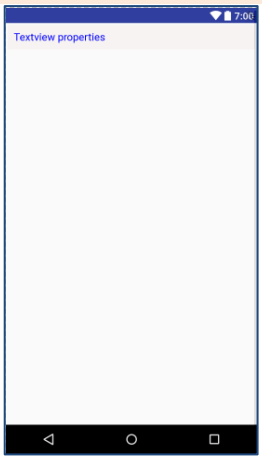
_Click "***OK***"



+ Edit the item_layout.xml as below to define the layout for each item of the list view, the TextView actually is used to format the items in the listview:

```
<?xml version="1.0" encoding="utf-8"?>
<!--Define the TextView properties for the items in the ListView-->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_margin="3dp"
    android:background="#f7f3f3"
    android:gravity="left"
    android:padding="10dp"
    android:text="Textview properties"
    android:textColor="#FF0000FF"
    android:textSize="17sp"
    android:textStyle="normal">
</TextView>
```

```
<?xml version="1.0" encoding="utf-8"?>

<!--Define the TextView properties for the items in the menu list-->
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:layout_margin="3dp"
    android:background="#f7f3f3"
    android:gravity="left"
    android:padding="10dp"
    android:text="Textview properties"
    android:textColor="#FF0000FF"
    android:textSize="17sp"
    android:textStyle="normal">

</TextView>
```



+ Now, open ListViewProgrammes_Activity.java, keep the auto-generated codes and add below codes:

1. Declare 2 variables: **listView** and **programs_array**;
2. Make the link between layout and java code: **find references and do casting for 2 above variables**
3. Retrieve the **all_programmes_array** declared in **strings.xml** by using **getResource() method**;
4. Declare an **ArrayAdapter** to create the list with "**programs_array"** being laid out;
5. **Make the ListView actionable** by declaring an **onClick() listener** for each of them;

+ Here is the full code of the ListView_Activity.java file:

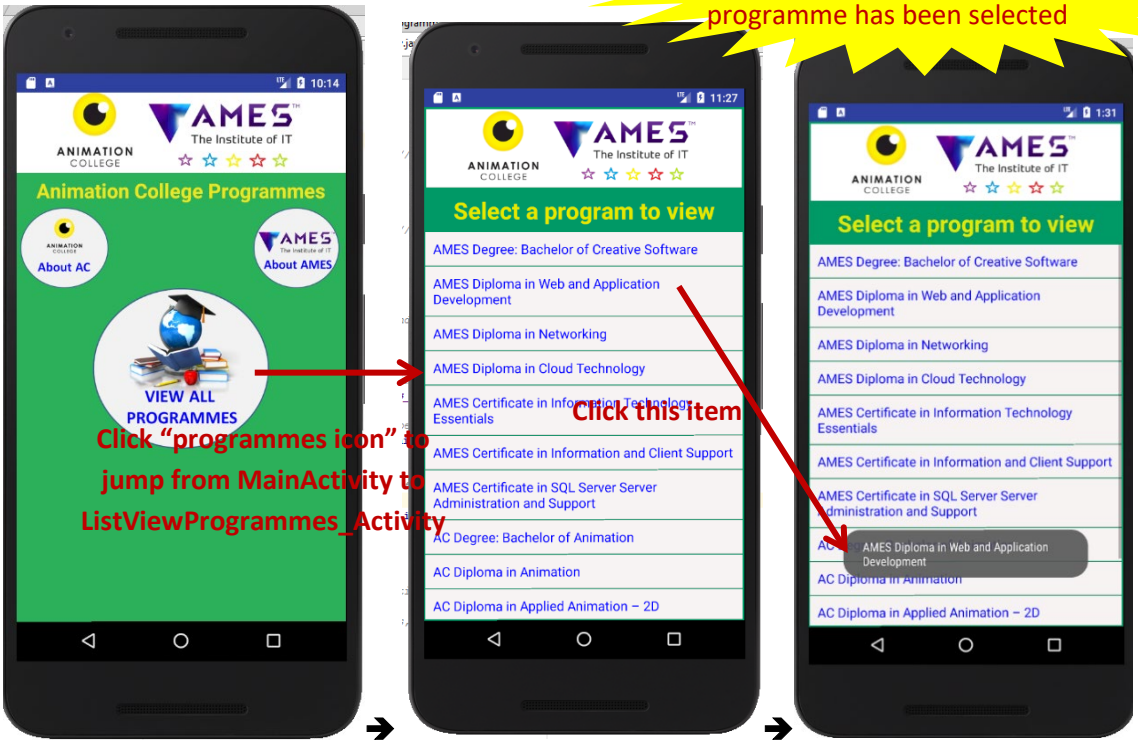
ListViewProgrammes_Activity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
```

```
import android.widget.ListView;
import android.widget.TextView;

public class ListViewProgrammes_Activity extends AppCompatActivity {
    ////////////////////////////////////////////////////////////////////////////////////////////
    //1: Declare variables
    private ListView listView;
    private String[] programs_array;

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

        //2: Find references and do casting to make the link between front-end and back-end
        listView = (ListView) findViewById(R.id.listview_menu);

        //3: Retrieve the all_programmes_array declared in strings.xml
        //Use getResource() function to access to String-array stored in strings.xml
        programs_array = getResources().getStringArray(R.array.all_programmes_array);

        //4: Declare an ArrayAdapter to create the list with programs_array being laid out
        ArrayAdapter<String> arrayAdapt = new ArrayAdapter<String>(this,
                                  R.layout.item_layout,
                                  programs_array);
        listView.setAdapter(arrayAdapt);

        //5: Make the ListView actionable by declaring an onClick() listener for each of them.
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent, View itemClicked, int position, long id) {
                //Read out the name of clicked item
                TextView tvItemClicked = (TextView) itemClicked;
                String strItemClicked = tvItemClicked.getText().toString();
                //Pop up a Toast to display the selected programme in the ListView
                Toast.makeText(getApplicationContext(), strItemClicked, Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

+ Run your app to see the result:

Now **ListView** is populated with all programme titles. When you click one programme in the ListView, only a Toast pop up showing which programme has been selected



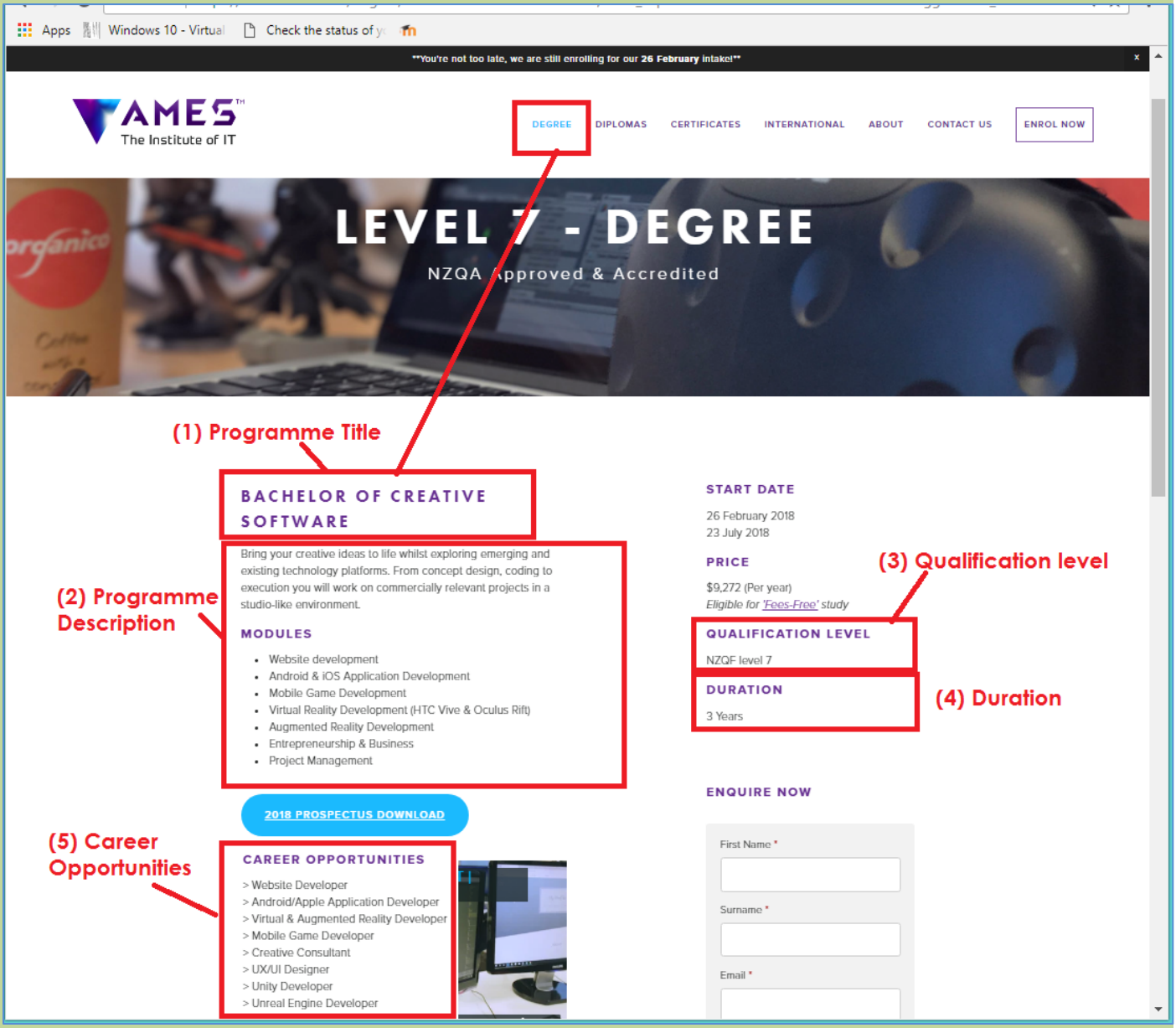
PART 5: Add the fifth Activity:

"ProgrammeDetail_Activity"

Step 1: Collect and prepare all resources (images & texts) for this feature:

+ Collect details (programme description & modules, NZQF level, duration, and career opportunities) for all 12 above programmes from websites: https://www.ames.ac.nz/ & http://www.animationcollege.co.nz/ and put them in a xml file (all_programs_detail.xml).

Below is an example of Bachelor of Creative Software programme on AMES IT website:



+ You can use Notepad/Nodepad++ Editor or any other Text Editor to create all_programms_detail.xml.

all_programs_detail.xml:

```
<?xml version="1.0"?>
<all_programs>

<program>
<title>AMES Degree: Bachelor of Creative Software</title>

<description>
Bring your creative ideas to life whilst exploring emerging and existing technology such as
Virtual and Augmented Reality, Mobile Games, Website Development and iOS/Android Application
Development. From concept design, coding to execution you will work on commercially relevant
projects in a studio-like environment.

MODULES
 + Website development
 + Android and iOS Application Development
 + Mobile Game Development
 + Virtual Reality Development (HTC Vive and Oculus Rift)
 + Augmented Reality Development
 + Entrepreneurship and Business
 + Project Management
</description>

<qualification_level>NZQF level 7</qualification_level>

<duration>3 Years</duration>

<career>
 > Website Developer
 > Android/Apple Application Developer
 > Virtual and Augmented Reality Developer
 > Mobile Game Developer
 > Creative Consultant
 > UX/UI Designer
 > Unity Developer
 > Unreal Engine Developer
</career>
</program>
```

<program>
<title>AMES Diploma in Web and Application Development</title>

<description>
This specialisation teaches you how to use SQL, HTML5, CSS3 and Javascript to develop interactive and responsive web sites. Using the C# programming language, learners will also develop mobile applications for the Windows Store, whilst integrating Microsoft's cloud services to the applications developed.

MODULES
 + Mobile Application Development - 30 Credits
 + Web Application Development - 30 Credits
 + Cloud Services - 30 Credits
 + Practical Projects - 30 Credits
</description>

<qualification_level>NZQF level 5</qualification_level>

<duration>32 weeks</duration>

<career>
 > Web Developer
 > Mobile Application Developer
 > Cloud Developer
 > Database Developer
</career>
</program>

<program>
<title>AMES Diploma in Networking</title>

<description>
In this specialisation you'll learn networks architecture, structure, functions, components and models, before progressing to routing and switching essentials for small networks, then on to scaling networks for more complex enrolments.

MODULES
 + Administering, Installing and Configuring Servers - 45 Credits
 + Network Theory and Practice - 30 Credits
 + Advanced Networking - 30 Credits
 + Specialisation Project - 15 Credits
</description>

<qualification_level>NZQF level 6</qualification_level>

<duration>32 weeks</duration>

<career>
 > Network Technician
 > IT Support Administrator
 > Service Technician
 > Technical Support
 > Help-desk Representative
</career>
</program>

<program>
<title>AMES Diploma in Cloud Technology</title>

<description>
Learn to solve complex problems using the latest in Cloud technology. You'll learn enterprise networks, network infrastructure, messaging solutions and information sharing systems using Cloud-based solutions that are scalable, reliable and on-demand.

MODULES
 + Active directory services
 + Configuring advanced server services
 + Server administration
 + Information sharing systems
 + Platform as a service
 + Administering System Centre Configuration Manager
 + Deploying System Centre Configuration Manager
 + Monitoring and operating a private Cloud
 + Configuring and deploying a private Cloud
</description>

<qualification_level>NZQF level 7</qualification_level>

<duration>45 weeks</duration>

<career>
 > Server Administrator
 > Network Administrator
 > Database Administrator
 > Network Analyst
 > Technical Support
 > Systems Specialist
 > Virtualisation Specialist
 > Cloud Solution Expert
</career>
</program>

<program>
<title>AMES Certificate in Information Technology Essentials</title>

<description>
If you've been looking for a foundation course to dip your toes into computer hardware or software development then our low-fees NZQA approved course is for you.This specialisation aims to equip students with a range of fundamental level IT skills. You'll cover computer hardware, software concepts, user interface design and business analysis in a hands-on project based learning environment. Learnings from this programme may be used to enter the workforce or to continue onto higher level IT Study.

MODULES
 + Computing Essentials
 + Business Essentials for IT
 + Software Essentials
</description>

<qualification_level>NZQF level 4</qualification_level>

<duration>16 weeks</duration>

<career>
 > Web Content Manager
 > Junior Web Designer
 > IT Office Specialist
</career>
</program>

<program>
<title>AMES Certificate in Information and Client Support</title>

<description>
This specialisation covers the fundamentals of computer technology, installation, configuration and troubleshooting of PCs and related hardware, as well as basic networking, scripting and installing/configuring Operating Systems. Learners will have guidance from our team of experienced lecturers in a hands-on practical learning environment.

MODULES
 + Scripting for Systems Administrators - 15 Credits
 + Hardware - 15 Credits
 + Operating Systems - 15 Credits
 + Introduction to Networking - 15 Credits
</description>

<qualification_level>NZQF level 5</qualification_level>

<duration>16 weeks</duration>

<career>
 > Field Service Technician
 > IT Support Technician
 > Help-Desk Representative
</career>
</program>

<program>
<title>AMES Certificate in SQL Server Administration and Support</title>

<description>
This specialisation teaches you how to use Microsoft Structured Query Language (SQL) Server for relational database systems. In this programme you'll also learn backup/restore strategies and security control. Using practical exercises, you'll become familiar with developing database solutions with Microsoft's SQL Server.

MODULES
 + Scripting for Systems Administrators - 15 Credits
 + Administering Database Servers - 15 Credits
 + Developing Databases - 15 Credits
 + Structured Query Language - 15 Credits
</description>

<qualification_level>NZQF level 5</qualification_level>

<duration>16 weeks</duration>

<career>
 > Database Administrator
 > Database Developer
</career>
</program>

<program>
<title>AC Degree: Bachelor of Animation</title>

<description>
Become a Character Animator, World-Builder or Interactive Game Designer.

Want to work with the best, and be the best? Our Bachelor of Animation is the ultimate training ground. It's New Zealand's only degree level course specialising in 2D and 3D character animation, and it produces world-class animators who are at the very top of their field. If you want to reach your full potential as an animator, there's no better pathway.
Offered in Auckland or Wellington.
</description>

<qualification_level>NZQF level 7 (360 credits) or NZQF Level 8 (480 credits)
</qualification_level>

<duration>3 years</duration>
<career>
 + CHARACTER DESIGNERS / CHARACTER ANIMATORS
 + DIRECTORS OF ANIMATION
 + NARRATIVE DESIGNERS
 + SCREENPLAY WRITERS / SCRIPT WRITERS / STORYBOARD ARTISTS
 + PREVIS ARTISTS / LAYOUT ARTISTS
 + VIRTUAL CINEMATOGRAPHERS
 + ANIMATIC ARTISTS
 + EXPERIMENTAL ANIMATORS
 + EDITORS
 + CONCEPT ARTISTS
 + WORLD-BUILDERS
 + SCENOGRAPHERS
 + ART DIRECTORS
 + ENVIRONMENT DESIGNERS
 + MATTE PAINTERS
 + 3D ARTISTS
 + 3D MODELLERS
 + RIGGERS
 + LIGHTING ARTISTS
 + TEXTURE ARTISTS
 + TECHNICAL ARTISTS
 + MOTION CAPTURE TECHNICIANS
 + COMPOSITORS
 + VISUAL EFFECTS ARTISTS
 + VISUAL GRADERS ILLUSTRATORS
 + GAME DESIGNERS
 + LEVEL DESIGNERS
 + CONCEPT ARTISTS
 + INTERACTIVE NARRATIVE DESIGNERS
 + INTERFACE DESIGNERS
 + INDIE GAME DEVELOPERS
 + MOBILE CONTENT DEVELOPERS
 + APP DESIGNERS
 + DIGITAL MEDIA ARTISTS
 + GAME ARTISTS
 + CHARACTER DESIGNERS
 + SPRITE ANIMATORS
</career>
</program>

<program>
<title>AC Diploma in Animation</title>

<description>
Become a 2D or 3D character animator, concept artist or game developer.

Are you a fan of Hayao Miyazaki? Do you have a soft spot for Disney or Anime? If you're captivated by the hand-drawn characters of legends like Miyazaki and Walt Disney or a super-savvy techie into using software, not pencils. Well, choose your weapon, and become either a master of 2D Digital or 3D Computer Animation.

Kick start your animation training with an introduction to the theory and practice of animation principles with the Diploma in Animation. This one-year foundation programme gives you the option to explore the fundamentals of character animation, pre-production, specialist software and performance animation, with anatomical knowledge at its very core.
</description>

<qualification_level>NZQF level 5</qualification_level>

<duration> 1 year (32 weeks)</duration>

<career>
 > Animator
 > Concept artist
 > 3D guru
</career>
</program>

<program>
<title>AC Diploma in Applied Animation – 2D</title>

<description>
Become a character animator, concept artist or game developer.
Are you a fan of Hayao Miyazaki movies? Do you have a soft spot for Disney or Anime? If you're captivated by the hand-drawn characters of legends like Miyazaki and Walt Disney, 2D animation could be your ultimate calling.
Sure, a lot has changed since the early days. 2D characters can now be digitally animated and we can use Adobe Flash and After Effects to bring them to life. Today's 2D animators do everything from straight ahead digital animation, to puppetry animation, to hybrid forms of 2D animation.
But the essence of 2D animation remains. It's a traditional hand-drawn craft. It's natural and intuitive. It has drawing techniques and anatomical knowledge at its very core. Movements are broken down into main poses, called 'keys', which tell a story when put into a sequence. The sequence is then composited and backed up by dialogue and music.
For purists and passionate artists, there's nothing quite like it!
</description>

<qualification_level>NZQF level 6</qualification_level>

<duration>2 years (32 weeks per year)</duration>

<career>
 > 2D Animator
</career>
</program>

<program>
<title>AC Diploma in Applied Animation - 3D</title>

<description>
NZQA Accredited, level 6 programme.

3D animation (or computer-generated imagery) uses three-dimensional computer models to animate characters by combining several styles of animation; these include 2D techniques of keys and breaks, the stop-motion process of sculpting a character as a model, and the puppeteer control system of articulated points of movement. These are more commonly known as key-framing, modelling and rigging.

Other 3D techniques such as texturing and lighting allow 3D animators to skin and light their character or create backgrounds that simulate real life environments. Students adopt these methods using Autodesk Maya as their principle software, but also learn Adobe Photoshop.

The techniques learned can be applied in TV advertising, TV series production, visual fx for film, AAA gaming for PC, Xbox and Playstation, architecture modelling and 3D visualisation.
</description>

<qualification_level>NZQF level 6</qualification_level>

<duration>2 years (32 weeks per year)</duration>

<career>
 > 2D Animator
</career>
</program>

<program>
<title>AC Diploma in Digital Media</title>

<description>
NZQA Accredited, level 6 programme.

The Diploma of Digital Media (DDM) is an intensive one-year course of study designed to refine students' fundamental animation skills and develop knowledge gained in the previous two years. This final year exposes students to the entire production pipeline encountered in a real working studio environment. It is largely self-directed and students are expected to manage their time and workload efficiently. The pace is demanding, replicating the pressures and deadlines of working life in the industry. This prepares our students for a much easier transition into the working world. The main focus, as always, is on producing top quality character animation and performance.
</description>

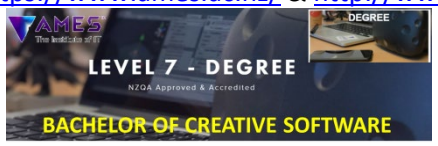
<qualification_level>NZQF level 7</qualification_level>

<duration>1 year (32 weeks)</duration>

<career>
 > 2D or 3D Animator
</career>
</program>

</all_programs>

+ Collect illustration images for 12 propgrammes from websites:
https://www.ames.ac.nz/ & http://www.animationcollege.co.nz/



(1) AMES Degree: Bachelor of Creative Software
Illustration_ image1.png

(2) AMES Diploma in Web and Application Development:
Illustration_ image2.png

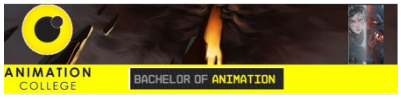
(3) AMES Diploma in Networking
Illustration_ image3.png

(4) AMES Diploma in Cloud Technology
Illustration_ image4.png

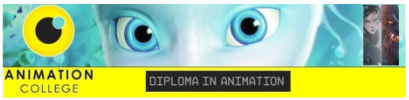
(5) AMES Certificate in Information Technology Essentials
Illustration_ image5.png

(6) AMES Certificate in Information and Client Support
Illustration_ image6.png

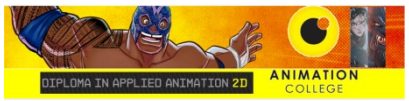
(7) AMES Certificate in SQL Server Server Administration and Support
Illustration_ image7.png



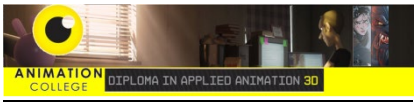
(8) AC Degree: Bachelor of Animation
Illustration_ image8.png



(9) AC Diploma in Animation
Illustration_image9.png



(10) AC Diploma in Applied Animation – 2D
Illustration_image10.png

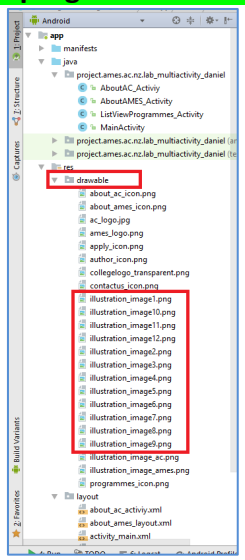


(11) AC Diploma in Applied Animation – 3D
Illustration_image11.png

(12) AC Diploma in Digital Media
Illustration_image12.png

Step 2: Transfer resources (images, texts, xml file) into Android Project

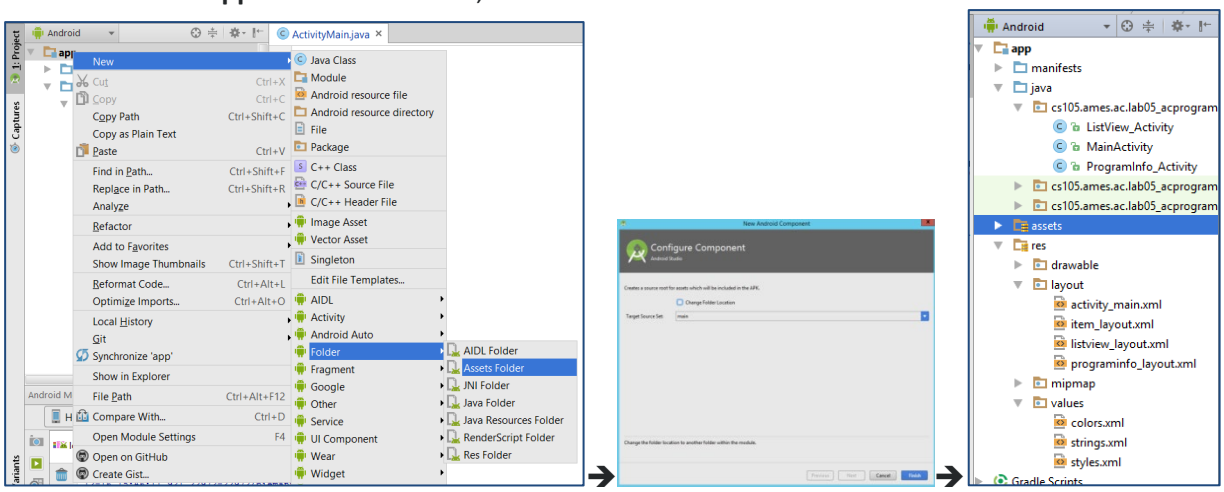
+ Copy 12 above illustration images for each programme into "drawable" folder:



+ Add "Assets" folder to the project:

Within the **"Android" tab** (see the drop-down in the topleft of my image):

- Right-click on the **"app" folder;**
- **Click: "New" → "Folder" → "Assets Folder"**;
- **A new window appears**: Click "**Finish**";

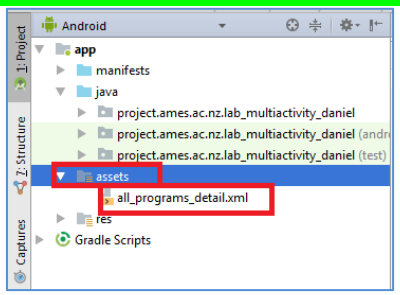


'Assets' folder will be distributed along with the APK, which contains all the raw files you need for application and Those files cannot be put into "**res" folder** as usual, such as:

- text files (.txt),
- non-Android XML files (.xml),
- Audio files (.wav, .mp3, .mid)…

Read more at: AssetManager from Android Developers' References

+ Copy the above all_programs_detail.xml file into the "Assets" folder:



Step 3: Add a new Activity to the project

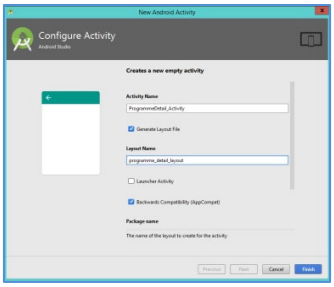
+ Add a new Activity, called "ProgrammeDetail_Activity" and its associated "programme_detail_layout"

*_Right click on the folder containing **MainActivity** → "**New**" → "**Activity**" → "**Empty Activity**" → a "Configure Acitivy" window is appeared:*

*_Enter the "Activity Name": **ProgrammeDetail_Activity***

*_Enter the "Layout Name": **programme_detail_layout***

*_Click "**Finish**"*



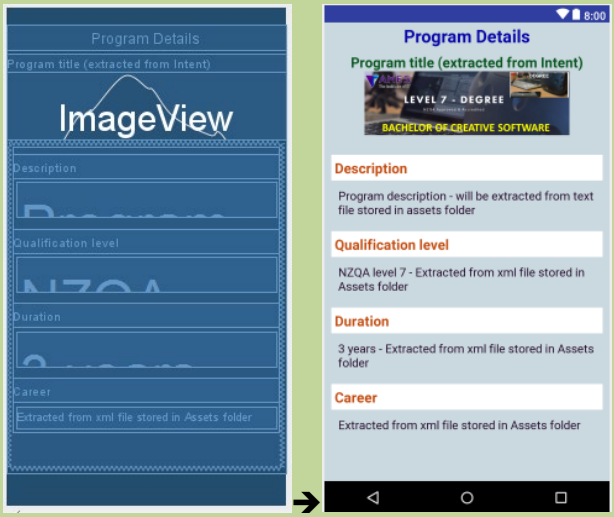
Step 4: Analyze & design the programme_detail_layout

When users click one programme in the ListView, the app will open a screen displaying all detailed information (extracted from all_programs_detail.xml file in "xml" folder) of that programmes.

+ Analyze the programme_detail_layout.xml layout that contains 10 TextView:

- TextView: display title "Program Details";
- TextView: display title "Program title";
- 1 ImageView : display selected **programme illustration image**;
- Other TextViews: display "program" disruption, qualification level, duration and career opportunities;
- Since the information of each program is quite long so we will use Scrollview and put "Other TextViews" inside;

➔ **The "programme detail layout" structure look like**:



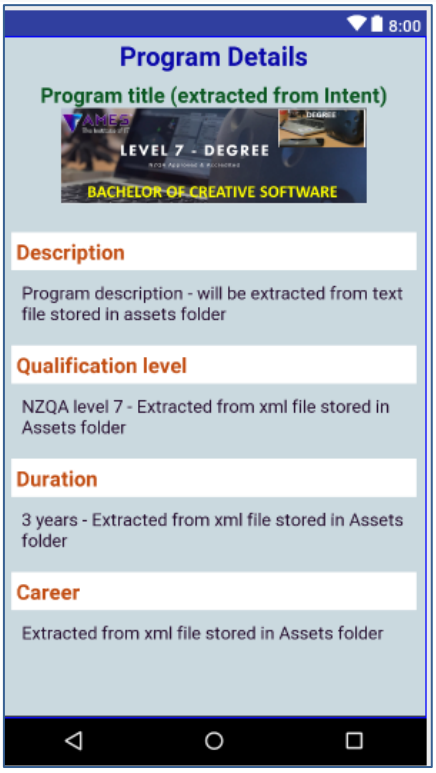
+ Open and edit the programme_detail_layout and edit it as below:

programme_detail_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#cad9df"
    android:orientation="vertical"
    tools:context=".ProgrammeDetail_Activity">

    <!--TextView: display title "Program Details"-->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:gravity="center_vertical|center_horizontal"
        android:padding="2dp"
        android:text="Program Details"
        android:textAlignment="center"
        android:textColor="#0b09a9"
        android:textSize="25sp"
        android:textStyle="bold" />

    <!--TextView: display the name of selected program-->
    <TextView
        android:id="@+id/selected_program"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:gravity="center_vertical|center_horizontal"
        android:text="Program title (extracted from Intent)"
        android:textColor="#0c5823"
        android:textSize="20sp"
        android:textStyle="bold" />
```

```
<!--ImageView: display selected programme illustration image-->
<ImageView
    android:id="@+id/programme_illustration_image"
    android:layout_width="match_parent"
    android:layout_height="100dp"
    android:contentDescription="college logo"
    android:gravity="center"
    android:scaleType="fitCenter"
    android:src="@drawable/illustration_image1" />

<!--ScrollView: display all program description in detail-->
<ScrollView
    android:id="@+id/scrollviewCourseInfo"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_margin="5dp"
    android:padding="5dp"
    android:scrollbars="vertical">

    <!--Define a sub linear-layout inside Scrollview-->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        android:orientation="vertical">

        <!--TextView: display "Description"-->
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:background="#FFFFFFFF"
            android:padding="5dp"
            android:text="Description"
            android:textColor="#c24d12"
            android:textSize="20sp"
            android:textStyle="bold" />

        <!--TextView: display program description-->
        <TextView
            android:id="@+id/programDescription"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:background="@null"
            android:padding="5dp"
            android:text="Program description - will be extracted from text file stored in assets folder"
            android:textColor="#240c2d"
            android:textSize="17sp"
            android:textStyle="normal" />

        <!--TextView: display "Qualification level"-->
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:background="#FFFFFFFF"
            android:padding="5dp"
            android:text="Qualification level"
            android:textColor="#c24d12"
            android:textSize="20sp"
            android:textStyle="bold" />

        <!--TextView: display program qualification level-->
        <TextView
            android:id="@+id/qualification_level"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:background="@null"
            android:padding="5dp"
            android:text="NZQA level 7 - Extracted from xml file stored in Assets folder"
            android:textColor="#240c2d"
            android:textSize="17sp"
            android:textStyle="normal" />

        <!--TextView: display "Duration"-->
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
```

```
            android:background="#FFFFFFFF"
            android:padding="5dp"
            android:text="Duration"
            android:textColor="#c24d12"
            android:textSize="20sp"
            android:textStyle="bold" />

        <!--TextView: display program duration-->
        <TextView
            android:id="@+id/durationTxt"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:background="@null"
            android:padding="5dp"
            android:text="3 years - Extracted from xml file stored in Assets folder"
            android:textColor="#240c2d"
            android:textSize="17sp"
            android:textStyle="normal" />

        <!--TextView: display "Career"-->
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="10dp"
            android:background="#FFFFFFFF"
            android:padding="5dp"
            android:text="Career"
            android:textColor="#c24d12"
            android:textSize="20sp"
            android:textStyle="bold" />

        <!--TextView: display program career-->
        <TextView
            android:id="@+id/careerTxt"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="5dp"
            android:background="@null"
            android:padding="5dp"
            android:text="Extracted from xml file stored in Assets folder"
            android:textColor="#240c2d"
            android:textSize="17sp"
            android:textStyle="normal" />
    </LinearLayout>
</ScrollView>
</LinearLayout>
```



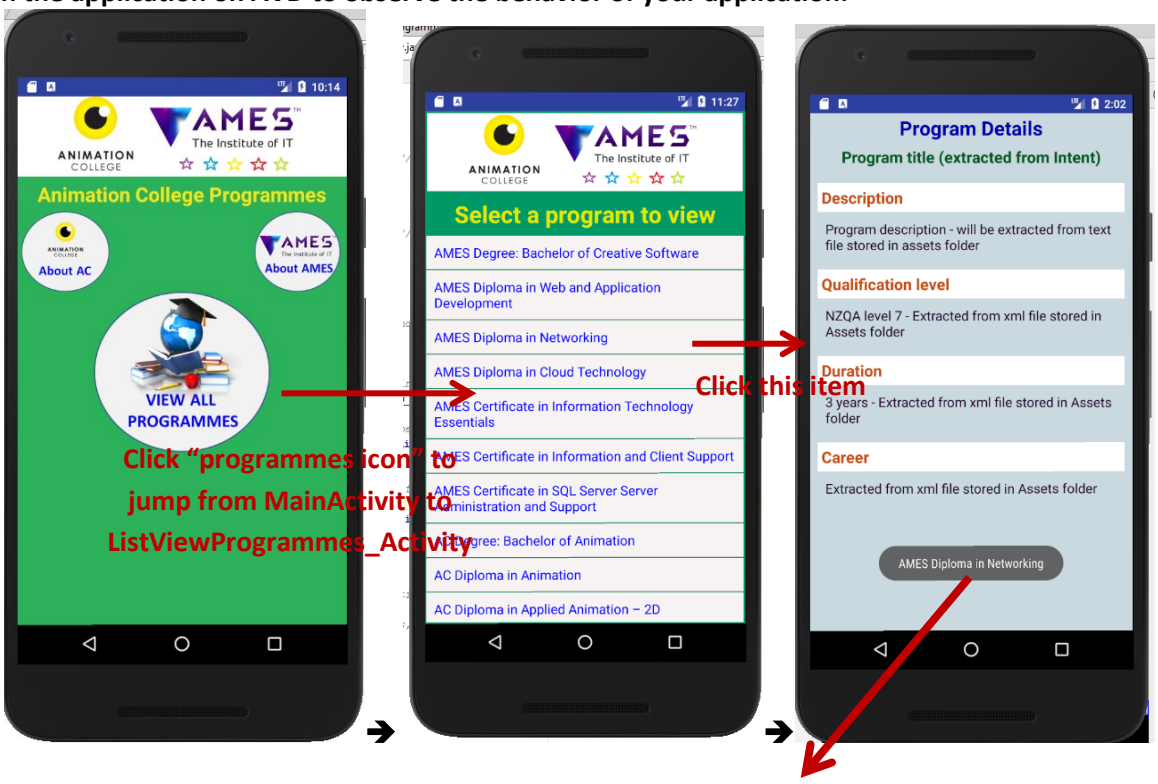
Step 5: Use "Intent" object to jump from ListViewProgrammes_Activity to ProgrammeDetail_Activity when users touch one item in ListView:

In this part, you will learn how to implement a **ListView** to display dynamically the list of programs offered by our college. When user clicks one program, the **programme_detail_layout** appears and shows detailed information of that program.
In terms of programming, when an **item in ListView has been clicked** by users, **use "Intent" object to open up programme_detail_layout**. We also user **putExtra() method** to attach the **program_id** and **program_name** to the **Intent** object. In order to retrieve relevant programme information stored in all_programs_detail.xml file, we will use "XML Parser" library in Android.

+ Open ListviewProgrammes_Activity.java, go to java code *listView.setOnItemClickListener*(), inside *onItemCLick()* method, we add codes to use explicit "Intent" object to open ProgrammeDetail_Activity.
We also attach data to "Intent" object to ProgrammeDetail_Activity:

```
//6: Create an "Intent" object to transition from this activity (ListViewProgrammes_Activity) to
// another activity (ProgrammeDetail_Activity)
Intent intent = new Intent(ListViewProgrammes_Activity.this, ProgrammeDetail_Activity.class);
//Attach "data" associated with "intent" object: "program_name" and "program_id"
intent.putExtra("program_name", programs_array[position]);//Store program title
intent.putExtra("program_id", position);//Store the order of programmes in "all_programmes_array"
//Start activity
startActivity(intent);
```

+ Now, run the application on AVD to observe the behavior of your application:



As you see, so far the "Programme_Detail" layout is empty. We'll implement it later.

Step 6: Let's program ProgrammeDetail_Activity so that the app will extract relevant program data stored in all_programs_detail.xml file and display it on programme_detail layout

In this part, you'll learn how to use "XML Parser" function in Android to retrieve relevant programme information stored in all_programs_detail.xml file and display it on layout.

What is XML Parser?

Source: http://www.tutorialspoint.com/android/android_xml_parsers.htm

XML stands for Extensible Mark-up Language. XML is a very popular format and commonly used for sharing data on the internet. This part explains how to parse the **XML file** and extract necessary information from it.

Android provides three types of XML parsers which are **DOM, SAX and XMLPullParser**. Among all of them android recommend **XMLPullPars**er because it is efficient and easy to use. So we are going to use XMLPullParser for parsing XML.

The first step is to identify the fields in the XML data in which you are interested in. For example. In the XML given below we interested in getting temperature only:

```
<?xml version="1.0"?>
<current>

  <city id="2643743" name="London">
    <coord lon="-0.12574" lat="51.50853"/>
    <country>GB</country>
    <sun rise="2013-10-08T06:13:56" set="2013-10-08T17:21:45"/>
  </city>

  <temperature value="289.54" min="289.15" max="290.15" unit="kelvin"/>
  <humidity value="77" unit="%"/>
  <pressure value="1025" unit="hPa"/>
</current>
```

XML - Elements

An xml file consists of many components. Here is the table defining the components of an XML file and their description:

Sr.No	Component & description
1	**Prolog:** An XML file starts with a prolog. The first line that contains the information about a file is prolog
2	**Events:** An XML file has many events. Event could be like this. Document starts , Document ends, Tag start , Tag end and Text e.t.c
3	**Text:** Apart from tags and events, and xml file also contains simple text. Such as **GB** is a text in the country tag.
4	**Attributes:** Attributes are the additional properties of a tag such as value e.t.c

XML - Parsing

In the next step, we will create XMLPullParser object, but in order to create that we will first create XmlPullParserFactory object and then call its **newPullParser() method** to create XMLPullParser. Its syntax is given below –

```
private XmlPullParserFactory xmlFactoryObject = XmlPullParserFactory.newInstance();
private XmlPullParser myparser = xmlFactoryObject.newPullParser();
```

The next step involves specifying the file for XmlPullParser that contains XML. It could be a file or could be a Stream. In our case it is a stream.Its syntax is given below –

```
myparser.setInput(stream, null);
```

The last step is to parse the XML. An XML file consist of events, Name, Text, AttributesValue e.t.c. So XMLPullParser has a separate function for parsing each of the component of XML file. Its syntax is given below:

```
int event = myParser.getEventType();
while (event != XmlPullParser.END_DOCUMENT)
{
  String name=myParser.getName();
  switch (event){
    case XmlPullParser.START_TAG:
    break;

    case XmlPullParser.END_TAG:
    if(name.equals("temperature")){
      temperature = myParser.getAttributeValue(null,"value");
    }
    break;
  }
  event = myParser.next();
}
```

The method **getEventType** returns the type of event that happens. e.g: **Document start, tag start** e.t.c. The method **getName** returns the name of the tag and since we are only interested in temperature, so we just check in conditional statement that if we got a temperature tag , we call the method **getAttributeValue** to return us the value of temperature tag.

Apart from these methods, there are other methods provided by this class for better parsing XML files. These methods are listed below:

Sr.No	Method & description
1	**getAttributeCount():** This method just Returns the number of attributes of the current start tag
2	**getAttributeName(int index):** This method returns the name of the attribute specified by the index value
3	**getColumnNumber():** This method returns the Returns the current column number, starting from 0.
4	**getDepth():** This method returns Returns the current depth of the element.
5	**getLineNumber():** Returns the current line number, starting from 1.
6	**getNamespace():** This method returns the name space URI of the current element.
7	**getPrefix():** This method returns the prefix of the current element
8	**getName():** This method returns the name of the tag
9	**getText():** This method returns the text for that particular element
10	**isWhitespace():** This method checks whether the current TEXT event contains only whitespace characters.

Read more:

[1]. Android - XML Parser Tutorial: http://www.tutorialspoint.com/android/android_xml_parsers.htm

[2]. Android - XML Parser Tutorial use of XMLPullParser parse xml on web api|android studio: https://www.youtube.com/watch?v=FQKmFCAYcmA&index=20&list=PLYDICjs3cYp_uBzvBmHZN_H6clYs7hq7W

[3]. Create a Weather App on Android: http://code.tutsplus.com/tutorials/create-a-weather-app-on-android--cms-21587

+ Open ProgrammeDetail_Activity.java, keep the auto-generated codes and edit it as following:

- Declare an array of 12 illustration images for 12 programmes;
- Extract data attached to "Intent" to know which programme has been selected and which programme illustration image must be displayed;
- Create XMLPullParser object and retrieve the xml file stored in Assets folder to extract data;

Here is the full java code of the ProgramDetail_Activity.java file:

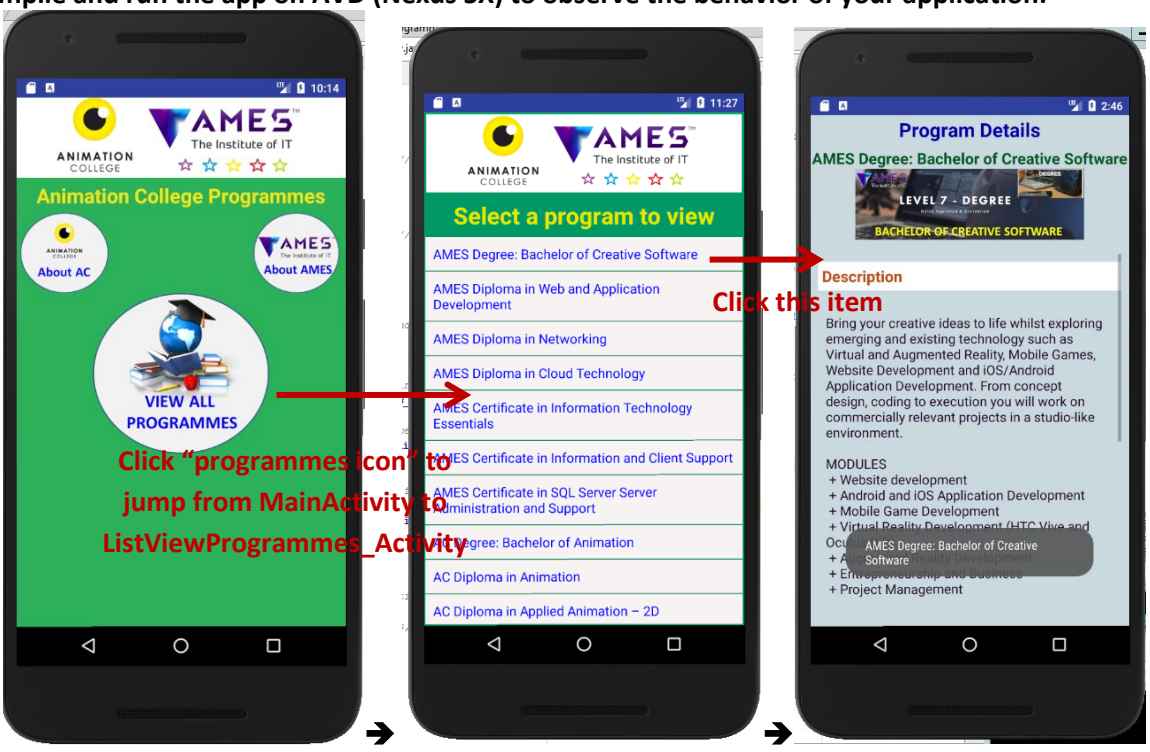
Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:

ProgramDetail_Activity.java

```
public class ProgrammeDetail_Activity extends AppCompatActivity {
    ////////////////////////////////////////////////////////////////////////////////////////
    //1: Declare variables
    private TextView selected_program;
    private ImageView programmeIllustrationImage;
    private TextView programDescription, programQualificationLevel, programDuration, programCareer;
    //Declare an array of 12 illustration images for 12 programmes
    private int[] illustration_images_array = {R.drawable.illustration_image1,
            R.drawable.illustration_image2, R.drawable.illustration_image3, R.drawable.illustration_image4,
            R.drawable.illustration_image5, R.drawable.illustration_image6, R.drawable.illustration_image7,
            R.drawable.illustration_image8, R.drawable.illustration_image9, R.drawable.illustration_image10,
            R.drawable.illustration_image11, R.drawable.illustration_image12};
    //Other "String" variables storing programme details
    private String programTitle, description, qualificationLevel, duration, career;
    private String program_name;//receive from ListviewProgrammes_Activity
    private int program_id;//receive from ListviewProgrammes_Activity

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

        //2: Find references and do casting for visual elements: make connections between UI and java code
        selected_program = (TextView) findViewById(R.id.selected_program);
        programmeIllustrationImage = (ImageView) findViewById(R.id.programme_illustration_image);
        programDescription = (TextView) findViewById(R.id.programDescription);
        programQualificationLevel = (TextView) findViewById(R.id.qualification_level);
        programDuration = (TextView) findViewById(R.id.durationTxt);
        programCareer = (TextView) findViewById(R.id.careerTxt);

        //3: Extract data stored in "Intent" object sent from ListView_Activity and display it
        //Extract "program_name" attached with "Intent" -> then display it in "selected_program" TextView
        program_name = getIntent().getExtras().getString("program_name");
        selected_program.setText(program_name);
        //Extract "program_id" attached with "Intent" -> display the corresponding programme image
        program_id = getIntent().getExtras().getInt("program_id");
        programmeIllustrationImage.setImageResource(illustration_images_array[program_id]);

        //4: Extract the relevant information from "all_programs_detail.xml" file by using XML parser
        try {
            //5: Open program_detail.xml file stored in Assets folder
            InputStream inputStream = getAssets().open("all_programs_detail.xml");

            //6: Use of XML DOM Parser for extracting data
            DocumentBuilderFactory documentBuilderFactory =
DocumentBuilderFactory.newInstance().newInstance();
            DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
            Document document = documentBuilder.parse(inputStream);
            Element element = document.getDocumentElement();
            element.normalize();

            //7: Read all the nodes containing tag "program"
            NodeList nodeList = document.getElementsByTagName("program");

            //8: Loop through all nodes to find the relevant selected program
            boolean program_found = false;
            for (int i = 0; i < nodeList.getLength(); i++) {
                //9: Get "node" in xml file
                Node node = nodeList.item(i);
                Element sub_Element = (Element) node;
                //9: Get the program "title"
                programTitle = sub_Element.getElementsByTagName("title").item(0)
                        .getChildNodes().item(0).getNodeValue();
                programDescription.setText(programTitle);
                //10: Check if the program title is the selected program. If yes, display its detail
```

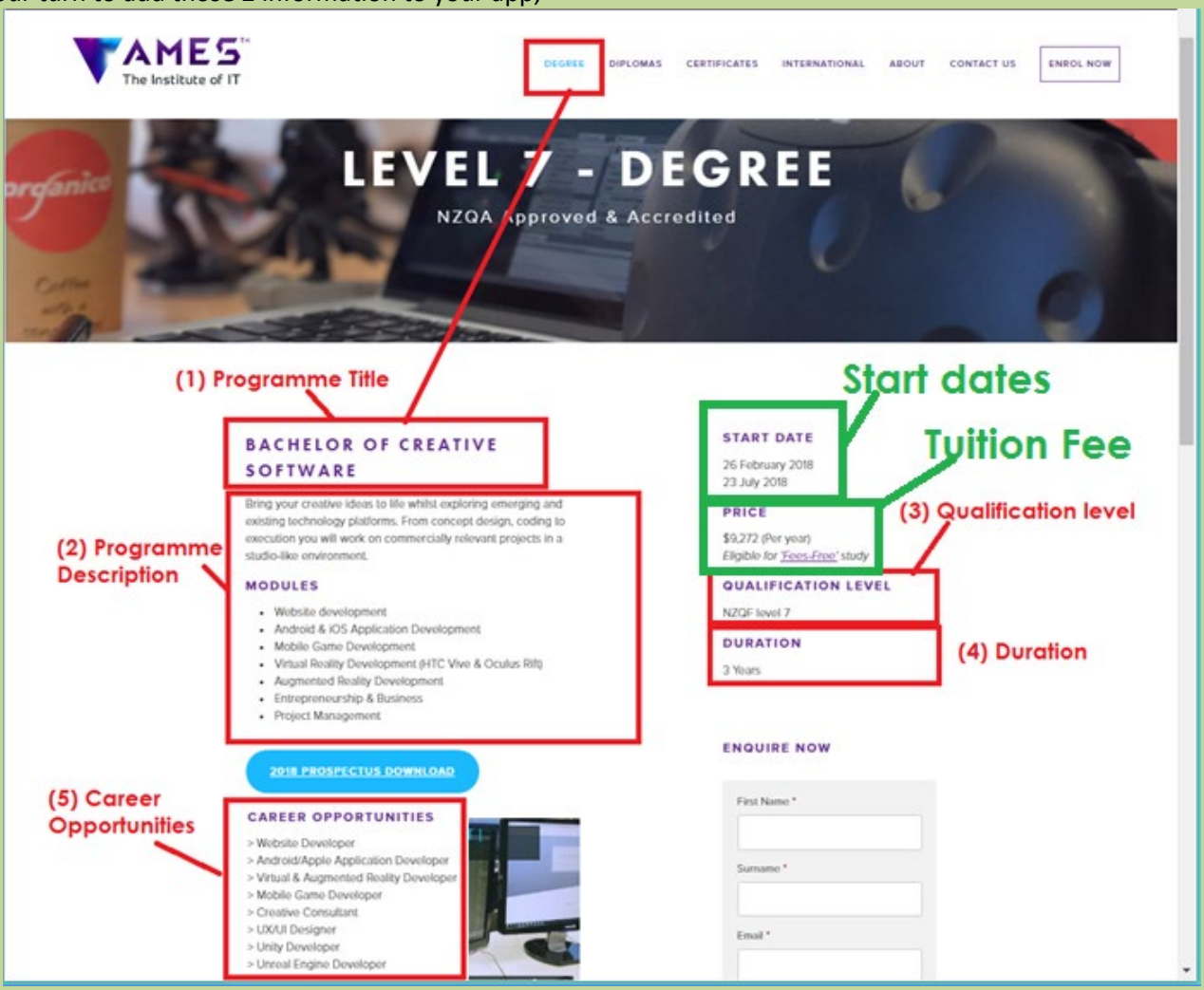
```
                if (programTitle.contains(program_name)) {
                    description = sub_Element.getElementsByTagName("description").item(0)
                            .getChildNodes().item(0).getNodeValue();
                    qualificationLevel = sub_Element.getElementsByTagName("qualification_level").item(0)
                            .getChildNodes().item(0).getNodeValue();
                    duration = sub_Element.getElementsByTagName("duration").item(0)
                            .getChildNodes().item(0).getNodeValue();
                    career = sub_Element.getElementsByTagName("career").item(0)
                            .getChildNodes().item(0).getNodeValue();
                    //11: Change the variable program_found to "true"
                    program_found = true;
                }
            }

            //11: If not found any program in the xml file, assign all variables to "Not found"
            if (!program_found) {
                description = "Not found";
                qualificationLevel = "Not found";
                duration = "Not found";
                career = "Not found";
            }

            //12: Display the extracted program details into the TextView on Layout
            programDescription.setText(description);
            programQualificationLevel.setText(qualificationLevel);
            programDuration.setText(duration);
            programCareer.setText(career);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

+ Now, compile and run the app on AVD (Nexus 5X) to observe the behavior of your application:



The ProgrammeDetail_Activity display (1) Programme description & modules, (2) NZQA level, (3) Duration and (4) Job opportunities. However there are not information of "Start dates" and "Tuition fee".

It's your turn to add those 2 information to your app;



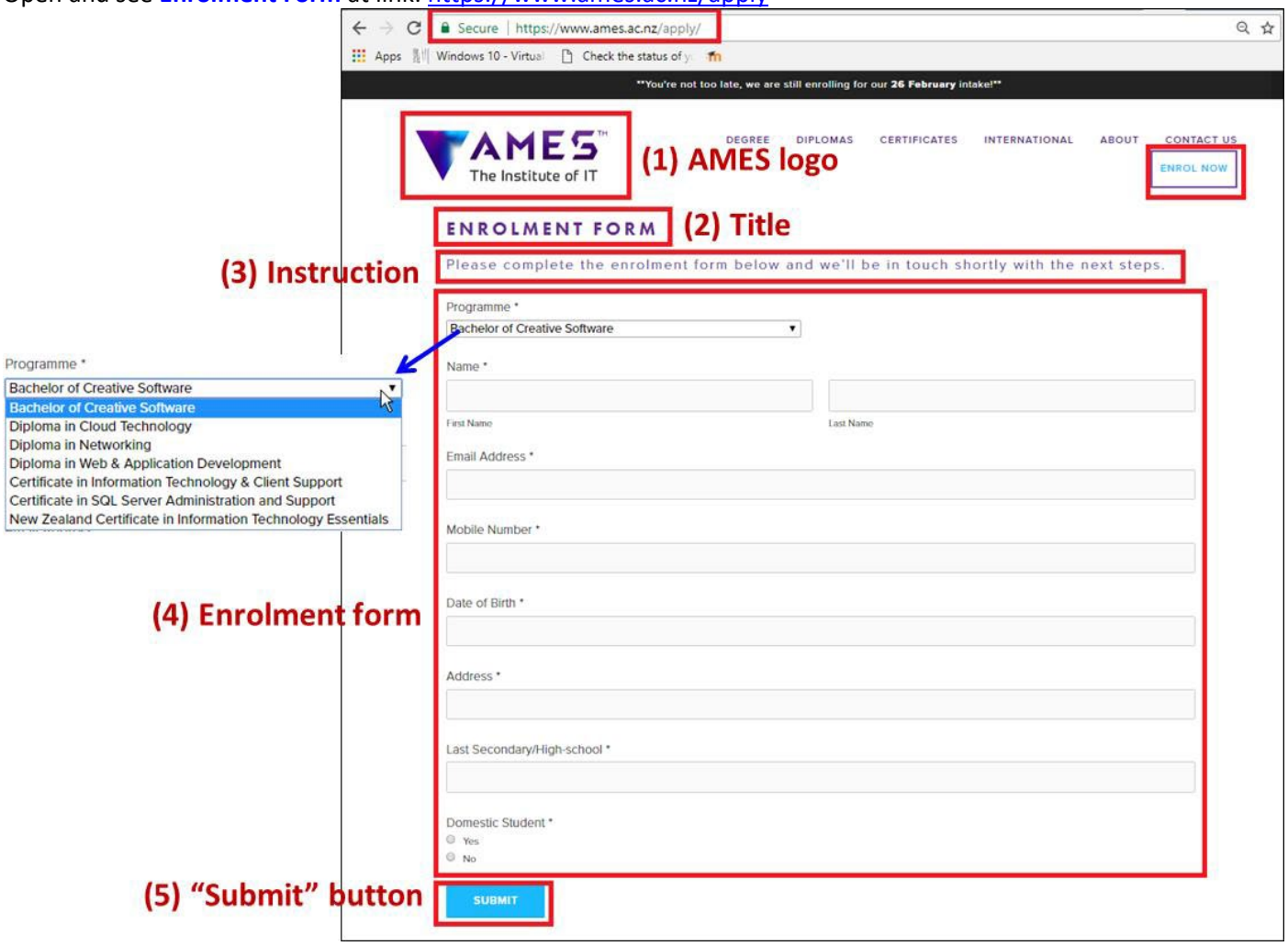
PART 6: Add the sixth Activity

"Enrol_Activity"

Add a new Activity, called "**Enrol_Activity**", to the project:

- The "**Online Application Form**" that allow users to fill the application form and then send it out to the AC college by Email or SMS.
- The enrolment form includes the fields needed to be inputed as shown on **AMES website**.

Open and see **Enrolment Form** at link: https://www.ames.ac.nz/apply



All the fields with asterisk symbol (*) are required to have data. The app checks them all to make sure that they are filled in. If not, pop up a message to ask users to complete before proceeding.

Step 1: Add a new Activity to the project

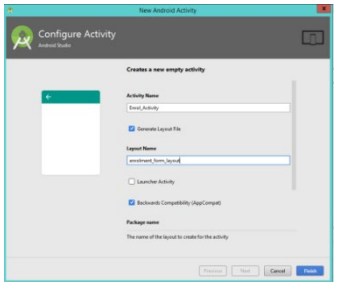
+ Add a new Activity, called "Enrol_Activity" and its associated "enrolment_form_layout"

*_Right click on the folder containing **MainActivity** → "**New**" → "**Activity**" → "**Empty Activity**" → a "Configure Acitivy" window is appeared:*

*_Enter the "Activity Name": **Enrol_Activity***

*_Enter the "Layout Name": **enrolment_form_layout***

*_Click "**Finish**"*

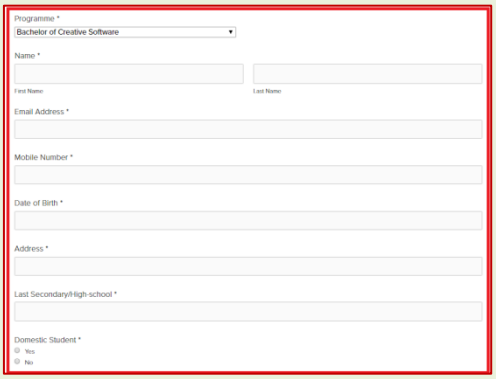


Step 2: Analyze & design enrolment_form_layout.xml

In the "enrolment_form" layout, we will show 5 information:

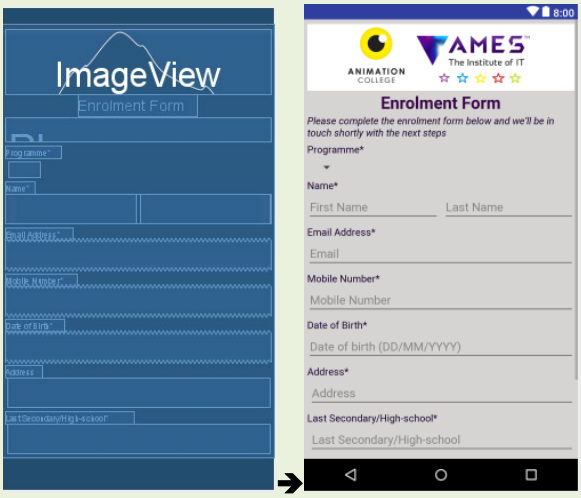
1. **college logo:**

2. **Title:** ***Enrolment Form***
3. **Instruction:** ***Please complete the enrolment form below and we'll be in touch shortly with the next steps*.**
4. **Enrolment form body:**



5. **"Submit" button**

➔ **The "Enrolment Form layout" structure looks like**:



Step 3: Design Graphic User Interface (GUI): enrolment_form_layout.xml

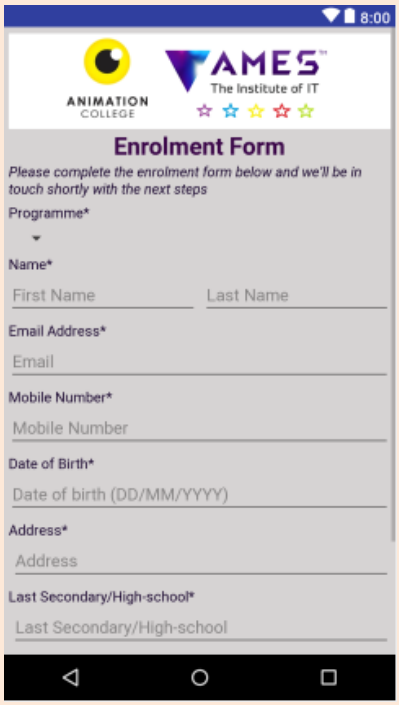
+ Now design the about_ac_layout.xml layout that contains:

- **1 ImageView: displays ames logo;**
- **1 TextView: displays title "Enrolment Form";**
- **1 TextView: display instruction "Please complete the enrolment form below and we'll be in touch shortly with the next steps";**
- **1 sub-layout contain: displays all fields of enrolment form;**
- **1 Button: "Submit" button to send enrolment form via Email or SMS**

+ Open and edit the enrolment_form_layout.xml as below:

enrolment_form_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#d6d3d3"
    android:scrollbars="vertical"
    tools:context=".Enrol_Activity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:padding="5dp">

        <!--ImageView: display ames logo-->
        <ImageView
            android:id="@+id/collegelogo"
            android:layout_width="match_parent"
            android:layout_height="100dp"
            android:layout_alignParentTop="true"
            android:layout_marginTop="2dp"
            android:background="#FFFFFFFF"
            android:contentDescription="ames logo"
            android:padding="5dp"
            android:src="@drawable/collegelogo_transparent" />

        <!--TextView: display title-->
        <TextView
            android:id="@+id/title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Enrolment Form"
            android:textColor="#3b0749"
            android:textSize="25sp"
            android:textStyle="bold" />

        <!--TextView: display instruction-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:text="Please complete the enrolment form below and we'll be in touch shortly with the
next steps"
            android:textColor="#15022b"
            android:textSize="15sp"
            android:textStyle="italic" />

        <!--TextView: Programme*-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginTop="5dp"
            android:text="Programme*"
            android:textColor="#15022b"
            android:textSize="15sp"
            android:textStyle="normal" />

        <!--Spinner: display a drop-down list of 7 AMES programmes-->
        <Spinner
            android:id="@+id/spinner_ames_programmes"
            android:layout_width="wrap_content"
```



```
        android:layout_height="wrap_content"
        android:textColor="#0818a1"
        android:layout_margin="5dp" />

    <!--TextView: Name*-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left"
        android:text="Name*"
        android:textColor="#15022b"
        android:textSize="15sp"
        android:textStyle="normal" />

    <!--Two EditText: for users to enter "first name" and "last name"-->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <!--EditText: for entering first name-->
        <EditText
            android:id="@+id/firstnameEdt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="3dp"
            android:layout_weight="1"
            android:hint="First Name"
            android:textColor="#0818a1"
            android:inputType="textPersonName" />

        <!--EditText: for entering last name-->
        <EditText
            android:id="@+id/lastnameEdt"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="3dp"
            android:layout_weight="1"
            android:hint="Last Name"
            android:textColor="#0818a1"
            android:inputType="textPersonName" />
    </LinearLayout>

    <!--TextView: Email Address*-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left"
        android:layout_marginTop="5dp"
        android:text="Email Address*"
        android:textColor="#15022b"
        android:textSize="15sp"
        android:textStyle="normal" />

    <!--EditText: for entering email address-->
    <EditText
        android:id="@+id/emailEdt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:textColor="#0818a1"
        android:inputType="textEmailAddress" />

    <!--TextView: Mobile Number*-->
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="left"
        android:layout_marginTop="5dp"
        android:text="Mobile Number*"
        android:textColor="#15022b"
        android:textSize="15sp"
        android:textStyle="normal" />

    <!--EditText: for entering phone number-->
    <EditText
        android:id="@+id/phoneNumberEdt"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="Mobile Number"
        android:textColor="#0818a1"
        android:inputType="number" />
```

```
        <!--TextView: Date of Birth*-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginTop="5dp"
            android:text="Date of Birth*"
            android:textColor="#15022b"
            android:textSize="15sp"
            android:textStyle="normal" />

        <!--EditText: for entering date of birth-->
        <EditText
            android:id="@+id/birthDateEdt"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textColor="#0818a1"
            android:hint="Date of birth (DD/MM/YYYY)"
            android:inputType="date" />

        <!--TextView: Domestic Student*-->
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left"
            android:layout_marginTop="5dp"
            android:text="Domestic student*"
            android:textColor="#15022b"
            android:textSize="15sp"
            android:textStyle="normal" />

        <RadioGroup
            android:id="@+id/optionGroup"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical">
            <!--RadioButton: Yes-->
            <RadioButton
                android:id="@+id/domestic_Yes"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:checked="true"
                android:text="Yes"
                android:textColor="#ff00f2"
                android:textSize="15dp"
                android:textStyle="normal" />
            <!--RadioButton: No-->
            <RadioButton
                android:id="@+id/domestic_No"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:checked="false"
                android:text="No"
                android:textColor="#ff00f2"
                android:textSize="15dp"
                android:textStyle="normal" />
        </RadioGroup>

        <!--Button: submit-->
        <Button
            android:id="@+id/submitBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:layout_gravity="center"
            android:layout_margin="5dp"
            android:layout_marginTop="5dp"
            android:background="#0be6d3"
            android:paddingBottom="10dp"
            android:paddingLeft="20dp"
            android:paddingRight="20dp"
            android:paddingTop="10dp"
            android:text="Submit"
            android:textColor="#f9f9f5"
            android:textSize="20sp"
            android:textStyle="bold" />
    </LinearLayout>
</ScrollView>
```

Step 4: Use "Intent" object to jump from MainActivity to Enrol_Activity when users touch "Apply icon" on mains screen:

In this part, we will add an image icon (**apply_icon.png**) to **activity_main.xml layout**. When users touchs this icon, the app will open **Enrol_Activity** to display its associated **enrolment_form_layout.xml**.

+ Open activity_main.xml and add "apply_icon" icon below "view app programmes" and located on the left hand side of the screen:

_First, copy the apply_icon.png image to "drawable" folder:

Image: apply_icon.png

_Then, add an ImageView to main layout (activity_main.xml): at the bottom of current codes:

```
<!--ImageView: display apply icon-->
<ImageView
    android:id="@+id/apply"
    android:layout_width="110dp"
    android:layout_height="100dp"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/viewAllProgrammes"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:background="@drawable/apply_icon"
    android:contentDescription="online apply icon" />
```

+ Open MainActivity.java file and add the below codes to implement the function:

When users touch "apply" icon, the app will move to Enrol_Activity to display the enrolment_form:

Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:

_Declare a variable "apply" right after the class declaration:

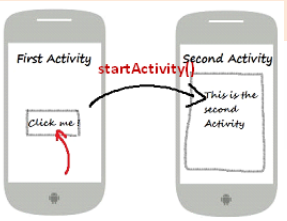
```
//Apply - 1: Declare "apply" variable as ImageView type
private ImageView apply;
```

_Inside onCreate() method, add code to find reference for "viewAllProgrammes":

```
////////////////////////////////////////////////////////////////////////////////////////////
//Apply - 2: Find references and do casting for "apply"
apply = (ImageView) findViewById(R.id.apply);
```

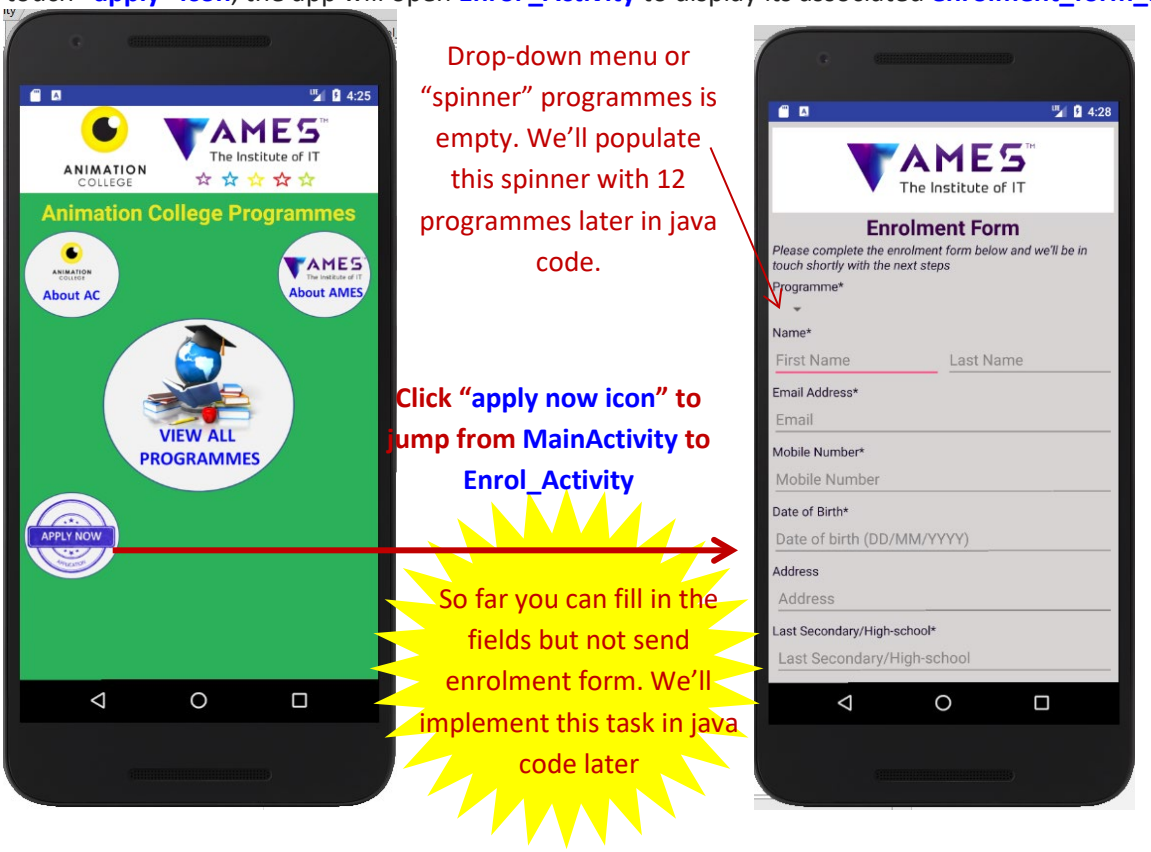
_Inside onCreate() method, set click listener for "viewAllProgrammes" imageView. Inside onCick() method of listener, create an Intent object to transition from this activity (MainActivity) to activity (ListViewProgrammes_Activity):

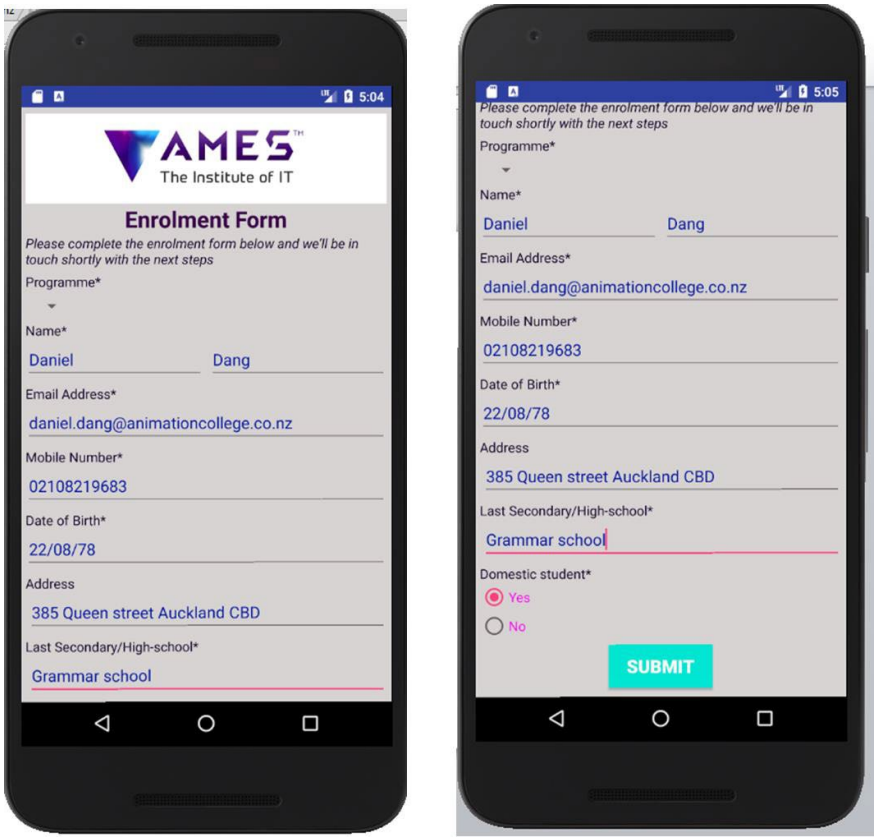
```
////////////////////////////////////////////////////////////////////////////////////////////
//View All Programmes - 3: Set click listener for the "viewAllProgrammes" ImageView
apply.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Create an Intent object to transition from this activity (MainActivity) to
        // another activity (Enrol_Activity)
        Intent intent = new Intent(MainActivity.this, Enrol_Activity.class);
        startActivity(intent);//Start Activity
    }
});
```



+ Run the app on AVD (Nexus 5X) the see the result:

If users touch **"apply" icon**, the app will open **Enrol _Activity** to display its associated **enrolment_form_layout**.





Step 6: Let's program Enrol_Activity so that the app will collect all entered data and send to our college email address

In this part, first of all, you will learn about Spinner in Android and learn how to populate the drop-down menu (Spinner) containing 12 ames programmes. Then you'll learn how to use **"SendingSMS" API** and **"Sending Email" function** in Android

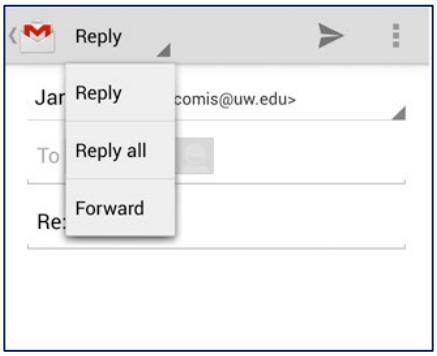
Implement Spinner (Drop-down) menu: Populate the drop-down menu (Spinner) containing 7 ames programmes

+ What is Spinner in Android?

Source:
https://www.tutorialspoint.com/android/android_spinner_control.htm

Spinner allows you to select an item from a drop down menu

For example, when you are using Gmail application you would get drop down menu as shown below, you need to select an item from a drop down menu.



SPINNER EXAMPLE

+ Open Enrol_Activity.java file and add the below codes to populate the drop-down menu (Spinner) containing 7 ames programmes:

Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:

_Declare a variable "selectPrograms_spinner" right after the class declaration:

```
//Spinner - 1: declare variable "selectPrograms_spinner"
private Spinner selectPrograms_spinner;
private String[] programs_array;
private int program_selected_id;
```

_Inside onCreate() method, add code to find reference & do casting for "selectPrograms_spinner":

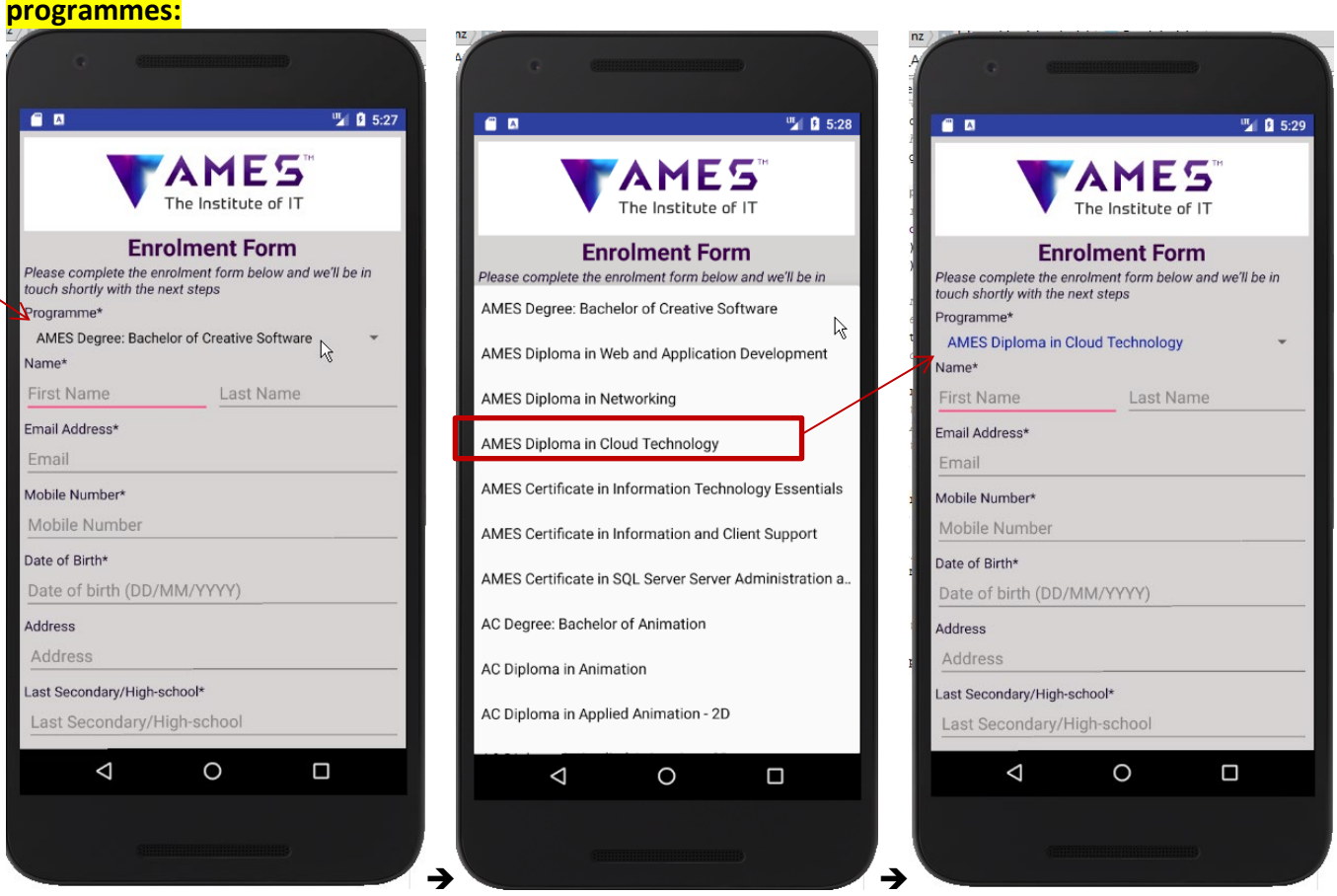
```
//////////////////////////////////////////////////////////////////////////////////
//Spinner - 2: Find control elements and do casting for "selectPrograms_spinner"
selectPrograms_spinner = (Spinner) findViewById(R.id.spinner_ames_programmes);
```

_Inside onCreate() method, Populate spinner or drop-down menu with 12 AMES programmes:

```
//////////////////////////////////////////////////////////////////////////////////
//Spinner - 3: Populate spinner or drop-down menu with 12 AMES programmes
//1: Retrieve 12 AMES programmes declared in strings.xml file ("all_programmes_array")
programs_array = getResources().getStringArray(R.array.all_programmes_array);
//2: Declare a list of programmes to be selected
List<String> program_list = new ArrayList<>(Arrays.asList(programs_array));
//3: Create Adapter [ArrayAdapter, datatype:String] for spinner
ArrayAdapter<String> program_Adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_spinner_item,
        program_list);
//4: Define drop layout style - list view with radio button for program_Adapter
program_Adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
//5: Attach Adapter to Spinner: Attach program_Adapter programs_spinner ()
selectPrograms_spinner.setAdapter(program_Adapter);
//6: Set the default
selectPrograms_spinner.setSelection(0, true);// Position = 0: BCS programme
//7: Set the text color of the spinner's default item
View program spinner item view = selectPrograms_spinner.getSelectedView();
((TextView) program_spinner_item_view).setTextColor(Color.parseColor("#FF000000"));
((TextView) program_spinner_item_view).setTextSize(15);

//8: Set the listener when each option (item) of programs_spinner has been clicked,
// and then set color for selected item
selectPrograms_spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
```

```
    //Override onItemSelected() method
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        //parent = AdapterView where the selection happend
        //view = the view within the AdapterView that was clicked
        //position = the position of the view in the adapter
        //id = the row id of the item that has ben selected
        //Set the text color and text size for the selected item
        ((TextView) view).setTextColor(Color.parseColor("#FF0818a1"));
        //Try other method: Color.rgb(200,0,0);
        ((TextView) view).setTextSize(16);
        //Get the id of selected item, this id will be used later process
        program_selected_id = position;
    }

    //Override onNothingSelected() method
    @Override
    public void onNothingSelected(AdapterView<?> parent) {
    }
});
```

+ Compile and run the app on AVD (Nexus 5X) to see the spinner now is populated with 12 AC programmes:



Implement "Submit" function: When users click "Submit" button, the app will check all the fields in the form with *symbol to make sure that all those fields are filled in. Then the app uses SmsManager API to send SMS to the college or use implicit Intent (Intent.ACTION_SEND) to send enrolment form by email to the college

+ What is Sending SMS API in Android?

Source: https://www.tutorialspoint.com/android/android_sending_sms.htm

In Android, you can use **SmsManager API** or **devices Built-in SMS application** to send SMS's.

SmsManager API:

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage("phoneNo", null, "sms message", null, null);
```

Built-in SMS application:

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
sendIntent.putExtra("sms_body", "default content");
sendIntent.setType("vnd.android-dir/mms-sms");
startActivity(sendIntent);
```

Both need **SEND_SMS permission**.

```
<uses-permission android:name="android.permission.SEND_SMS" />
```

Apart from the above method, there are few other important functions available in SmsManager class. These methods are listed below –

Sr.No.	Method & Description
1	**ArrayList<String> divideMessage(String text)** This method divides a message text into several fragments, none bigger than the maximum SMS message size.
2	**static SmsManager getDefault()** This method is used to get the default instance of the SmsManager
3	**void sendDataMessage(String destinationAddress, String scAddress, short destinationPort, byte[] data, PendingIntent sentIntent, PendingIntent deliveryIntent)** This method is used to send a data based SMS to a specific application port.
4	**void sendMultipartTextMessage(String destinationAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents, ArrayList<PendingIntent> deliveryIntents)** Send a multi-part text based SMS.
5	**void sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent)** Send a text based SMS.

+ How to send Email in Android by using implicit Intent (Intent.ACTION_SEND)?

Source: https://www.tutorialspoint.com/android/android_sending_email.htm

***Email** is messages distributed by electronic means from one system user to one or more recipients via a network.*

Before starting Email Activity, You must know **Email functionality with intent**, Intent is carrying data from one component to another component with-in the application or outside the application.

To send an email from your application, you don't have to implement an email client from the beginning, but you can use an existing one like the default Email app provided from Android, Gmail, Outlook, K-9 Mail etc. For this purpose, we need to write

an Activity that launches an email client, using an implicit Intent with the right action and data. In this example, we are going to send an email from our app by using an Intent object that launches existing email clients.

Following section explains different parts of our Intent object required to send an email.

Intent Object - Action to send Email

You will use **ACTION_SEND** action to launch an email client installed on your Android device. Following is simple syntax to create an intent with ACTION_SEND action.

```
Intent emailIntent = new Intent(Intent.ACTION_SEND);
```

Intent Object - Data/Type to send Email

To send an email you need to specify **mailto:** as URI using setData() method and data type will be to **text/plain** using setType() method as follows:

```
emailIntent.setData(Uri.parse("mailto:"));
emailIntent.setType("text/plain");
```

Intent Object - Extra to send Email

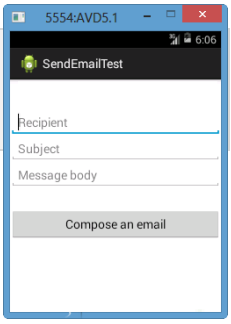
Android has built-in support to add TO, SUBJECT, CC, TEXT etc. fields which can be attached to the intent before sending the intent to a target email client. You can use following extra fields in your email –

Sr.No.	Extra Data & Description
1	**EXTRA_BCC:** A String[] holding e-mail addresses that should be blind carbon copied.
2	**EXTRA_CC:** A String[] holding e-mail addresses that should be carbon copied.
3	**EXTRA_EMAIL:** A String[] holding e-mail addresses that should be delivered to.
4	**EXTRA_HTML_TEXT:** A constant String that is associated with the Intent, used with ACTION_SEND to supply an alternative to EXTRA_TEXT as HTML formatted text.
5	**EXTRA_SUBJECT:** A constant string holding the desired subject line of a message.
6	**EXTRA_TEXT:** A constant CharSequence that is associated with the Intent, used with ACTION_SEND to supply the literal data to be sent.
7	**EXTRA_TITLE:** A CharSequence dialog title to provide to the user when used with a ACTION_CHOOSER.

Here is an example showing you how to assign extra data to your intent:

```
emailIntent.putExtra(Intent.EXTRA_EMAIL  , new String[]{"Recipient"});
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "subject");
emailIntent.putExtra(Intent.EXTRA_TEXT   , "Message Body");
```

The output of above code is as shown in the beside image.



Read more:

[1]. Android Essentials: Creating Simple User Forms: https://code.tutsplus.com/tutorials/android-essentials-creating-simple-user-forms--mobile-1758
[2]. Activities: https://developer.android.com/guide/components/activities.html
[3]. Android Activity: https://www.tutorialspoint.com/android/android_acitivities.htm
[4]. Intents and Intent Filters: https://developer.android.com/guide/components/intents-filters.html
[5]. Intents and Filters: https://www.tutorialspoint.com/android/android_intents_filters.htm
[6]. Android – Sending SMS: https://www.tutorialspoint.com/android/android_sending_sms.htm
[7]. SmsManager: https://developer.android.com/reference/android/telephony/SmsManager.html

+ Open Enrol_Activity.java, edit it as following:

1. **Check all the fields in the form with *symbol** to make sure that all those fields are filled in;
2. **Use SmsManager API** to send SMS to the college or use implicit **Intent (Intent.ACTION_SEND)** to send enrolment form by email to the college;

+ Open Enrol_Activity.java file and add the below codes submit enrolment:

Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:

_Declare few variables right after the class declaration:

```
//Enrolment form - 1: Declare variables
private EditText firstname, lastname;
private EditText emailAddress, phoneNumber, birthDate;
//private EditText address, lastSecondary;
private RadioButton domestic_yes, domestic_no;
private Button submitBtn;
private String enrolment_form;
```

_Inside onCreate() method, add code to find reference & do casting for all visual elements on UI:

```
////////////////////////////////////////////////////////////////////////////
//Enrolment form - 2: Find control elements and do casting for all UI elements
firstname = (EditText) findViewById(R.id.firstnameEdt);
lastname = (EditText) findViewById(R.id.lastnameEdt);
emailAddress = (EditText) findViewById(R.id.emailEdt);
phoneNumber = (EditText) findViewById(R.id.phoneNumberEdt);
birthDate = (EditText) findViewById(R.id.birthDateEdt);
//address
//lastSecondary
domestic_yes = (RadioButton) findViewById(R.id.domestic_Yes);
domestic_no = (RadioButton) findViewById(R.id.domestic_No);
submitBtn = (Button) findViewById(R.id.submitBtn);
```

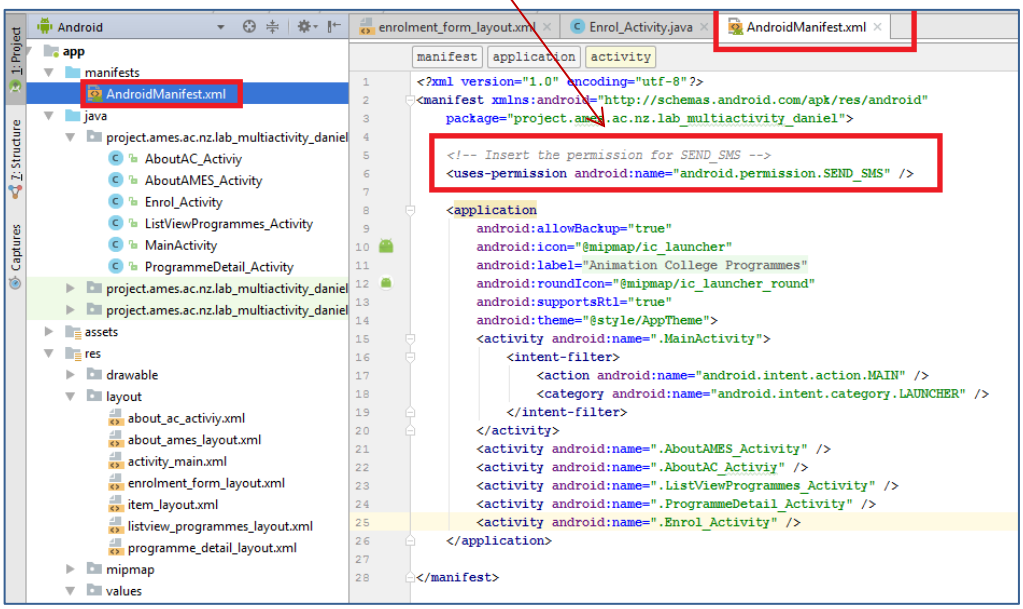
_Inside onCreate() method, collect info on Enrolment Form and send it via SMS to college phone number:

```
////////////////////////////////////////////////////////////////////////////
//Enrolment form - 3: When users click "Submit" button, check all the fields in the form
// with *symbol to make sure that all those fields are filled in.
// Then the app uses SmsManager API to send SMS to the college or use implicit Intent
// (Intent.ACTION_SEND) to send enrolment form by email to the college
//Set Click Listener for "Submit" button
submitBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //1: Get all information on Enrolment Form
        //Get programme selected/interested
        String program_selected = programs_array[program_selected_id];
        //Get full-name
        String fullname = firstname.getText().toString() + " " + lastname.getText().toString();
        //Get email address
        String email = emailAddress.getText().toString();
        //Get phone number
        String phone = phoneNumber.getText().toString();
        //Get date of birth
        final String birthday = birthDate.getText().toString();
        //Get "Domestic" student status
        String domestic = "";
        if (domestic_yes.isChecked()) {
            domestic = "Yes";
        } else if (domestic_no.isChecked()) {
            domestic = "No";
        }

        //2: Compose the content of "enrolment form" by combining all fields in the enrolment form
        enrolment_form = "Fullname: " + fullname + ". "
                + "Email: " + email + ". "
                + "DoB: " + birthday + ". "
                + "Mobile: " + phone + ". "
                + "Program: " + program_selected + ". "
                + "Domestic: " + domestic;

        //3: Check whether firstname, lastname, email, phone, DoB have been filled in
        if (TextUtils.isEmpty(firstname.getText().toString())
                || TextUtils.isEmpty(lastname.getText().toString())
                || TextUtils.isEmpty(emailAddress.getText().toString())
                || TextUtils.isEmpty(phoneNumber.getText().toString())
```

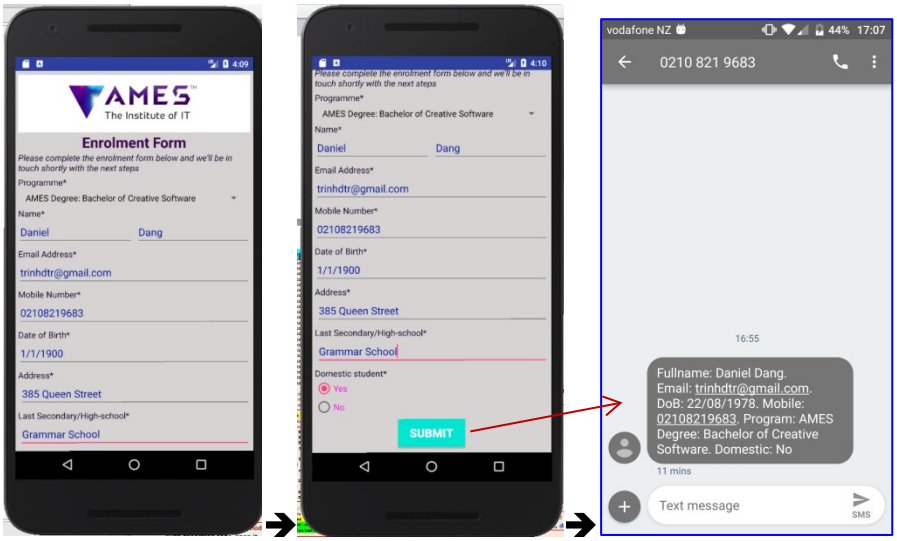
```
                || TextUtils.isEmpty(birthDate.getText().toString())) {
            //If any of these fields is empty, pop up a message to ask user to fill in
            Toast.makeText(getApplicationContext(), "Fill all fields (*) first",
                    Toast.LENGTH_LONG).show();
            //Return enrolment_form_layout
            return;
        }

        //4: Send enrolment form by SMS to college number: 02108219683
        //College contact information
        String college_phoneNumber = "02108219683";//Change to your phone number to TEST
        String college_email = "bcs.amesit@gmail.com";
        //IMPLEMENT SENDING SMS IN ANDROID
        //Check the "android.permission.SEND_SMS" permission in AndroidManifest
        if (ContextCompat.checkSelfPermission(Enrol_Activity.this,
                Manifest.permission.SEND_SMS) != PackageManager.PERMISSION_GRANTED) {
            //Should we shown an explanation?
            if (ActivityCompat.shouldShowRequestPermissionRationale(Enrol_Activity.this,
                    Manifest.permission.SEND_SMS)) {
                Toast.makeText(getBaseContext(), "Permission required", Toast.LENGTH_SHORT).show();
            } else {
                //No explanation needed, we can request the permission
                ActivityCompat.requestPermissions(Enrol_Activity.this,
                        new String[]{Manifest.permission.SEND_SMS}, 1);
            }

        } else {
            //Permission already granted run sms send -> send enrolment_from by SMS
            //Initiate an object "intent" linked to SmsSender class
            Intent intent = new Intent(getApplicationContext(), Enrol_Activity.class);
            //initiate an object of PendingIntent
            PendingIntent pi = PendingIntent.getActivity(getApplicationContext(), 0, intent, 0);
            //Call the method sendTextMessage() to send SMS messages with a PendingIntent
            SmsManager sms = SmsManager.getDefault();
            //Send message
            sms.sendTextMessage(college_phoneNumber, null, enrolment_form, pi, null);
            Toast.makeText(Enrol_Activity.this, "Message Sent", Toast.LENGTH_SHORT).show();
        }
    }
});
```

_Open AndroidManifest.xml file and add a permission, *android.permission.SEND_SMS*, to your app:

```
<!-- Insert the permission for SEND_SMS -->
<uses-permission android:name="android.permission.SEND_SMS" />
```



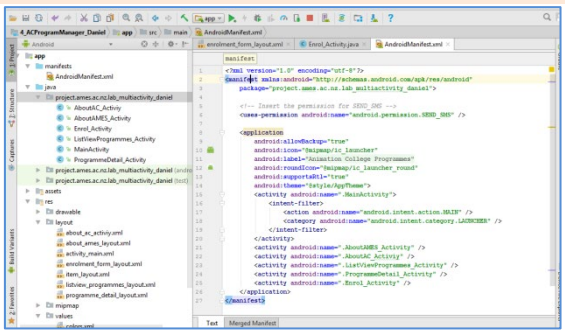
+ Compile and run the app on AVD (Nexus 5X) to see the enrolment form is sent via SMS to college phone number:



So far, we've got 6 Activities in the app. Open AndroidManifest.xml file to see them all:

1. MainActivity as a launcher
2. AboutAMES_Activity;
3. AboutAC_Activity;
4. ListViewProgrammes_Acitivty;
5. ProgrammeDetail_Activityl
6. Enrol_Activity;

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="project.ames.ac.nz.lab_multiactivity_daniel">

    <!-- Insert the permission for SEND_SMS -->
    <uses-permission android:name="android.permission.SEND_SMS" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".AboutAMES_Activity" />
        <activity android:name=".AboutAC_Activiy" />
        <activity android:name=".ListViewProgrammes_Activity" />
        <activity android:name=".ProgrammeDetail_Activity" />
        <activity android:name=".Enrol_Activity" />
    </application>
</manifest>
```



Exercise 1:

There haven't been 2 fields in Enrolment Form:

1. **Address(*)**
2. **Last Secondary/High School**

It's your turn to add those 2 fields to the enrolment form;

Exercise 2:

When users click "Submit" button, the enrolment form is sent via SMS. Your turn is to add one more option for users to send Enrolment form via Email.

PART 7: Add the seventh Activity "ContactUs_Activity"

Add a new Activity, called "**ContactUs_Activity**", to the project.

When users click "Contact Us" icon, the app will open a screen displaying college phone, college address and its location on Google map. Link to "Contact us": https://www.ames.ac.nz/contact-us/

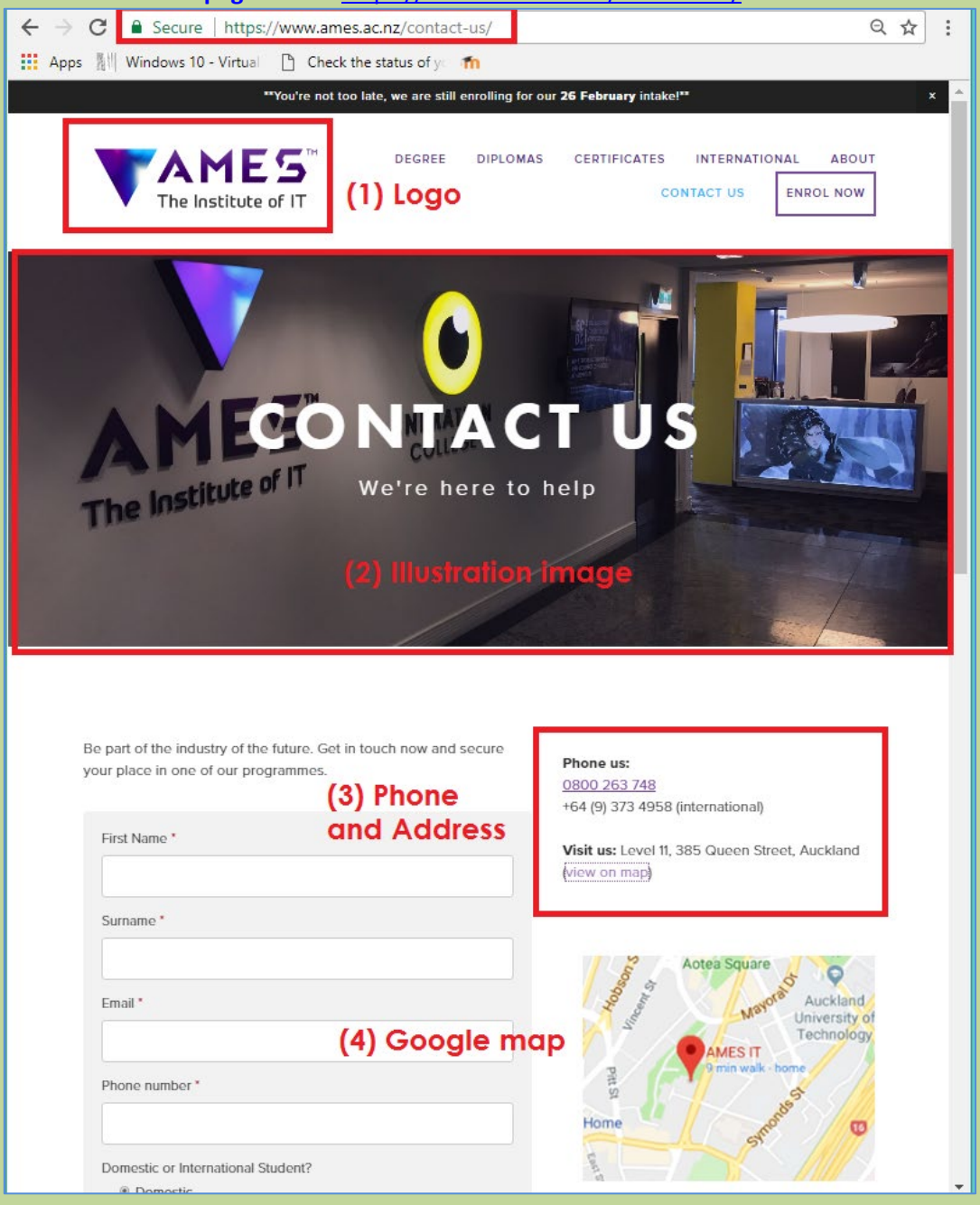
Phone us: 0800 263 748

+64 (9) 373 4958 (international)

Visit us: Level 11, 385 Queen Street, Auckland

View college on Google map:

Open and see **the "Contact Us" page** at link: https://www.ames.ac.nz/contact-us/



Step 1: Add a new Activity to the project

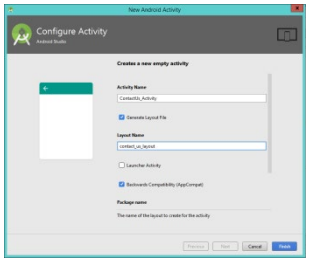
+ Add a new Activity, called "ContactUs_Activity" and its associated "contact_us_layout"

*_Right click on the folder containing **MainActivity** → "**New**" → "**Activity**" → "**Empty Activity**" → a "Configure Acitivy" window is appeared:*

*_Enter the "Activity Name": **ContactUs_Activity***

*_Enter the "Layout Name": **contact_us_layout***

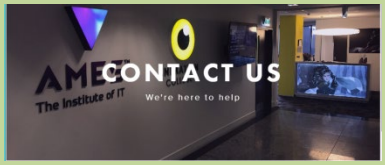
*_Click "**Finish**"*



Step 2: Analyze & design the contact_us_layout.xml

In the "Contact Us" layout, we will show 3 information of AC:

1. **Illustration image:**

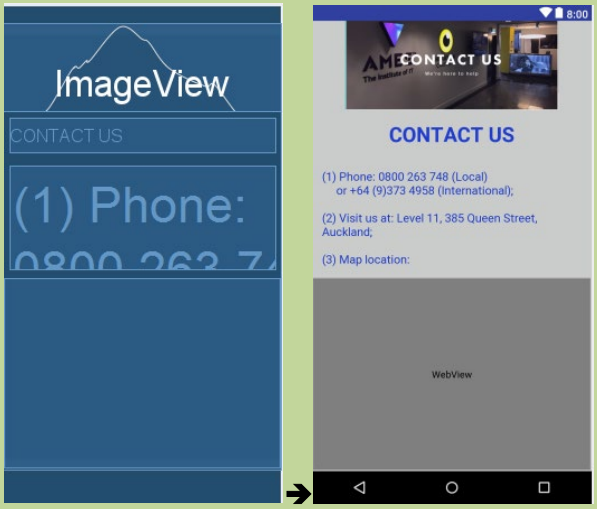


2. **Title "CONTACT US"**
3. **Info of Phone and Address:**

Phone us: 0800 263 748
+64 (9) 373 4958 (international)
Visit us: Level 11, 385 Queen Street, Auckland

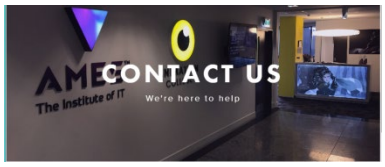
4. **Google map:**

⇨ **The layout of contact_us_layout looks like:**



Step 3: Prepare resources (images & texts) for the contact_us_layout.xml:

+ First of all, copy below image into the "drawable" folder:



illlustration_image_contact_us.png

+ Then we declare 1 String variable ("contact_us") in strings.xml file:

Open strings.xml file in "values" folder and add 1 variables "contact_us":

```
<!--Add "contact_us" variable-->
    <string name="contact_us">1) Phone: 0800 263 748 (Local) \n     or +64 (9)373 4958
(International);
\n(2) Visit us at: Level 11, 385 Queen Street, Auckland;
\n(3) Map location: </string>
```

Step 4: Design Graphic User Interface (GUI): contact_us_layout.xml

+ Now design the contact_us_layout.xml layout that contains:

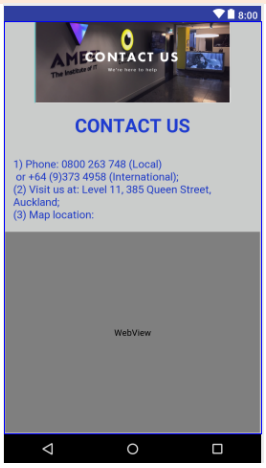
- **1 ImageView: display illustration image;**
- **1 TextView: display title "CONTACT US";**
- **1 TextView : display phone and address;**
- **1 WebView: display the college address on Google Map;**

+ Open and edit the contact_us_layout.xml as below:

contact_ac_layout.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:background="#cacccb"
    android:orientation="vertical"
    tools:context=".ContactUs_Activity">

    <!--ImageView: display ames logo-->
    <ImageView
        android:id="@+id/ames_logo"
        android:layout_width="match_parent"
        android:layout_height="130dp"
        android:layout_gravity="center"
        android:background="#FFFFFFFF"
        android:contentDescription="college logo"
        android:gravity="center"
        android:scaleType="fitCenter"
        android:src="@drawable/ames_logo" />

    <!--TextView: display "title"-->
    <TextView
        android:id="@+id/title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:gravity="center"
        android:padding="5dp"
        android:text="CONTACT US"
        android:textColor="#1f3ccf"
        android:textSize="30sp"
        android:textStyle="bold" />

    <!--TextView: contact detail-->
    <TextView
        android:id="@+id/aboutAMESIT"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:gravity="left"
        android:padding="5dp"
        android:text="@string/contact_us"
```

```
        android:textColor="#1b37c6"
        android:textSize="17sp"
        android:textStyle="normal" />

    <!--WebView: display college address on Google Map-->
    <WebView
        android:id="@+id/webView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_margin="2dp"
        android:layout_weight="1" />
</LinearLayout>
```



Step 5: Use "Intent" object to jump from MainActivity to ContactUs_Activity when users touch "Contact us" icon on mains screen:

In this part, we will add an image icon (**contact_us.png**) to **activity_main.xml layout**. When users touchs this icon, the app will open **AboutUs_Activity** to display its associated **about_us_layout.xml**.

+ Open activity_main.xml and add "about_ames_icon" icon below on the right hand side of "Apply" icon:

_First, copy the contact_us_icon.png image to "drawable" folder:

Image: contact_us_icon.png

_Then, add an ImageView to main layout (activity_main.xml): at the bottomof current codes

```
<!--ImageView: display contact_us icon-->
<ImageView
    android:id="@+id/contact_us"
    android:layout_width="110dp"
    android:layout_height="100dp"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/viewAllProgrammes"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:background="@drawable/contactus_icon"
    android:contentDescription="contact us icon" />
```

+ Open MainActivity.java file and add the below codes to implement the function:

When users touch "Contact Us" icon, the app will move to ContactUs_Activity to display the detailed information of AMES College (address, phone, and google map):

Read carefully all the comments so that you can understand the purpose of each java syntax and java code line:

_Declare a variable "contactUs" right after the class declaration:

```
//Contact us - 1: Declare "contactUs" variable as ImageView type
private ImageView contactUs;
```

_Inside onCreate() method, add code to find reference for "contactUs":

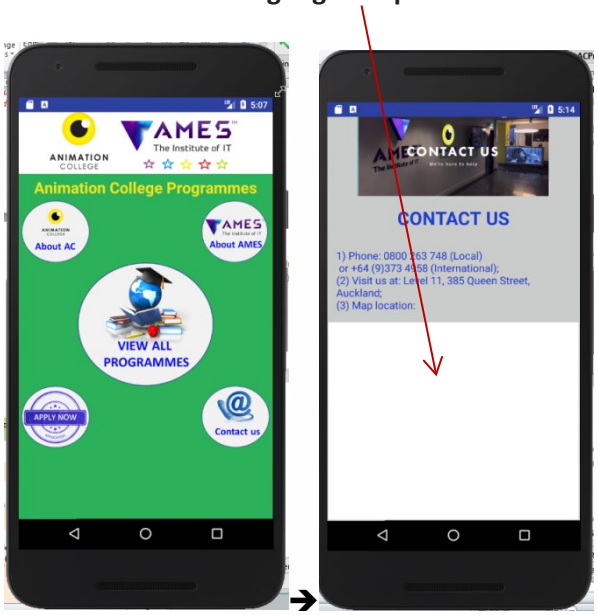
```
////////////////////////////////////////////////////////////////////////////////////////////////
//Apply - 2: Find references and do casting for "contactUs"
contactUs = (ImageView) findViewById(R.id.contact_us);
```

_Inside onCreate() method, set click listener for "contactUs" imageView. Inside onCick() method of listener, create an Intent object to transition from this activity (MainActivity) to activity (ContactUs_Activity):

```
////////////////////////////////////////////////////////////////////////////////////////////////
//Apply - 2: Find references and do casting for "contactUs"
contactUs = (ImageView) findViewById(R.id.contact_us);

////////////////////////////////////////////////////////////////////////////////////////////////
//View All Programmes - 3: Set click listener for the "contactUs" ImageView
contactUs.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Create an Intent object to transition from this activity (MainActivity) to
        // another activity (ContactUs_Activity)
        Intent intent = new Intent(MainActivity.this, ContactUs_Activity.class);
        startActivity(intent);//Start Activity
    }
});
```

_Run the app on AVD (Nexus 5X) the see the result:

If you run your app now, you will see that the **contact_us_layout.xml** layout will not display the **college address on google map**.



Step 6: Implement the Webview to display map

If you run your app now, you will see that the contact_us_layout.xml layout will not display the college address on google map.

+ Open the ContactUs_Activity.java file, and edit it as below:

ContactUs_Activity.java:

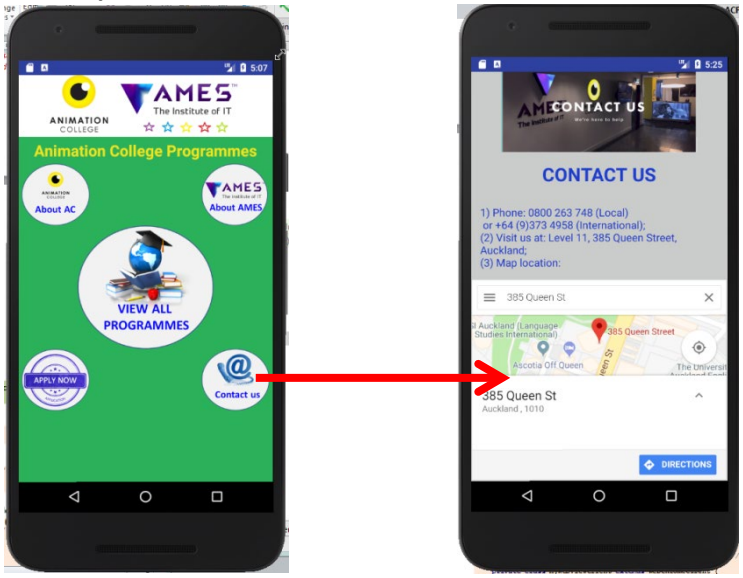
```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class ContactUs_Activity extends AppCompatActivity {
    ////////////////////////////////////////////////////////////////////////////////////////////////
    //Declare variables
    private WebView webview;

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

        //Find reference and do casting for the webview
        webview = (WebView) findViewById(R.id.webView);

        //Create a variable WebSettings, called webSettings, and assign it to "myWV" setting
        WebSettings webSettings = webview.getSettings();
        ///Set javascript enabled in your WebView
        webSettings.setJavaScriptEnabled(true);
        //Set Zoom properties
        webSettings.setBuiltInZoomControls(true);
        webSettings.setSupportZoom(true);
        //setClientClient() method is used to display website and other url link within
        //Webview, not open a new browser window.
        //If I click on a link in the webview, the resulting page will also be
        //loaded into the same webview.
        webview.setWebViewClient(new WebViewClient());
        //Set WebChromeClient to get the progress of page loading
        webview.setWebChromeClient(new MyWebViewClient());

        //Load the google map address of the college
        webview.loadUrl("https://www.google.co.nz/maps/place/385+Queen+St,+Auckland,+1010/@-
36.8562355,174.7616089,18.75z/data=!4m5!3m4!1s0x6d0d47e8bbf193f3:0x7d2cc8e3ccf8ff98!8m2!3d-
36.8560585!4d174.7619698");
    }

    ////////////////////////////////////////////////////////////////////////////////////////////////
    // Create MyWebViewClient class here
    private class MyWebViewClient extends WebChromeClient {
        @Override
        public void onProgressChanged(WebView view, int newProgress) {
            super.onProgressChanged(view, newProgress);
        }
    }
}
```

+ Open Android Manifest.xml file and add permission to the app:

WebView required **INTERNET permission**, add below into **AndroidManifest.xml**:

```
<!-- Insert the permission for access Internet -->
<uses-permission android:name="android.permission.INTERNET" />
```

+ Run the app on AVD (Nexus 5X) the see the result:



End of Lab!

YOU DID A GOOG JOB.

Congratulation!