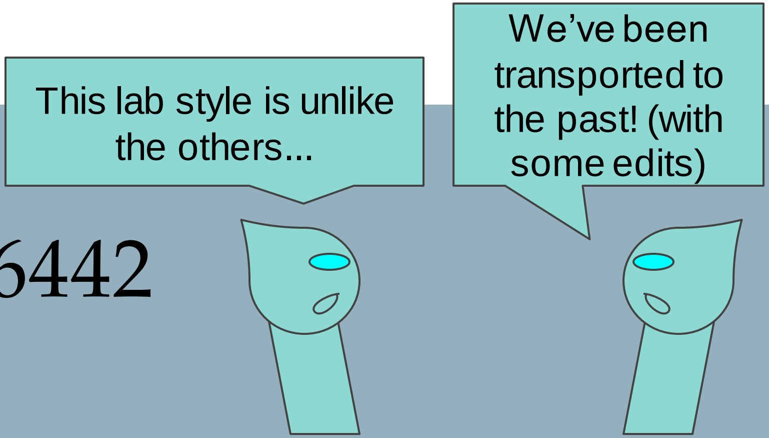
Australian National University

Lab 6 - Android Part 1: A Simple Bookmark Application

COMP2100/6442


This lab style is unlike the others...
We've been transported to the past! (with some edits)

Important reminder: Carefully read the Submission Guidelines on slide 16!

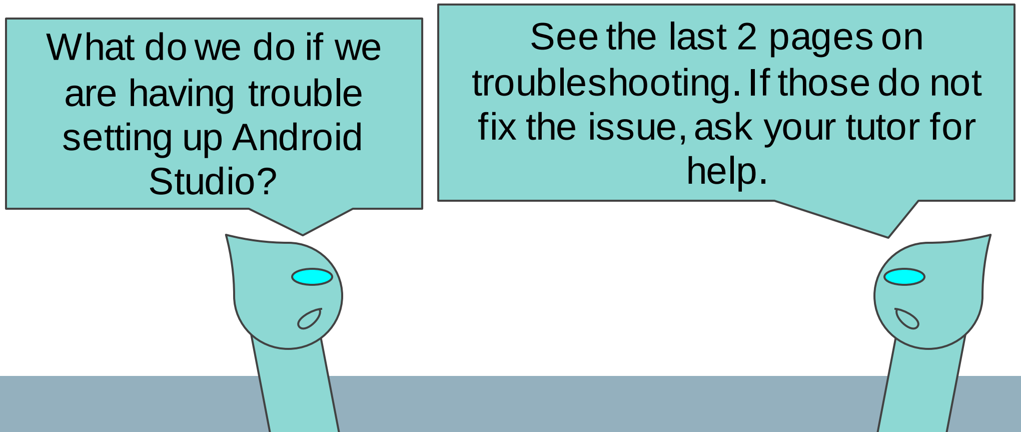
Introduction and Agenda

In this lab you will be making a simple bookmarking application!

Task 1: Create a responsive layout for your application.

Task 2: Add the functionality to add, list and view bookmarks.

Note that a lot of the content is covered in the Android Lecture, which may be useful to (re-)watch if you are unsure about whether you are doing something correctly or not.

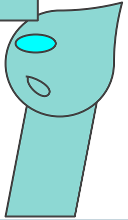

What do we do if we are having trouble setting up Android Studio?
See the last 2 pages on troubleshooting. If those do not fix the issue, ask your tutor for help.

Marks for this week

- Both tasks for this simple Bookmark Android App are worth 1 mark.
- Task 1 is worth 1 mark
- Task 2 is worth 1 mark
- This lab contains assessable items! This lab will be assessed DURING YOUR LAB SESSION.

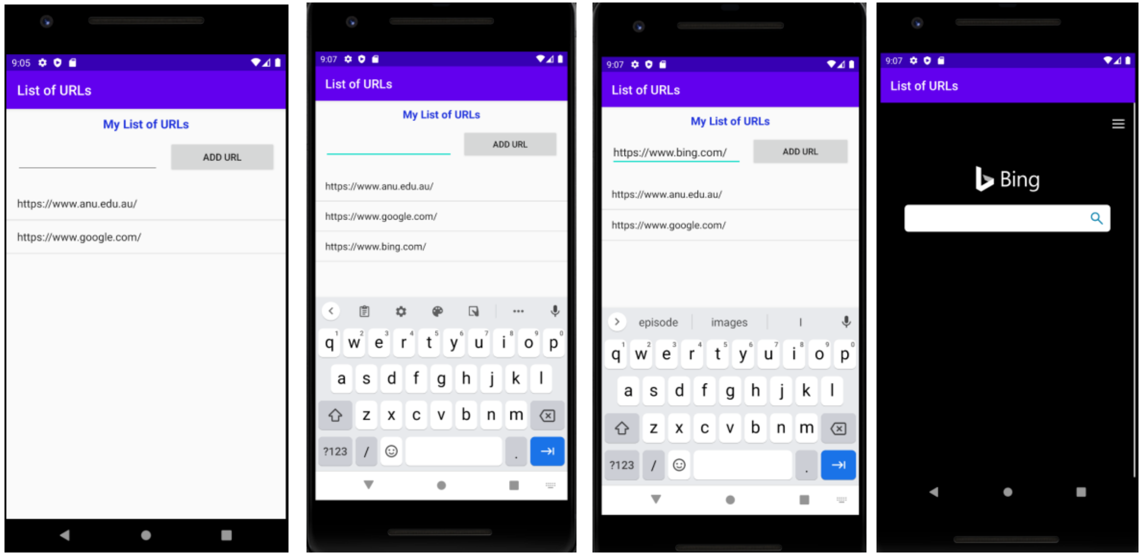
- Submission Guidelines
  - Slide 16 contains information about the submission
  - **Read it carefully to avoid losing marks!**

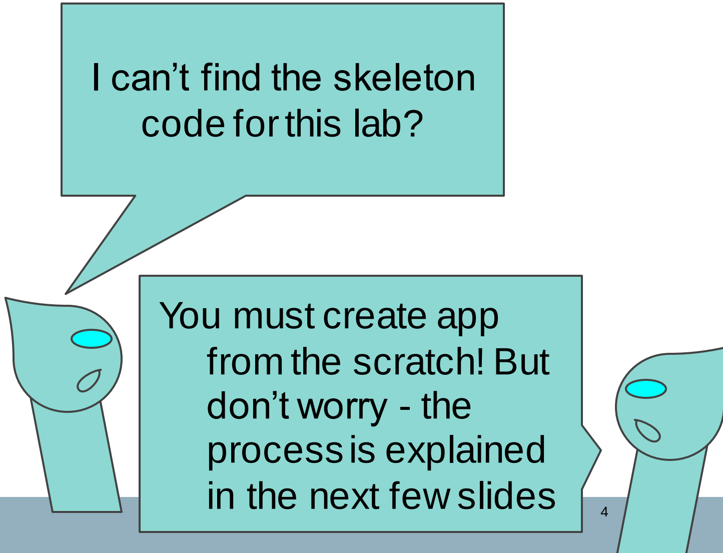
These slides act as a short ‘introduction’ to android studio basics so make sure to go through them carefully!



Task – Bookmark App

The main objective of this lab is to create a simple bookmark app that adds a URL to a ListView and when the user taps on one of the links, a new activity loads that URL into a WebView.

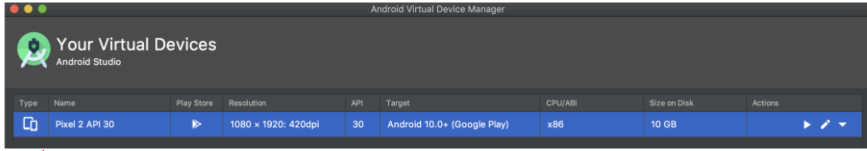



I can't find the skeleton code for this lab?
You must create app from the scratch! But don't worry - the process is explained in the next few slides

Project Setup

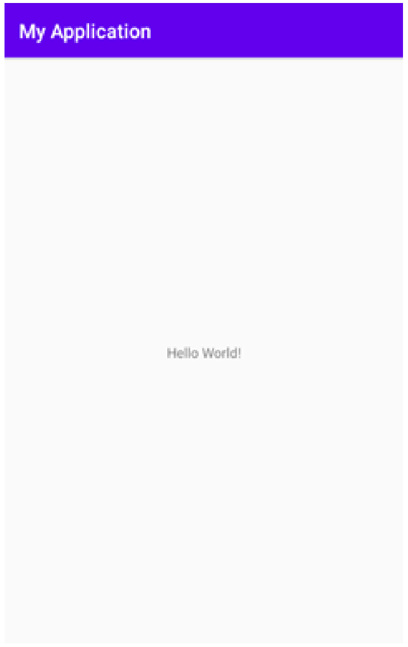
Before you start your project, let's create a new Android Studio project, with the **EmptyActivity** template.

- Ensure that the language is set to Java.
- You may have to wait a while for Gradle to configure your project. Ensure that you are connected to the internet while this is occurring. It should be done when you see "Gradle sync finished in …" in the bottom left of the window.
- Click the green play button to run your project. Note that you will need to create a new virtual device if you haven't installed one yet on your personal machine (use the default on the lab computers). If so, go to Tools > Device Manager > Create Device (ensure it is in the virtual tab), select a device you would like to emulate (ensure it is in the phone tab), click Next and download a system image with **API level 27+**. Once this is done click Next and Finish.
- If you encounter an error while installing the system image, see the troubleshooting slide (last slides). You should now be able to select your virtual device (if it is not already selected) and run it. Once the emulator has finished loading you should see a screen that looks like this:


Android Virtual Device Manager

Your Virtual Devices
Android Studio

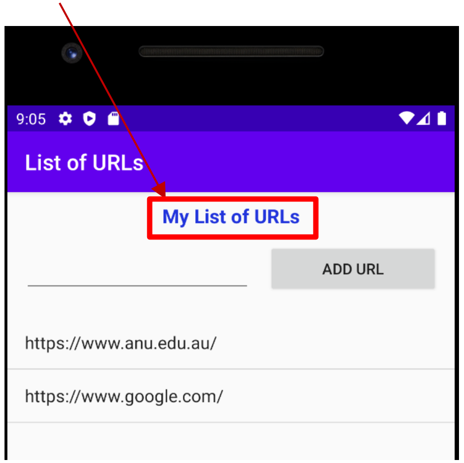
Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Pixel 2 API 30		1080 × 1920: 420dpi	30	Android 10.0+ (Google Play)	x86	10 GB	

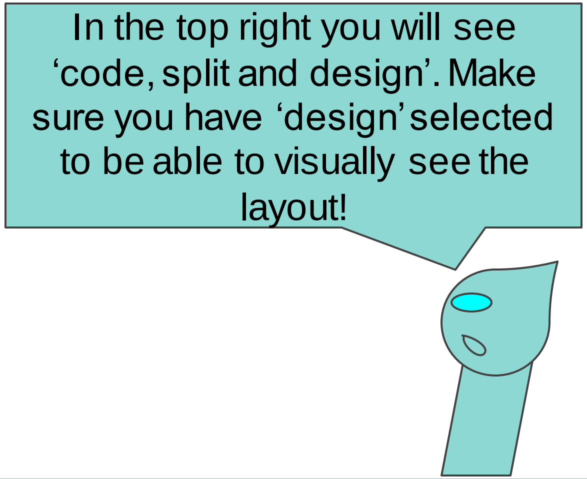


Task 1 – Responsive Layout (1 mark)

If you go back to your project, there should be a file open called activity_main.xml (if not, you can find it under app > res > layout). This will show a preview of what the activity will look like on a device.

Click on the Hello World! TextView. You will be able to see its attributes in the right panel. At the top, set its id to title_bookmark and set the title of your list of URLs (Common Attribute -> text).




In the top right you will see 'code, split and design'. Make sure you have 'design' selected to be able to visually see the layout!

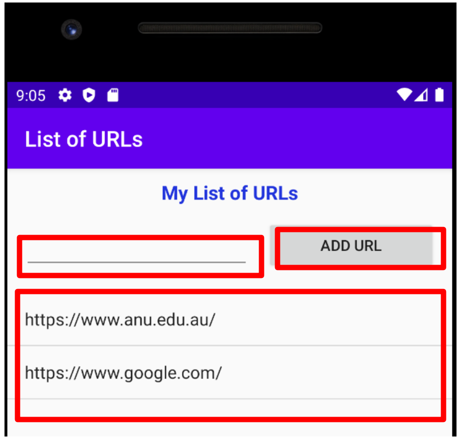
Task 1 – Constraint Layout (1 mark)

Now, add the following views: PlainText (which we will be referring to as EditText), Button and ListView.

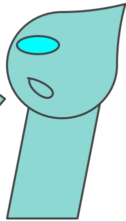
Use the Constraint Widget (for each view) in the right panel to specify the position of each view.

- The TextView must be **centered** horizontally and **15dp** from the top parent.
- The EditText must be **5dp** from the left parent and 60**dp** from the top parent.
- The Button (ADD URL, see figure) must be vertically **aligned** with the EditText , and horizontally: **10dp** from the EditText and **10dp** from the parent.
- The ListView must be placed below the EditText and Button (1**0dp** from the bottom of those views as well as match the bottom parent (**0dp**).

Hint: The "parent" in this context is referring to the parent view – ie. the view that activity_main sits within.



PlainText in the code of your layout are called 'EditText'. Unlike TextView these allow you to edit the text inside the element.



Australian National University

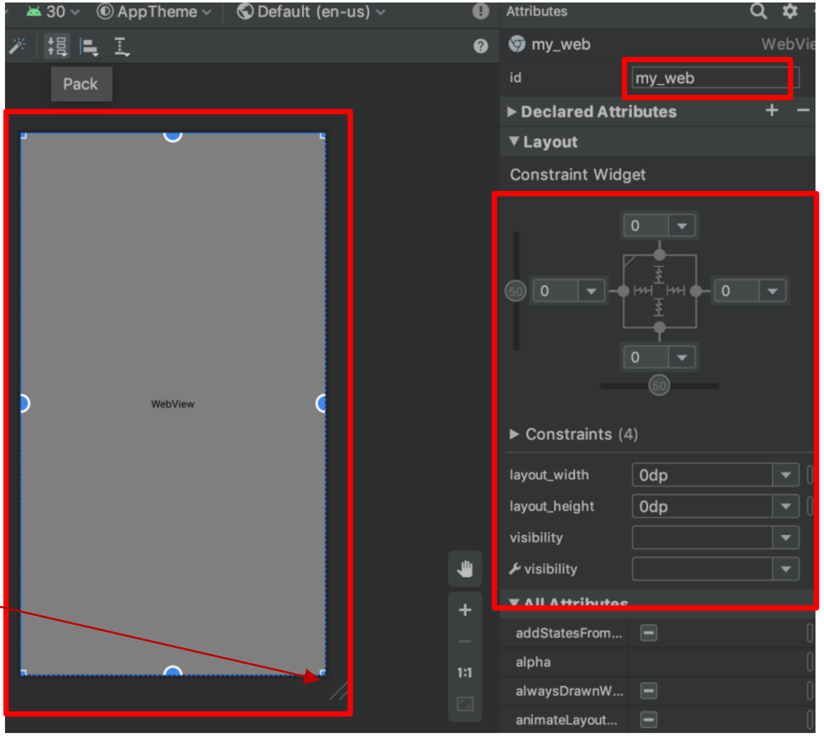
Task 1 – Constraint Layout (1 mark)

Now, create a new activity (name it: ActivityWeb, it will generate two files: activity_web.xml and ActivityWeb.java), and add a WebView.

The WebView must match parent screen (full width).

To receive marks in this task you need to:

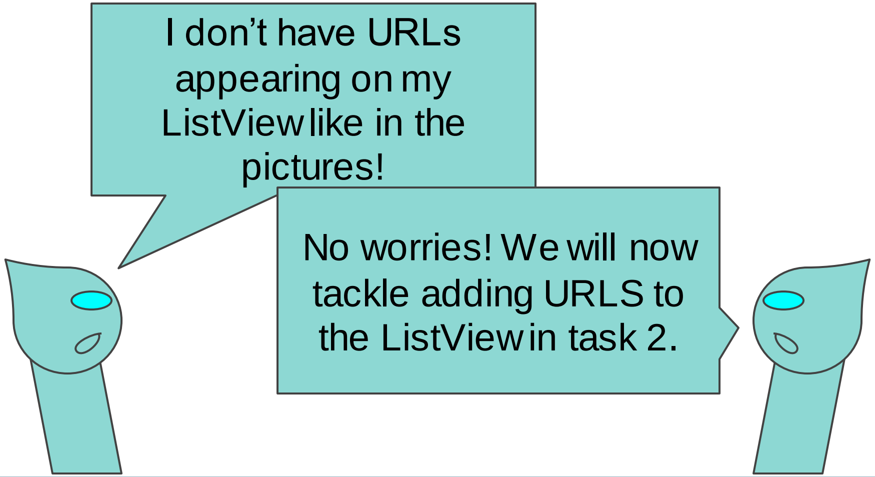
- Create an intent to go from the main activity to the activity containing the WebView. The intent must be associated to a button or to a ListView item.
- All items must be sensibly positioned (no overlaps or extreme misalignment).
- Note that we will ask you to change the screen size of your app (ask your tutor to demonstrate it).



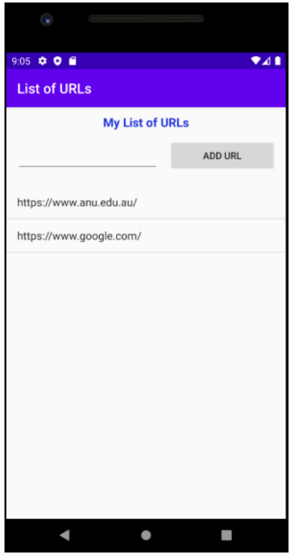
id: my_web

layout_width	0dp
layout_height	0dp
visibility	
visibility	

Task 1 Complete


I don't have URLs appearing on my ListView like in the pictures!
No worries! We will now tackle adding URLS to the ListView in task 2.

Our MainActivity
(yours may look different):



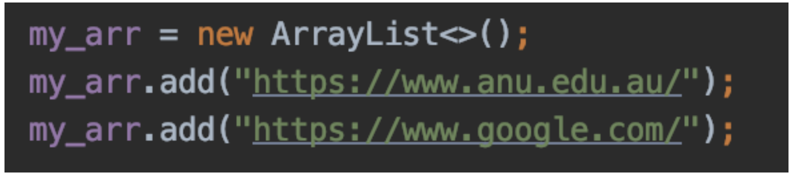
https://www.anu.edu.au/
https://www.google.com/

Task 2 – Bookmark App (1 mark)

Now, you need to do the following in your MainActivity class:

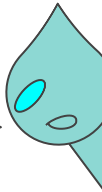
- Capture your button from layout (tip: use findViewById and do not forget to cast your view to Button).
- Set an OnClickListener to the button. This listener should get the content written on the EditText view and insert into the ListView.
    - Tips: Capture the EditText view using findViewById, add the content of the EditText view to an array (as shown during the lecture), clear the EditText (tip: use the method setText and set it to “”).
    - Finally, you must notify the ArrayAdapter that the array was updated. For this, simply use the command: name_of_your_ArrayAdapter.notifyDataSetChanged();
- After you capture the ListView from layout, create an array and add a few URLs to it.

See example:



```
my_arr = new ArrayList<>();
my_arr.add("https://www.anu.edu.au/");
my_arr.add("https://www.google.com/");
```

Buttons also have an ‘onClick’ attribute which you can look into!



Task 2 – Bookmark App (1 mark)

- To link the Array you created and ListView, you need to create an ArrayAdapter. Something similar to the code below:

```
ArrayAdapter aa = new ArrayAdapter( … );
```

- The layout resource is (one of the ArrayAdapter's parameters):

```
android.R.layout.simple_list_item_1
```

-

- The simple_list_item_1 is a built-in XML layout for presenting a list of text items.
- The next step is to create a listener to the ListView. When the user clicks on one of the items of the list view, s/he must go to the next activity (WebView) and load the URL associated to clicked item.

```
your_listview.setOnItemClickListener(somelistener);
your_listview.setAdapter(aa);
```

Australian National University

Task 2 – Bookmark App (1 mark)

- Note that we use the setOnItemClickListener, it will allow you to retrieve the item clicked from the array.
- To create this listener you have to:
  - Create an intent to go from the current activity to the activity containing the WebView.
  - Add some data to the intent to send and make it available to the next activity. See incomplete example below:

```
Intent intent = new Intent(?, ?);
intent.putExtra("URL", your_array.get(i).toString());
```

  - Do not forget to: startActivity(…).

Variable name associated to the String. This variable name will be used in the target activity to access the string assigned to it.

The OnItemClickListener must receive the position of the clicked item (variable i). The variable URL will hold this string.

Task 2 – Bookmark App (1 mark)

- In the Activity containing the WebView, you must:
  - Get the intent and the data sent from the main activity. How do you do this? See below, you must use the command getIntent() and then use the method getStringExtra to access the string assigned to the URL variable created in the previous activity.

```
Intent intent = getIntent();
String URL = intent.getStringExtra("URL");
```

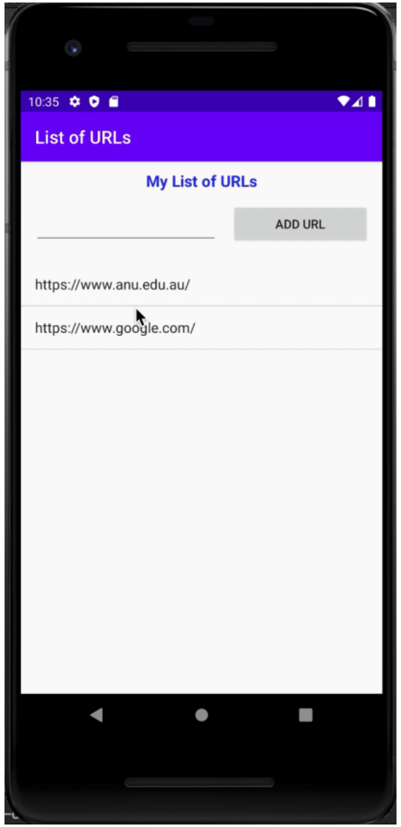
  - Now, you should create a WebView (tip: use the findViewById) and use the method loadUrl to load the "URL".
  - You must check if the URL is valid (if it is not NULL or empty (""))
  - If the URL is NULL or empty, a toast message **must** be displayed ("URL is invalid").

*Note that a few Websites may open a new browser (this is totally fine! No worries about that!).

Task 2 – Bookmark App (1 mark)

To receive marks in this task your app should add an item to the listview, and each clicked item should redirect to a new activity containing a WebView loading the corresponding URL. If the URL is NULL or empty, a toast message must be displayed.

The lab video shows what your app should do in a demo (see ‘video’ directory for this lab on Gitlab).

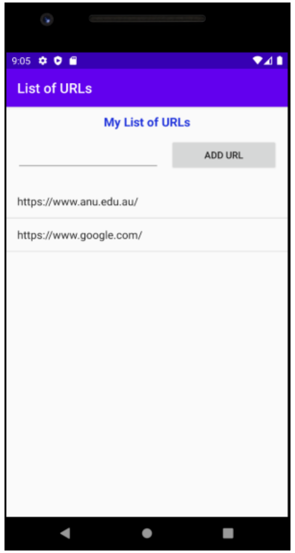


Australian National University

Task 2 Complete

Our Application
(yours may look different):

Remember to provide a toast when clicking on an empty URL!

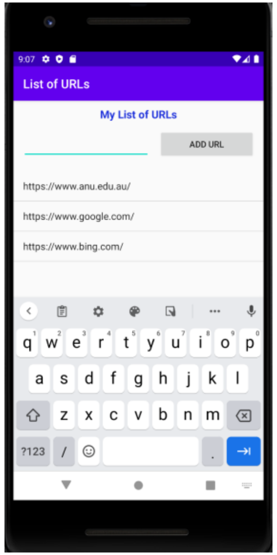

List of URLs

My List of URLs

ADD URL

https://www.anu.edu.au/

https://www.google.com/


List of URLs

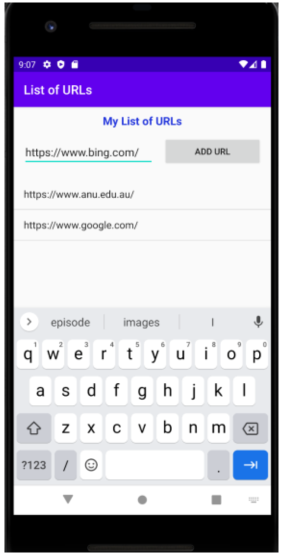
My List of URLs

ADD URL

https://www.anu.edu.au/

https://www.google.com/

https://www.bing.com/


List of URLs

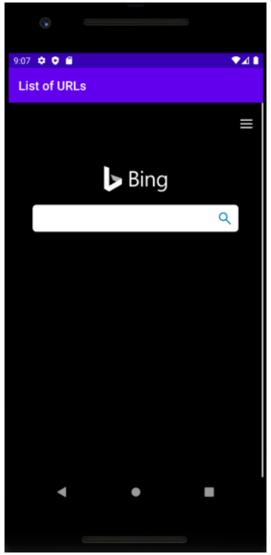
My List of URLs

https://www.bing.com/

ADD URL

https://www.anu.edu.au/

https://www.google.com/


List of URLs

Bing

Submission Guidelines

- **Assignment deadline: see the deadline on Wattle (always!)**
- Submission mode: presentation + Wattle (Lab Android)
- **To be marked you MUST show your App running to your tutor during your lab session. Remember you must submit your files to Wattle to be assessed.**
- The version of your files you present should be exactly the same as the ones submitted to Wattle
- **After submitting your files to Wattle (check the deadline on Wattle), you can present your app to your tutor in Week 7 or Week 8 during your scheduled lab session.**
- Since this lab has limited complexity, we expect students to be able to finish and present their work within the week 7 or week 8 lab sessions.

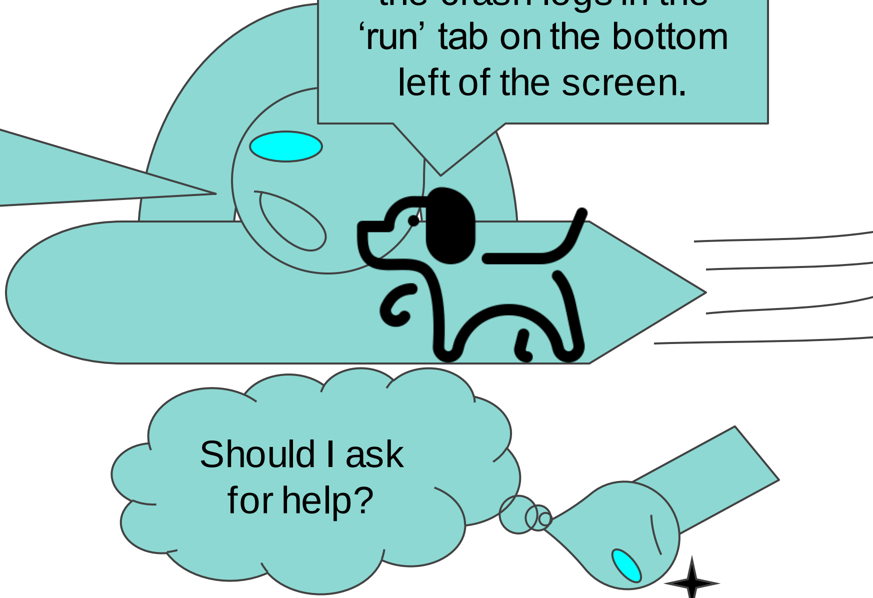
Submission format and information (IMPORTANT):

- **Upload the files MainActivity.java, ActivityWeb.java, activity_main.xml and activity_web.xml** to Wattle
- The answers will be marked by your tutor
- **Any** violation of the submission guidelines **will result in zero marks**
- Reference for this lab: see Android Lecture and demos on Echo360

The first of two Android Studio labs!

Android Studio can be quite overwhelming. If you are having difficulties understanding or navigating the environment, you should ask your tutor for help.

If your application crashes, you can find the crash logs in the 'run' tab on the bottom left of the screen.



Troubleshooting - HAXM Manager Part 1

If you receive an error about Intel HAXM Manager (or similar) not being installed properly follow the instructions pertaining to your operating system to determine your CPU manufacturer:

Windows

Press Windows + X and click System or search for System Information in the search bar (Windows 7 and 10). Windows 8 users can press Windows + R and type msinfo32 and click OK. Determine whether the CPU is Intel or AMD by looking at the processor name.

Linux

Open a new terminal and run the lscpu command. Check the vendor ID to determine whether it is GenuineIntel (meaning your manufacturer is Intel) or AuthenticAMD (your manufacturer is AMD).

Mac OS (OSX)

If your apple device was manufactured before 2020, your CPU manufacturer is (almost certainly) Intel. If it after 2020 and you posses a M1 (or later) processor, please speak to your Tutor. As of writing this, we do not know of a solution to the issue on M processors (as they are relatively new).

Troubleshooting - HAXM Manager Part 2

Once you have determined your CPU manufacturer, follow the appropriate steps:

Intel
In Android Studio head to: Tools → SDK Manager → SDK Tools. Check 'Intel x86 Emulator Accelerator (HAXM Installer)' and click OK. Follow the installer. If you are on Windows and are still receiving errors, there may be an installer in C:\users\%Your_Username%\AppData\Local\Android\sdk\extras\intel\Hardware_Accelerated_Execution_Manager\ that you can run instead.

AMD
In Android Studio head to: Tools → SDK Manager → SDK Tools. Check 'Android Emulator Hypervisor Driver for AMD Processors (installer)' and click OK. Follow the installer.

Once you have finished either Intel or AMD steps
Make sure that Android Emulator (also in the SDK tools tab) is fully checked (not just a dash) and click OK.

If you are on Windows and still having problems, you may need to enable Virtualisation in your BIOS. Here is a video on youtube to help you (but the steps after restarting your computer will be specific to your motherboard BIOS):
https://www.youtube.com/watch?v=Y1WhS2yuF8I

If you are still experiencing issues, please let your tutor know and they will attempt to assist you with troubleshooting. Please be patient as your may have an issue that they have not experienced before and you will need to work together to solve it.