Mobile Computing

COMP5216

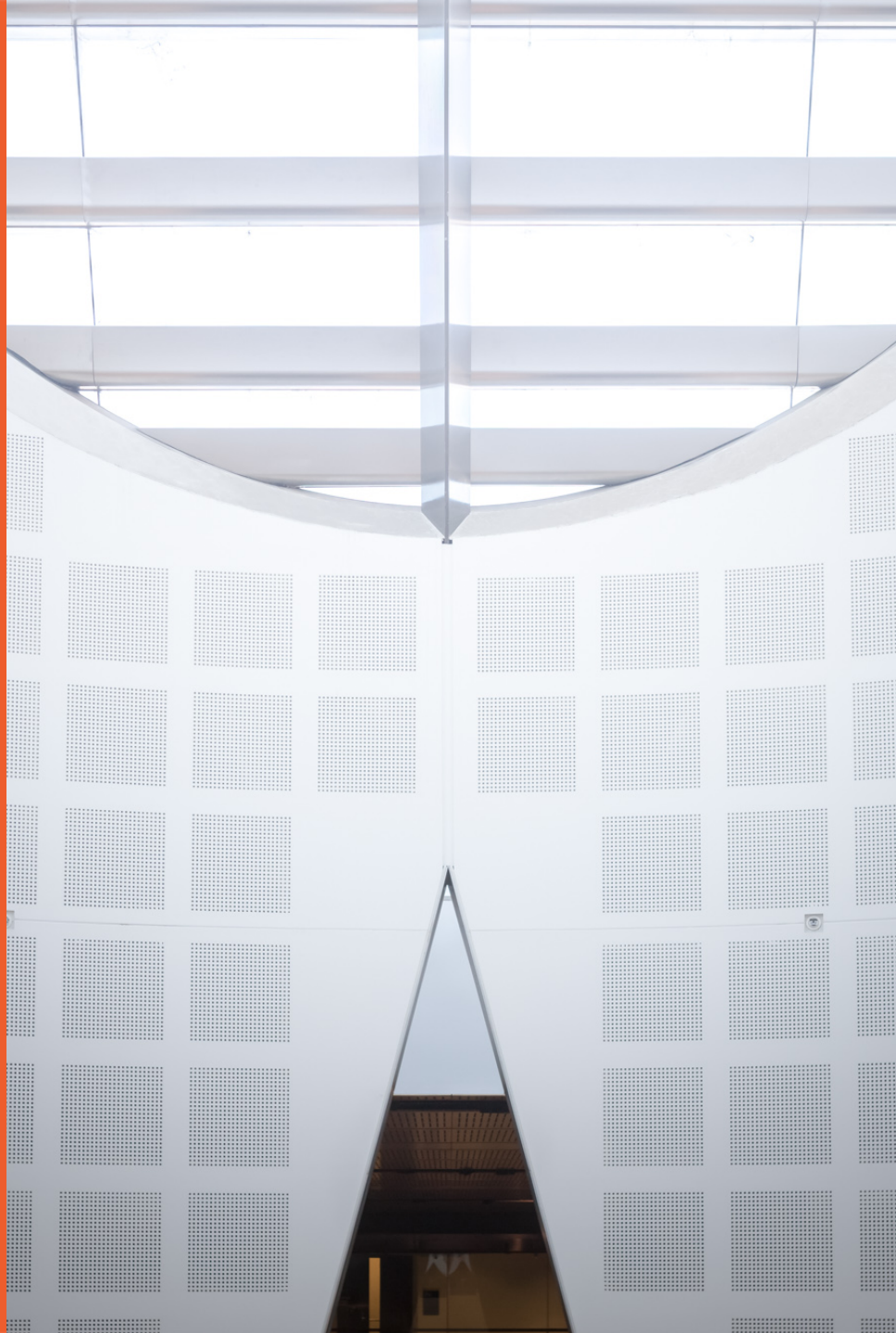
Week 02

Semester 2, 2020

Dr. Kanchana Thilakarathna

School of Computer Science

THE UNIVERSITY OF SYDNEY



Outline

- App development workflow
- How to come up with app ideas through problems ?
- Group Project
- Android Programming Basics
  - Android app development process
  - Where are the resources ?
  - Programming basics

App development workflow

Six Steps

1. Define Goals
2. Analyse Requirements
3. Design UI: sketch, wireframe or storyboard
4. Design project structure
5. Implement codes
6. Test, debug, and release

1. Design Goals

- Goals vs Ideas
  - Value

Value Propositions

- To meet the needs of customers/users
  - What value do we deliver to customers ?
  - Which one of customers' problems we are helping to solve ?
  - What bundles of products and services are we offering to each Customer Segment ?
  - Which customer needs are we satisfying ?

- In business context
  - Offerings from the product
  - What distinguishes itself from its competitors
    - Quantitative – price and efficiency
    - Qualitative – overall customer experience and outcome

2. Requirement Analyses

- How to deliver the values
  - Tasks
  - User scenarios
- How the app will function
  - Key interactions in the app
  - Secondary functions
    - Features in the first version that are “nice to have”
- Understand the field
  - Find out whether there are other apps out there doing the same thing
- Obtain insights of the problem
  - Find design inspiration for your app
  - Find information on the technical requirement for your app
  - Find out how you can market and monetize your app

Business Model Canvas

- Proposed by business theorist Alexander Osterwalder
- A strategic management and lean startup template for developing new or documenting existing business models.
- It is a visual chart with elements describing a firm's or product's value proposition, infrastructure, customers, and finances.
  - https://www.alexandercowan.com/business-model-canvas-templates/

The Business Model Canvas

Designed for:

Designed by:

On: Day Month Year

Iteration: No.

Key Partners

Key Activities

Key Resources

Value Propositions

Customer Relationshi

Channels

Customer Segments

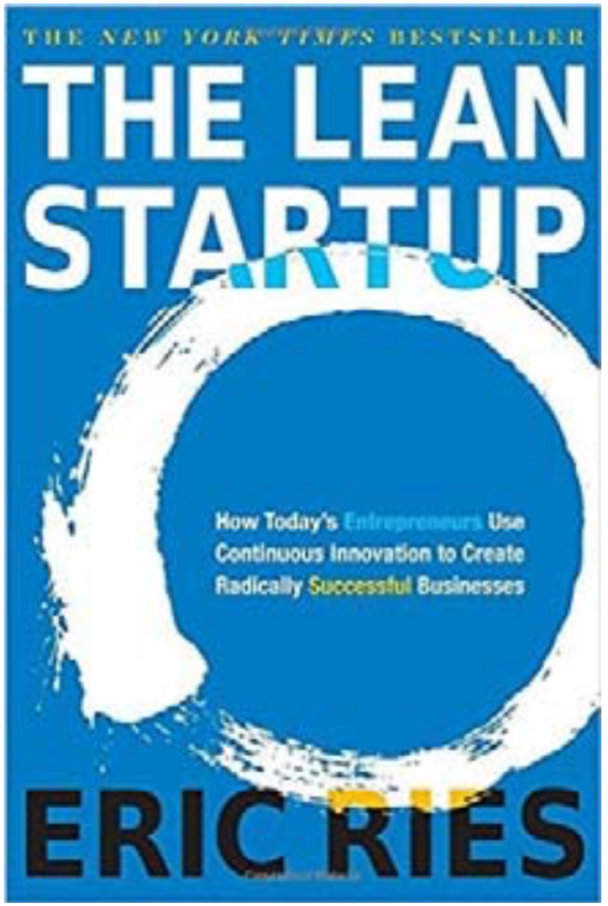
Cost Structure

Revenue Streams

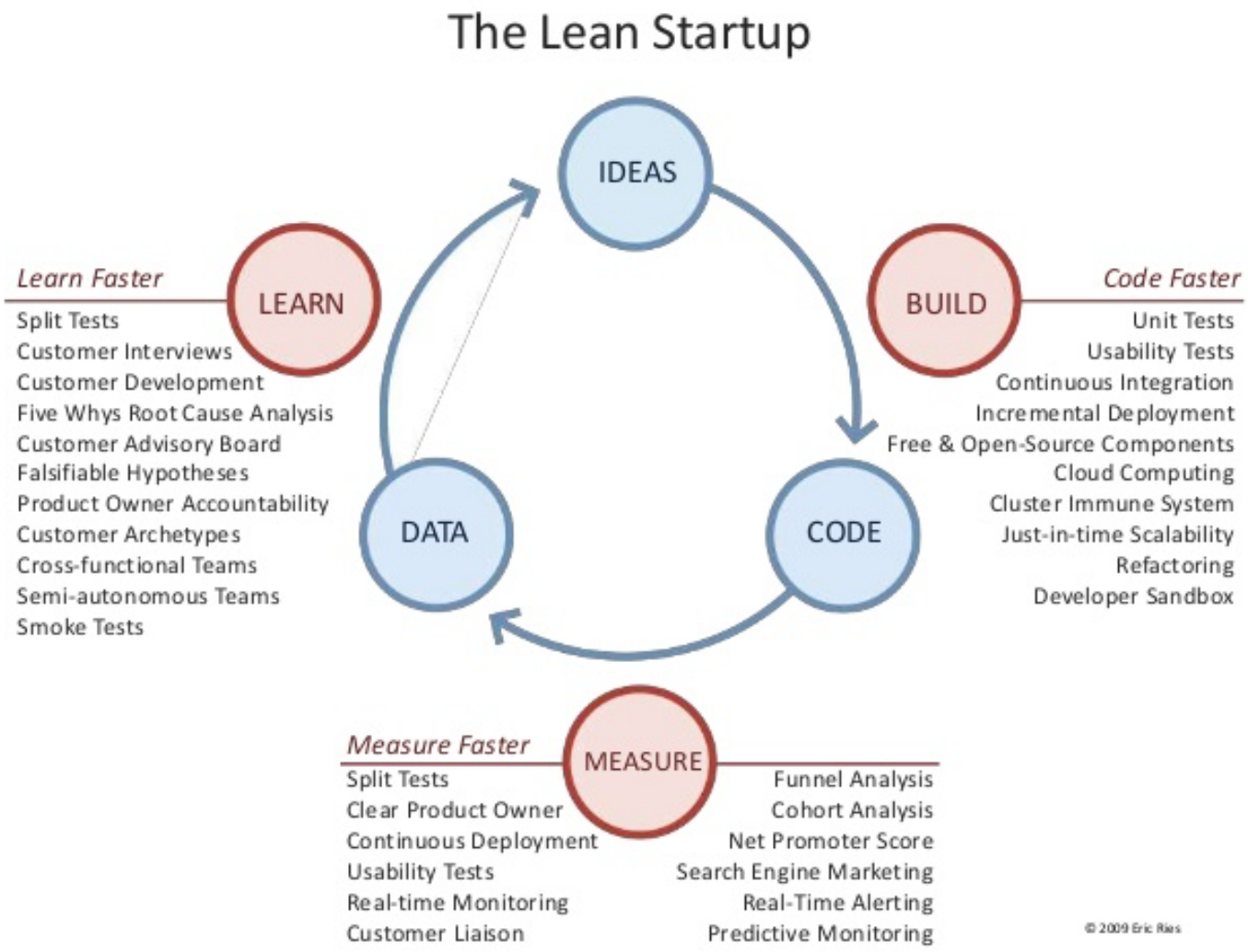
www.businessmodelgeneration.com

The Lean Principle

- **Entrepreneurs are everywhere**
  - There's never been a better time to innovate
  - Think big. Start small. Scale Fast
- Entrepreneurship is management
- Validating learning
- Innovation accounting
- **Build-Measure-Learn**



Build – Measure - Learn



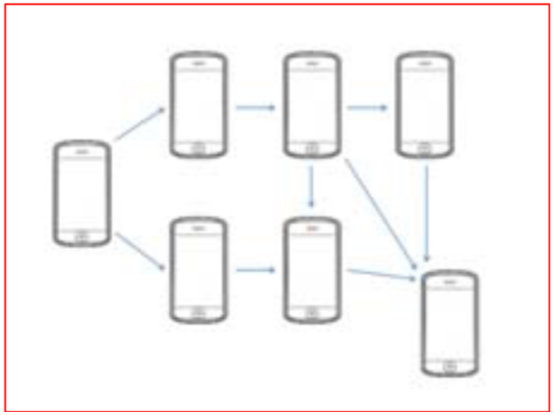
Minimum Viable Product

3. UI (User Interface)

- Platform specific UI styles
  - iOS - https://developer.apple.com/design/human-interface-guidelines/
  - Android - https://developer.android.com/design/

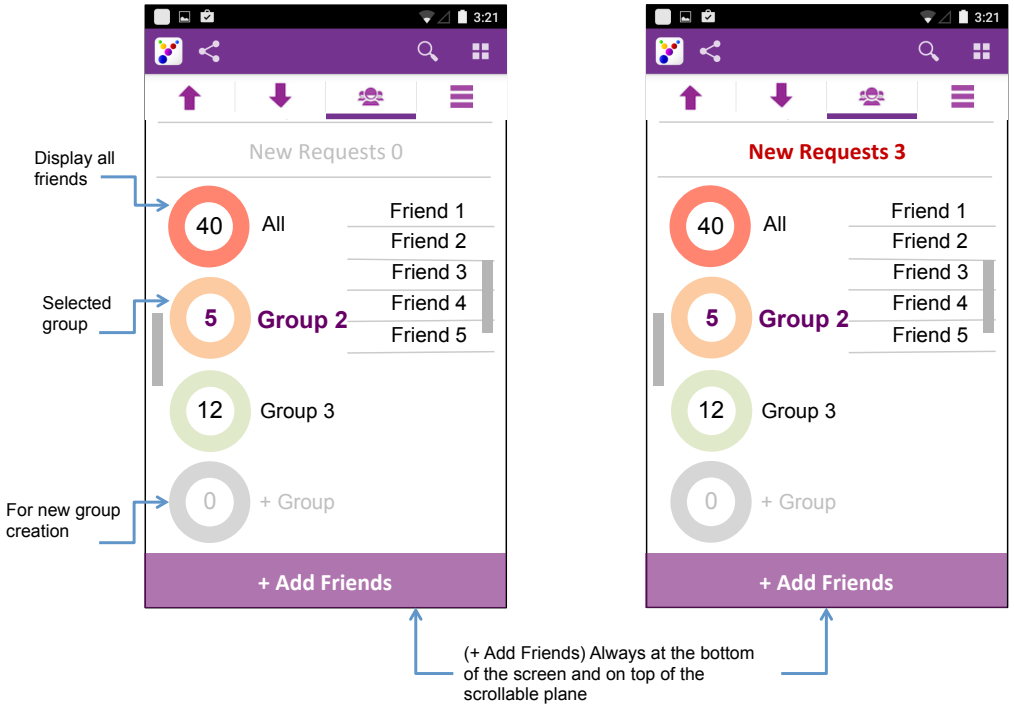
3. UI (User Interface)

- Sketching wireframes
- Tools (not free)
  - Balsamiq - https://balsamiq.com/
  - Moqups- https://moqups.com/
  - HotGloo - https://www.hotgloo.com/



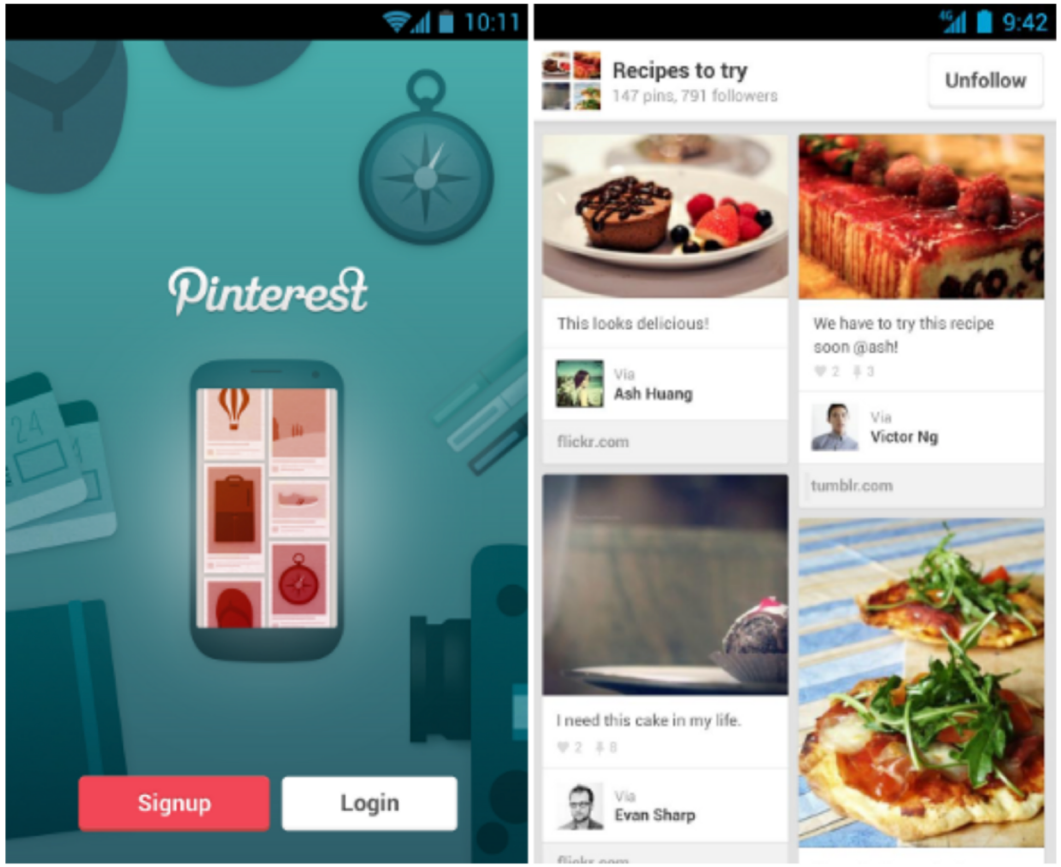
3. UI (User Interface)

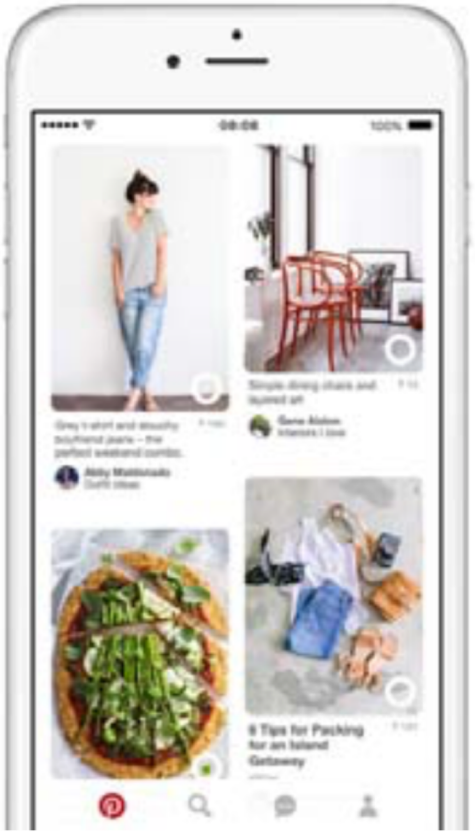
- Tools (free)
  - MS PowerPoint
  - Apple Keynote
  - Apple - https://developer.apple.com/design/resources/
  - Android - https://material.io/design/introduction/#principles



3. UI (User Interface)

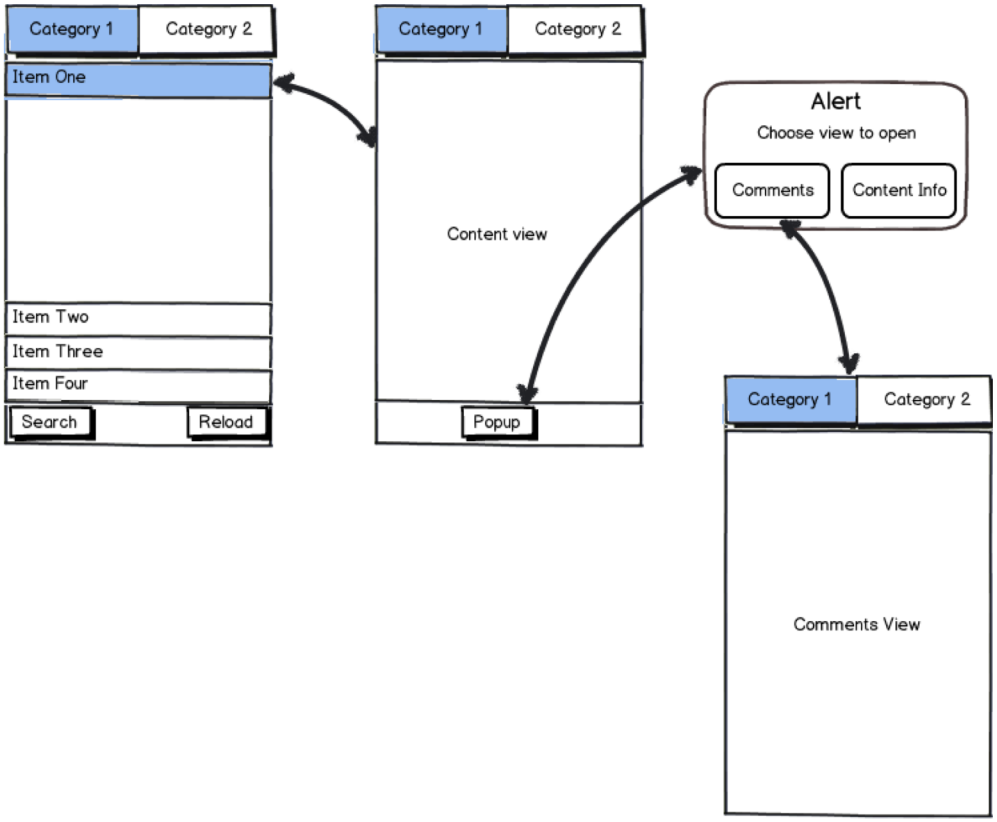
- Learn from good designs





3. UI (User Interface)

- Test your UI on the device
  - POP - https://marvelapp.com/pop/
  - Balsamiq - https://balsamiq.com/



4. Design Project Structure

- Frontend
  - UI implementation
    - MVC framework (Model-View-Controller)
    - MVP (Model-View-Presenter) and MVVM (Model-View-View-Model)
    - Avoid holistic placement of UI elements
- Backend
  - Use cloud service as much as possible
- Always think about the impact of your design on other stakeholders of the eco-systems
  - Users
  - Networks
  - …,

5. Implementing the Code

- **Mobile app development is increasingly getting easier !**
  - OS APIs
  - Web/Cloud services APIs
  - Third party SDKs
  - Third party libraries

- Develop as a team
  - 1- Front-end, 1- Back-end, 1- UI/UX, 1- Manager
  - Agile software development
  - Scrum - https://www.atlassian.com/agile/scrum

- **Take advantage from collaborative tools**
  - Bitbucket - https://www.bitbucket.org
  - Github - https://github.com
  - Slack - https://slack.com

6. Testing and Debugging

- Not only functions
- But also the feedback from users, such as how they use the app
  - Verifying value propositions
  - Taking note of their actions
  - Adapting your UI/UX to them
  - App Analytics SDKs

- **Beta version release**
  - Lets you to distribute the app to known users

- Customer development
  - Users other than your friends and family members: getting out of the building

How to find app ideas?

Ideas

- **Existing ideas**
  - Even if you find someone else working on the same thing, you're probably not too late.
  - Worrying that you're late is one of the signs of a good idea

- **Extension to existing ideas**
  - Analogy between two domains
    - Airbnb vs Bike sharing
    - Facebook vs Linkedin
  - Add an extra feature
  - Opposite thinking
    - Hospital vs Home health-care

- **Innovative ideas**

Ideas

- The way to get good ideas is not to try to think of your assignment
  - The most common mistake startups make is to solve problems no one has
  - It yields bad ideas that sound plausible enough to fool you into working on them
  - "made-up" or "sitcom" startup ideas

- Two types of successful apps
  - Entering a market with existing competitors, but armed with some secret weapon that will get them all the users.
    - Example?
  - Entering a market that looks small, but which will turn out to be big.
    - Example?

- **Look for problems, preferably problems you have yourself**

Own problems

- The general problems you have
  - Online social network
  - Something to get real-time bus timetable
  - Garage sale
  - Discussion forum
  - Something needed in your previous job
  - …
- Something at least some users who really need, not just may be one day, but want it urgently.

Where are the problems ?

- Alert to opportunities
  - Dropbox: forgetting USB sticks
- **Live in the future, then build what's missing.**

- Be open-minded
  - Turn off the filters that usually prevent you from seeing them
    - Why is your inbox overflowing? Why do you get so much email? What problems are people trying to solve by sending you email? Are there better ways to solve them? Why do you keep emails around after you've read them? Is an inbox the optimal tool for that?

- **Live in the future, then build what seems interesting.**

Inter-disciplinary Problems

- Learning about some other field, you'll probably see problems that software could solve.
  - (a) the inhabitants of that domain are not as likely as software people to have already solved their problems with software, and
  - (b) since you come into the new domain totally ignorant, you don't even know what the status quo is to take it for granted.

- Taking a class on, say, genetics; or better still, go work for a biotech company.
  - **One way to ensure you do a good job solving other people's problems is to make them your own.**

Problems

- Serve a small initial group
  - High demands → high competition
  - E.g. Apple, Google, Facebook, Microsoft, etc.
- Make the users who care about your product happy
  - First iPhone does not have copy/paste function
- Germ problems
  - Hard to tell, even experienced investors
  - Airbnb
    - Let hosts rent out space on their floors during conventions. They didn't foresee the expansion of this idea; it forced itself upon them gradually

Any ideas ?

Critical Factors of Startup Success

1. Idea
2. Leader
3. Team
4. Capital
5. Plan
6. Execution
7. Timing
8. Crisis response
9. Marketing
10. Growth

> It's not just the classes that make a university such a good place to crank oneself into the future. You're also surrounded by other people trying to do the same thing. If you work together with them on projects, you'll end up producing not just organic ideas, but organic ideas with organic founding teams—and that, empirically, is the best combination.

- https://www.entrepreneur.com/article/252813

- **These are experiences, rather than rules.**

Group Project

COMP5216

S2, 2020

THE UNIVERSITY OF SYDNEY

Group Project

- Refer to the Project Guidelines document on Canvas
- Two Phases – Proposal and Final
- Minimum feature set:
  - Graphical user interface (GUI) to effectively interact with the user.
  - At least one form of data communication using either Cellular, WiFi, Bluetooth, etc.
  - At least one technique to save network bandwidth usage, computation resource usage or device battery usage.
  - At least one method to secure the communication or data storage.

- **Come and test/discuss your idea with me !**

App development workflow

Six Steps

1. Define Goals
2. Analyse Requirements
3. Design UI: sketch, wireframe or storyboard
4. Design project structure
5. Implement codes
6. Test, debug, and release

Proposal Phase

Final Phase

Group Project submission

- Refer to the Project Guidelines document on Canvas
- Proposal Phase: Report (hard & electronic)
- Final Phase:
  - Report (hard & electronic)
  - Presentation slides
  - Video
  - Source Code
  - Presentation and Demo

Deliverables		Due Time
Proposal	Electronic submission	5:00pm, 28/09/2019 (Monday, Week 06)
Final	Electronic submission	5:00pm, 06/11/2019 (Friday, Week 10)
	Presentation & Demo	5:00pm, 09/11/2019 (Monday, Week 11)

Group enrolment

- Maximum Group size is **FIVE**
- **Enroll to groups via Canvas.**
- Pick a group number attached to the tutorial of most number of group members.

- Tutorial swap procedure.
  - First, try to swap it yourself.
  - If the system does not allow you do so, record your details here.
  - https://unisyd-my.sharepoint.com/:x:/g/personal/chamara_kattadige_sydney_edu_au/Eb4wS1Y01s9Es_pokqWGCrwB0UgFR6NEWJQgvTgDpn8aPA?rtime=6jwiy1IN2Eg

Android Programming Basics - 1

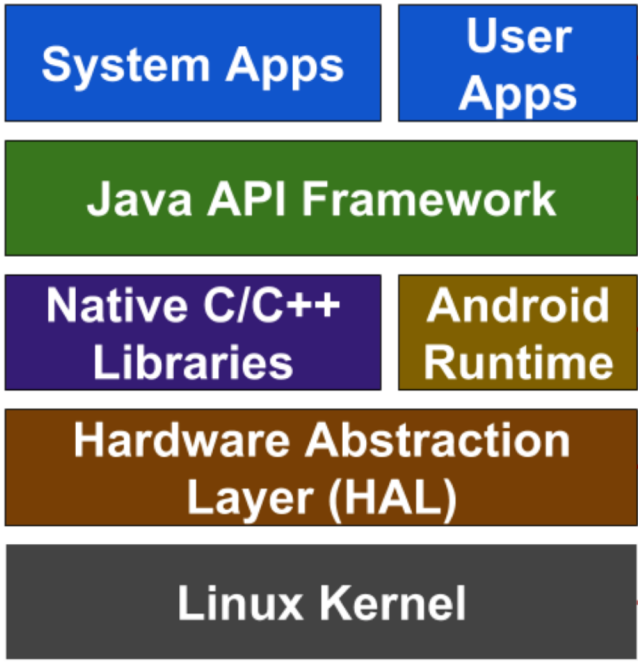
COMP5216

S2, 2020

THE UNIVERSITY OF SYDNEY

What is Android?

Major components of Android Stack

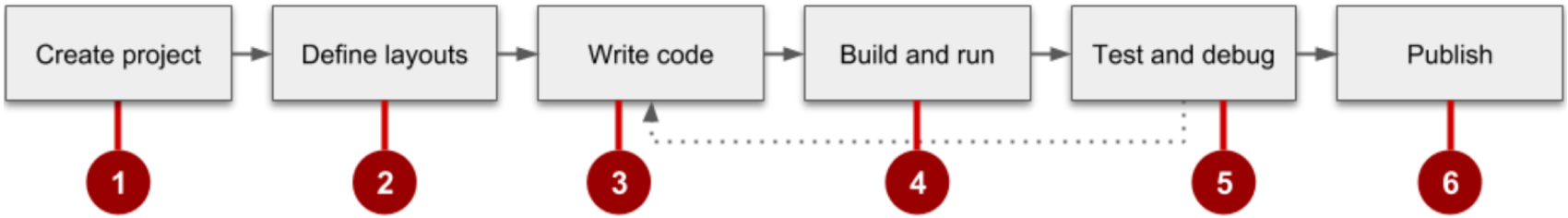


- **Applications:** Users interact with the device via the apps. Can be either first party or third party.
- **Android Framework:** Provides basic functions such as communication between apps, managing voice calls or managing app life cycles.
- **Native Libraries:** C/C++ libraries that contain instructions to the device on handling different types of data. E.g. Webkit, SSL, SQLite, and OpenGL.
- **Android Runtime:** Dalvik Virtual Machine and Core Libraries.
- **Hardware Abstraction Layer (HAL):** Converts the Java API calls to system calls that is understood by the Linux kernel.
- **Linux Kernel:** Additional modifications done by Google to make it suitable for smartphones (E.g. power management). Handles all conventional operating system functions such as process management and memory management.

It is easy to develop apps…

- IDE – Android Studio
- Languages
  - Java/Kotlin to develop apps
  - XML to describe data resources
- SDKs hide the complexity from developers
  - E.g.:
    - Java API Framework
    - Google Play Services
- You don't need to know the details of how the APIs work.

Android app development process



1. Create the project in Android Studio and choose an appropriate template.
2. Define a layout for each screen that has UI elements. You can place UI elements on the screen using the layout editor, or you can write code directly in the Extensible Markup Language (XML).
3. Write code using the Java programming language. Create source code for all of the app's components.
4. Build and run the app on real and virtual devices. Use the default build configuration or create custom builds for different versions of your app.©
5. Test and debug the app's logic and UI.
6. Publish the app by assembling the final APK (package file) and distributing it through channels such as Google Play.

Resources to help you learn

- **Android developer training courses**
  - Codelabs with suggested homework assignments: Codelabs for Android Developer Fundamentals
  - Concept reference chapters: Android Developer Fundamentals — Concepts
  - Source code in GitHub for starter apps and solution code for apps that you create in the codelabs

- The Android Developer YouTube channel is a great source of tutorials and tips.

- Android vocabulary

- Inspire looking at Android Developer Stories

- **Community Support:**
- The Android team posts news and tips in the official Android blog.
- Android developers blog.
- Stack Overflow. If you run into a problem, chances are high that someone has already posted an answer.
  - Type [android] in the search box. The [] brackets indicate that you want to search for posts that have been tagged as being about Android.

Resources to help you learn

- Android developer fundamentals course
  - Unit 1.4: Resources to help you learn

- This page contains a great collection of links to many resources organized into categories;
  - Android Developer documentation
  - Android studio documentation
  - Design
  - Develop
  - Distribute
  - Blogs
  - Other resources

Building blocks of Android

App components

- **Activities**
- Services
- Broadcast Receivers
- Content Providers

Activating components

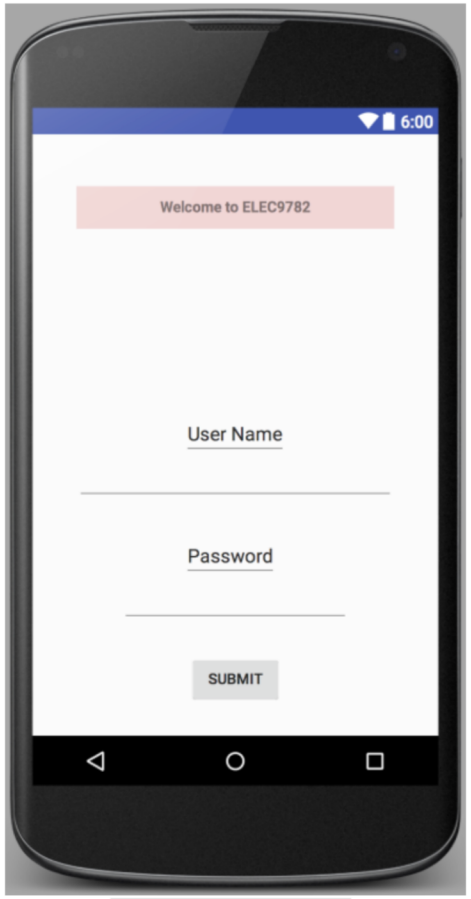
- **Intent**

Activity

- One of the basic building block of Android
- Most common component of Android development
- Represents a single screen with a user interface
- A single app can have multiple activities.
  E.g. A game app might have different activities for login screen, scores page, and the game play screens
- Associated with a XML file that defines the arrangement of GUI components.
- https://developer.android.com/guide/components/activities/intro-activities

Activity Example

Activity Operation is written in Java



Design View

activity_main.xml

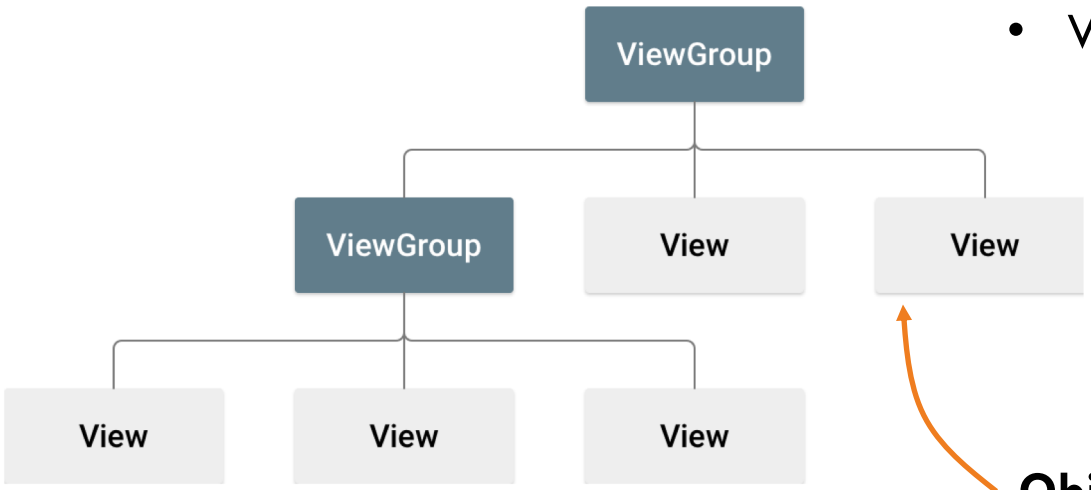
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal|top"
    android:textAlignment="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to ELEC9782"
        android:id="@+id/textView"
        android:layout_marginTop="49dp"
        android:background="#edcccc"
        android:textStyle="bold"
        android:width="300dp"
        android:height="40dp"
        android:gravity="center"
        android:textAlignment="center"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText1"
        android:textAlignment="center"
        android:layout_below="@+id/editText3"
        android:layout_alignStart="@+id/textView"
        android:layout_alignEnd="@+id/textView" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textPassword"
        android:ems="10"
        android:id="@+id/editText2"
        android:password="true"
        android:gravity="center_horizontal"
        android:textAlignment="center"
        android:layout_below="@+id/editText4"
        android:layout_centerHorizontal="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText3"
        android:text="User Name"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/editText4"
        android:text="Password"
        android:layout_marginTop="30dp"
        android:layout_below="@+id/editText1"
        android:layout_alignStart="@+id/editText3" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Submit"
        android:id="@+id/button"
        android:layout_below="@+id/editText2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="28dp" />

</RelativeLayout>
```

Text View

Activity Layout and other resources

Various Layouts

- Layout defines the visual structure of the GUI.
- View hierarchy



Objects - Layouts

- Linear Layout
- Relative Layout
- Web view

Objects - Widgets

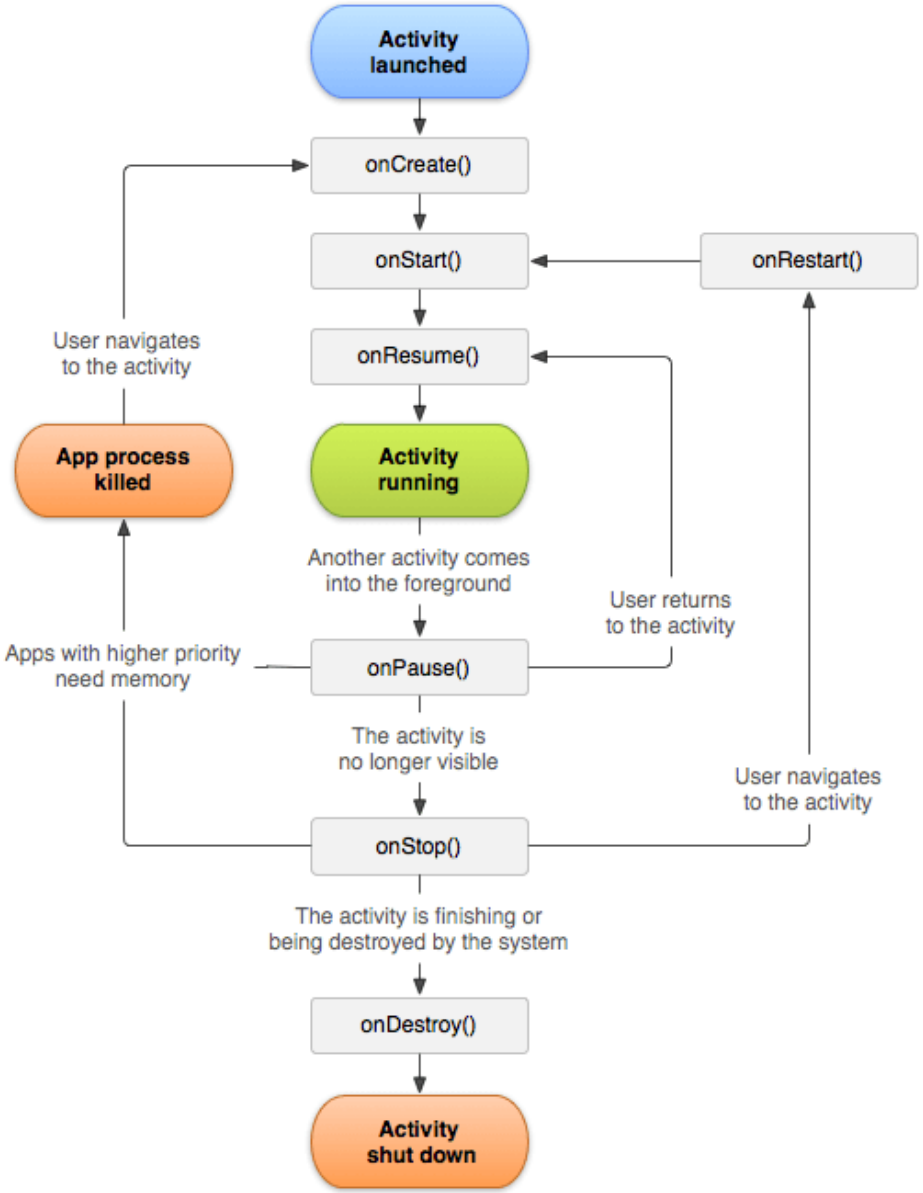
- Buttons
- Text view

Layouts

- Can declare Layouts
  - Writing the XML
  - Using Android Studio's "Layout Editor"


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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_centerHorizontal="true"
        android:layout_alignParentTop="true">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button1" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BUTTON2"
        android:id="@+id/button3"
        android:layout_weight="1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Button3" />
</LinearLayout>
```

Activity Lifecycle



Activity Lifecycle

- You can override lifecycle methods to develop your customized activity.
  - E.g. What to do after starting the app → Override onCreate() method
  - From tutorial 1;

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Use "activity_main.xml" as the layout
    setContentView(R.layout.activity_main);

    // Reference the "listView" variable to the id "lstView" in the layout
    listView = (ListView) findViewById(R.id.lstView);
    addItemEditText = (EditText) findViewById(R.id.txtNewItem);

    // Create an ArrayList of String
    items = new ArrayList<String>();
    items.add("item one");
    items.add("item two");
```

Few ideas on object oriented programming

- The elements we discussed such as Activities, Buttons, TextViews are all classes. What does it mean?
- **Classes** and **Instances**
- **Attributes** and **Methods**

- **Class** is a blueprint of something. For example, a Smartphone class will define that each smartphone must have **attributes** such as length, width, thickness, brand, OS, owner etc. and **methods** such as switch on, switch off, sleep.

- Kanchana's phone is a specific **instance** of the smartphone class, length=8cm, width=5cm, brand="Nexus", owner="Kanchana" .... and so on.

- Thilakarathna's phone is a specific **instance** of the smartphone class, length=6cm, width=4cm, brand="iPhone", owner="Thilakarathna" .... and so on.

Few ideas on object oriented programming

- In the previous example, we creates **instances** of Activity **class** such as MainActivity.

- When we set the Text, Width, and Height of these elements we were assigning values the **attributes**.

- We used the onCreate() **method** of the Activity class to include our GUI logic.

Declaring components – AndroidManifest.xml

- Describes essential information about your app to Android OS
- Examples:
  - App package name
  - Minimum API level required by the app
  - User permissions
  - Declare third party API libraries
  - Declare app's components, e.g. Activity
  - Declare component capabilities through **Intents** and **Intent filters**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

https://developer.android.com/guide/topics/manifest/manifest-intro

Intents

- Intent is a messaging object to request an action from another app component.
- Primary use-cases:
  - **To start an activity**
  - **To start a service**
  - **To deliver a broadcast**
- Intent types:
  - **Explicit Intents**: Communicate within the same application. Need to specify the exact name of the component , e.g. class name.
  - **Implicit Intents**: Communicate between applications. Requested by declaring the general action to perform, e.g. location.

- https://developer.android.com/guide/components/intents-filters

Building bocks of an Intent

1. Component name

- Name of the component to start
  - Must specify the name for *Explicit* Intent, e.g. class name of the new Activity.
  - Empty for *Implicit* Intent

2. Action

- String that specifies the desired operation, e.g. view or pick
  - `ACTION_DIAL` - Dial a number
  - `ACTION_EDIT` - Display data to edit
  - `ACTION_SYNC` - Synchronise device data with a server
  - `ACTION_MAIN` - Start as initial activity of the app.

Building bocks of an Intent

3. Data

- Data and type of data (MIME type) associated with the Intent
- Type of data should be related to the action
  - E.g. If the action is ACTION_DIAL, data should be the phone number.
- Formatted as URI object (Uniform Resource Identifier)
  - `Uri.prase("http://www.google.com")`
- Note: To set both URI and MIME type, use **`setDataAndType()`**

Building bocks of an Intent

4. **Category**
   - String containing additional information about the component
     - CATEGORY_BROWSABLE – To start a web browser to display data
   - Specify the category with `addCategory()`

5. **Extras**
   - Key-value pairs that carry additional information to complete the action
   - Add extra info with `putExtra()`

6. **Flags**
   - Metadata for the intent
     - E.g. How to launch the activity, how to treat it after launching, etc.
   - Can set flags using `setFalgs()`

Example 1

- Start another activity using an Intent
- Example: Tutorial 2
  - What type of an Intent is used ?

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String updateItem = (String) itemsAdapter.getItem(position);
    Log.i("MainActivity", "Clicked item " + position + ": " + updateItem);

    Intent intent = new Intent(MainActivity.this, EditToDoItemActivity.class);
    if (intent != null) {
        // put "extras" into the bundle for access in the edit activity
        intent.putExtra("item", updateItem);
        intent.putExtra("position", position);
        // brings up the second activity
        startActivityForResult(intent, EDIT_ITEM_REQUEST_CODE);
        itemsAdapter.notifyDataSetChanged();
    }
}
```

Example 2

- Communicate between apps.
- By declaring the general action to perform. In this case,
  - Action: **ACTION_VIEW**
  - Data: Formatted as Uniform Resource Identifier (URI object) to send Intent Data

```
        Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.google.com"));
          startActivity(browserIntent);
        Intent chooser = Intent.createChooser(browserIntent, "Load http://www.google.com");


});
```

- What type of intent is this?
- What can go wrong with the code above code block?

Intent Filters

- Declare which Intents that your app can receive with **intent-filter** element in your **AndroidManifest.xml**
  - The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.
- This is how Android pass Implicit Intents to relevant apps
- Define `<action/>`, `<data/>` and `<category/>`
- E.g. Declaration to receive `ACTION_SEND` intent with text data

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

Intent Filters

- Who had a look at the AndroidManifest.xml files of Tutorial 1?
- Were there any Intent filter?

Intent Filters

- `ACTION_MAIN` indicates this activity is the main entry point when the user launch the app and does not expect any intent data.
- `CATEGORY_LAUNCHER` indicates that activity's icon should be placed in the system's app launcher.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="comp5216.sydney.edu.au.todolist">

    <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=".EditToDoItemActivity"
            android:label="@string/app_name" >
        </activity>
    </application>

</manifest>
```

What's Next ?

- In Week 3;
  - Capabilities of Mobile Devices, e.g. sensors, location, etc.
  - Android building blocks
    - Services
    - Broadcast Receiver
    - Content Provider

- Project
  - Guideline is on Canvas
  - Start forming groups and thinking about ideas

- Tutorial 2
  - Tutors will help you to sign-in/register for project groups
  - Register the group with the tutorial class of most members

- **Please use "COMP5216" on the subject when you email me.**