CSC 4360 / CSC 6360

Data Persistence and Sensors

Using Textbook Examples

Getting textbook source code working on your machine

Textbook

- Textbook PDF purchase link is available on the class website.
- Class Webpage top navigation bar > Textbook
- Please purchase the textbook.
- Examples and exercises from the textbook will be used in class from now on, to expedite class progress :)

https://github.com/LearningMobile/BookApps2.0

Please have the textbook source code downloaded and projects imported for CH 3-8 by class start-time on Wednesday, Feb 14.

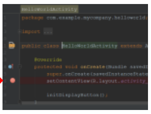
Importing Textbook Code

- Step 1: Make sure that you have installed Android Studio!
- Step 2: Make sure that you download the sourcecode from github
- Step 3: Save the unzipped folders into your local apps folder
- Step 4: Open Android Studio
- Step 5: Choose "Open an existing Android Studio Project"
- Step 6: Update as required
- Step 7: Run the application ☺

Debugging Applications

- Android Studio includes an integrated debugger
- *Select the bug icon for Debug As/Android Application*

- Set a breakpoint on a line of code by putting the cursor to the left of a line and right clicking
- Remove a breakpoint by right-clicking the RED circle at the left end of the line and selecting
- *Toggle Breakpoint DEMO*



HelloWorldActivity

```
package com.example.mycompany.helloworld;

import ...

public class HelloWorldActivity extends A

    @Override
    protected void onCreate(Bundle savedI
        super.onCreate(savedInstanceState
        setContentView(R.layout.activity_

        initDisplayButton();
    }
```

Data Persistence

Data Persistence

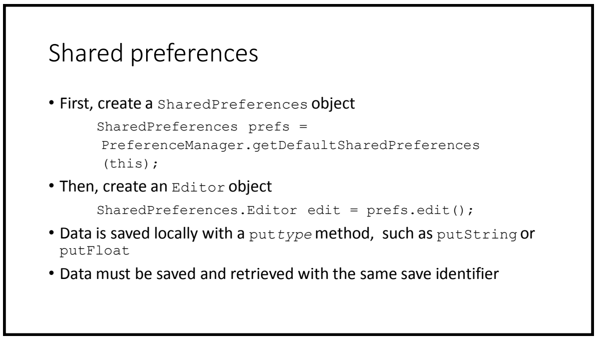
- To be useful, sometimes information (such as a grocery list) must be saved between uses of the application
- There are three different ways to persist data in Android: shared preferences (for primitive data), files, and SQLite relational databases

Android: Multiple options

- Most common data persistence approaches:
  - Shared Preferences
    - NOTE: on iOS, this functionality is provided by *NSUserDefaults*
  - Flat Files
    - NOTE: File I/O functionality is similar in iOS
  - Local device database (SQLite)
    - NOTE: SQLite accessibility is similar in iOS.

SharedPreferences

- Usually used for limited set of data
  - User preferences, App configuration
  - Small data that needs to persist beyond current app life cycle
- Shared Preferences object
  - Key-Value pairs of primitive data types (int, string, etc)
  - Persists as long as the app remains on the device
  - Not affected by app “upgrades”



Shared preferences

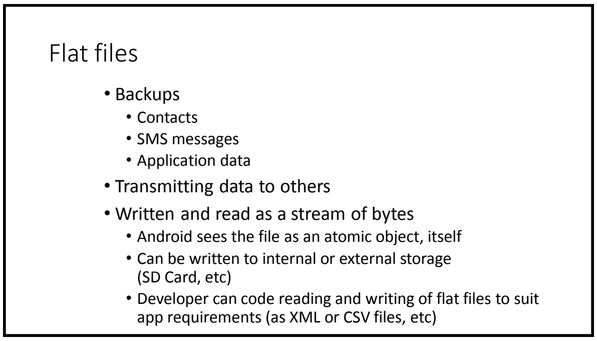
- First, create a `SharedPreferences` object

```
SharedPreferences prefs =
 PreferenceManager.getDefaultSharedPreferences
  (this);
```

- Then, create an `Editor` object

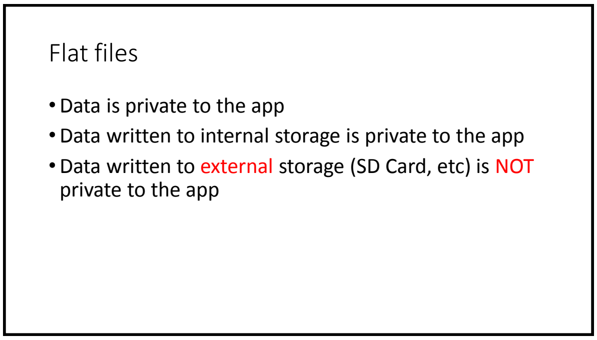
```
SharedPreferences.Editor edit = prefs.edit();
```

- Data is saved locally with a `put`*`type`* method, such as `putString` or `putFloat`
- Data must be saved and retrieved with the same save identifier



Flat files

- Backups
  - Contacts
  - SMS messages
  - Application data
- Transmitting data to others
- Written and read as a stream of bytes
  - Android sees the file as an atomic object, itself
  - Can be written to internal or external storage (SD Card, etc)
  - Developer can code reading and writing of flat files to suit app requirements (as XML or CSV files, etc)



Flat files

- Data is private to the app
- Data written to internal storage is private to the app
- Data written to external storage (SD Card, etc) is NOT private to the app

Local SQL Database, SQLite

- Android ships with a version of the SQLite Database

Android reference:
http://developer.android.com/reference/android/database/sqlite/package-summary.html

SQLite reference:
https://www.sqlite.org/android/doc/trunk/www/index.wiki

Local SQL Database, SQLite

- Fully functional RDBMS database
  - Doesn't require independent server process to run
  - Store and manipulate data via SQL Queries
- Data in the database is private to the app
- Data persists for as long as the app is installed on the device

Using the textbook src files

NOTE: full path to CH5 textbook Java src files:
BookApps/Ch05 Android Persistent Data/MyContactList/

- Follow the process from slide 4 for all of the files from the Java source folder on GitHub
- NOTE: Make sure this works while you are at home!! You may need to **update items** after you have opened the textbook code/projects.

Lists in Android:
Navigation and Information Display

- Two components are required for any Android list
  - a ListView widget
  - an adapter
- ListView widget
  - object that can display a vertical list of items that can be scrolled through.
- Adapter
  - provides access to the underlying data source for the list
  - dynamically associated with the ListView
- Bonus! For simple lists, after an adapter is associated with the widget, insertion of data as a list item is handled automatically.
- (For complex lists, developer must create a subclass of an Adapter object and code the display and behavior of the list in the new subclass)

Lists

- An AdapterView is the super class of all views that are bound to an underlying data source.
- A View is the base class for all user interface components, such as the widgets used in creating layouts.
- The AdapterView has several subclasses,
  - GridView
  - ListView
  - Spinner widgets
- The visible component of a list is implemented with a ListView widget in an XML layout file.

Widgets

- Has attributes that allow the user to configure some aspects of the display
- ListView is a standard widget, with (bonus!) the special ID of @id/android:list and a special subclass of Activity
  - many of the tasks of list implementation are easier because the developer can take advantage of many built-in features of the SDK
  - How? the Activity associated with the layout containing the ListView must be a subclass ofListActivity rather than Activity. (this is optional, but much easier)

Adapters

- Act as a link between the view and the underlying data source for the list
- Lists require the use of an adapter
- Adapters provides access to the data items and is responsible for creating a View for each item
- A view determines how each list item is displayed.
  - Usual: display is uniform for each data item
  - Optional: display does not have to be uniform, but in that case, developers must implement their own adapters to create the different views

ArrayAdapter

- ArrayAdapter is used to bind an Array or ArrayList to a view.
  - Always parameterized
  - Commonly used in list implementation
  - Must know in advance what ***type*** of data it will be displaying
    - can be simple, such as String
    - can be more complex, such as the Contact object

CursorAdapter

- abstract class that binds data from a database cursor to a view
- concrete subclass, SimpleCursorAdapter
  - Used to map a row layout to fields in a cursor
  - SimpleAdapter class is used to bind static data to a view

Steps

Chapter 6 activity: Lists in Android

1. Create the data source (p.100)
2. Create the layout (p.101)
3. Create the custom adapter (p. 103)
4. Code the actions in the Activity

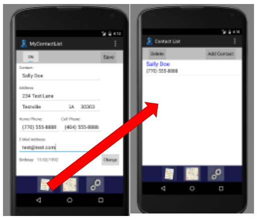
    (CH6 example: ContactListActivity.java)

Sensors and More!

Introduction to Android Sensors and Maps API

Review – CH 5 activity (saving data locally)

- Switch Edit button to ON
- Enter Contact Info
- Click Save
- Review Contact List



Contact: Sally Doe

Address: 234 Test Lane

Testville 1A 30303

Home Phone: (770) 555-8888

Cell Phone: (404) 555-8888

E-Mail Address: test@test.com

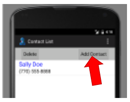
Birthday: 11/02/1992

Sally Doe

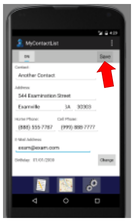
(770) 555-8888

Review – CH 6 Activity (more than 1 contact)

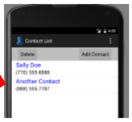
- Add A Contact
- Enter Contact Info
- Save
- …and Review



Sally Doe
(770) 555-8888



Another Contact
544 Examination Street
Examville 3A 30303
(888) 555-7787 (999) 888-7777
exam@exam.com
Birthday: 01/01/2000



Sally Doe
(770) 555-8888
Another Contact
(888) 555-7787

Navigation and Display, including Maps!

- Note that on p. 105 that the code to respond to an Item Click should look very familiar!!
- Review Double-clicking, Adding, Deleting contacts from Ch6 and ensure that you are comfortable with the Chapter source code for Ch5, Ch6, Ch7 (Maps and Location in Android) – we will cover more on Ch 7 in lab on Wed.

Sensors

- Sensors are hardware built-in to the mobile device to allow an app to capture some kind of environmental data
- Location sensors
  - Typically 2 location sensors
    - Network sensor
      - Cell towers and wifi access points to which the device is connected.
      - Faster
    - Built-in GPS (Global Positioning System) receiver.
      - Slower
  - GPS Usually accurate to within a few meters (+-10 feet), and not all devices have built-in GPS sensor.

Sensors

p.123

- Location information is accessed in the app by the LocationManager object (Android system service).
  - Accessed via getSystemService()
  - Can obtain information from either or both sensors
  - To use the data from the service, instantiate a LocationListener, which captures the sensor information whenever a location change is reported

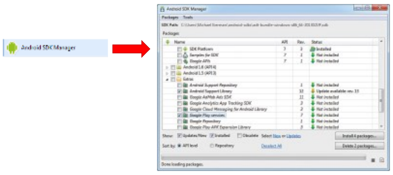
Maps

- Maps are used to display data as a visual representation of a location.
- Implemented with GoogleMaps object (Java) and a MapFragment in the xml/layout file.

- Must use Android SDK manager to obtain “Extras > Google Play Services” in order to use this functionality!

Google Play Services

- Use Android SDK Manager to install required SDKs:
  - Google Play Services are required!



Name	API	Rev.	Status
□ SDK Platform	7	3	Installed
□ Samples for SDK	7	1	Not installed
□ Google APIs	7	1	Not installed
□ Android 1.6 (API 4)			
□ Android 1.5 (API 3)			
□ Extras			
□ Android Support Repository		1	Not installed
☑ Android Support Library		12	Update available: rev. 13
□ Google AdMob Ads SDK		11	Not installed
□ Google Analytics App Tracking SDK		2	Not installed
□ Google Cloud Messaging for Android Library		3	Not installed
☑ Google Play services		7	Not installed
□ Google Repository		1	Not installed
□ Google Play APK Expansion Library		3	Not installed

Show: ☑ Updates/New ☑ Installed □ Obsolete Select New or Updates

Sort by: ● API level ○ Repository Deselect All

Done loading packages.

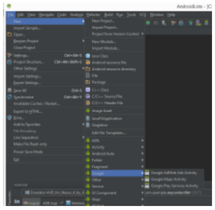
Ready Yet? Nope.

- Google API Key Required.
- Basic directions:

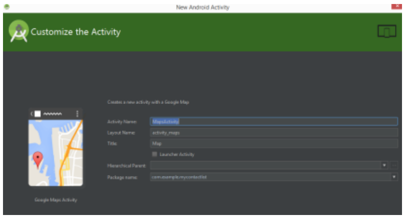
https://developers.google.com/maps/documentation/android-api/start#the_google_maps_api_key

Setting up your first Map use

- Android context
- Click on package
- File > New > Google > Google Map Activity



Add the activity



Creates a new activity with a Google Map

Activity Name: MapsActivity

Layout Name: activity_maps

Title: Map

□ Launcher Activity

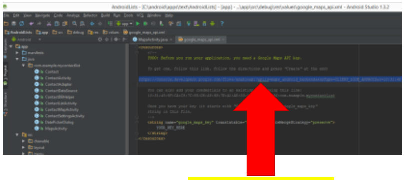
Hierarchical Parent:

Package name: com.example.mycontactlist

google_maps_api.xml

- google_maps_api.xml will display.
- Follow the directions to get an API Key
  - URL is in the comments.
  - Will look something like:
    - *https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=10:31:45:8F:CA:C9:7C:55:D5:28:98:7B:A1:A5:33:13:32:47:FC:82%3Bcom.example.mycontactlist*

Get the URL



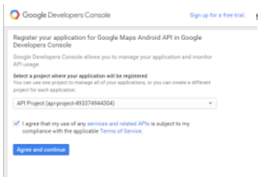
AndroidLists - [C:\android\apps\text\AndroidLists] - [app] - ...\app\src\debug\res\values\google_maps_api.xml - Android Studio 1.3.2

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=[illegible]

    You can also add your credentials to an existing [illegible] using this line:
    10:31:45:8F:CA:C9:7C:55:D5:28:98:7B:A1:A5:33:[illegible];com.example.mycontactlist

    Once you have your key (it starts with "[illegible]google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" translatable="[illegible]templateMergeStrategy="preserve">
        YOUR_KEY_HERE
    </string>
</resources>
```

URL to copy into your browser

Get your API Key

- Use your browser to go to the link, and follow the directions



Register your application for Google Maps Android API in Google Developers Console

Google Developers Console allows you to manage your application and monitor API usage.

Select a project where your application will be registered

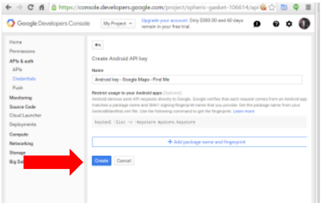
You can use one project to manage all of your applications, or you can create a different project for each application.

API Project (api-project-493374944304)

☑ I agree that my use of any services and related APIs is subject to my compliance with the applicable Terms of Service.

Choose Android Key

- Enable Maps under APIs and then go to Credentials



https://console.developers.google.com/project/spheric-gasket-106614/api

Create Android API key

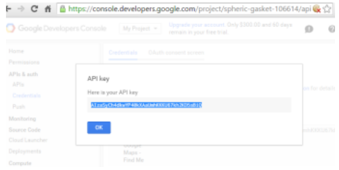
Name: Android key - Google Maps - Find Me

Restrict usage to your Android apps (Optional)

Android devices send API requests directly to Google. Google verifies that each request comes from an Android app matches a package name and SHA1 signing-fingerprint name that you provide. Get the package name from your AndroidManifest.xml file. Use the following command to get the fingerprint. Learn more

```
keytool -list -v -keystore mystore.keystore
```

Click Create, and Copy the Key



https://console.developers.google.com/project/spheric-gasket-106614/api

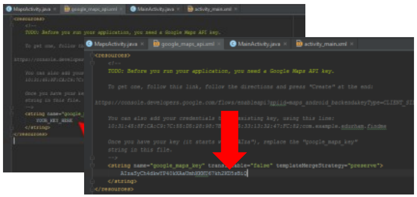
API key

Here is your API key

AIzaSyCh4dkwYP40kXAaUmhKKXU67kh2KD5sBiQ

Paste the Key into Android Studio

- Go back to the google_maps_api.xml and paste the key as directed



```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SI

    You can also add your credentials to an existing key, using this line:
    10:31:45:8F:CA:C9:7C:55:D5:28:98:7E:[illegible]:33:13:32:47:FC:82;com.example.edurham.findme

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" translatable="false" templateMergeStrategy="preserve">
        AIzaSyCh4dkwYP40kXAaUmhKKXU67kh2KD5sBiQ
    </string>
</resources>
```

CH 7 source code

- Review and implement CH 7 source code to modify the CH 6 Project to display a map for each contact
  - NOTE: You may want to start using some real addresses for the contacts, or your results will vary :)

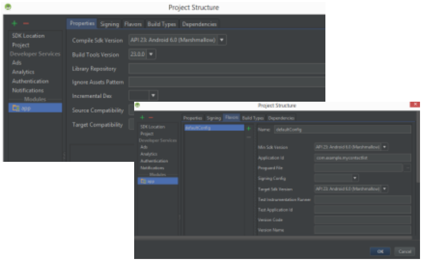
GOTCHAS!!

- Make sure that you have the correct SDK version in the project:

  File > Project Structure
- Make sure that you have at least an SDK-23 device to run in the emulator:

  Add to AVD, if required

File > Project Structure



Project Structure

Properties | Signing | Flavors | Build Types | Dependencies

SDK Location
Project
Developer Services
Ads
Analytics
Authentication
Notifications
Modules
app

Compile Sdk Version: API 23: Android 6.0 (Marshmallow)
Build Tools Version: 23.0.0
Library Repository:
Ignore Assets Pattern:
Incremental Dex:
Source Compatibility:
Target Compatibility:

Project Structure

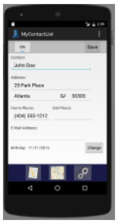
Properties | Signing | Flavors | Build Types | Dependencies

defaultConfig

Name: defaultConfig
Min Sdk Version: API 23: Android 6.0 (Marshmallow)
Application Id: com.example.mycontactlist
Proguard File:
Signing Config:
Target Sdk Version: API 23: Android 6.0 (Marshmallow)
Test Instrumentation Runner:
Test Application Id:
Version Code:
Version Name:

OK | Cancel

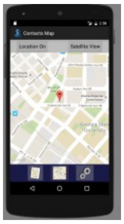
CH. 7 app (Expected Results)



MyContactList

ON | Save

Contact: John Doe
Address: 25 Park Place
Atlanta GA 30303
Home Phone: (404) 555-1212
Cell Phone:
E-Mail Address:
Birthday: 11/11/2015 | Change



Contacts Map

Location On | Satellite View

Page 141

CH 8

- Import CH 8 source code and add API key to bottom of AndroidManifest.xml

```
<meta-data android:name="com.google.android.maps.v2.API_KEY"
   android:value="your api key here" />
```

NOTES:

- Make sure that you have the correct SDK version in the project:
  - File > Project Structure
- Make sure that you have at least an SDK-23 device to run in the emulator:
  - Add to AVD, if required

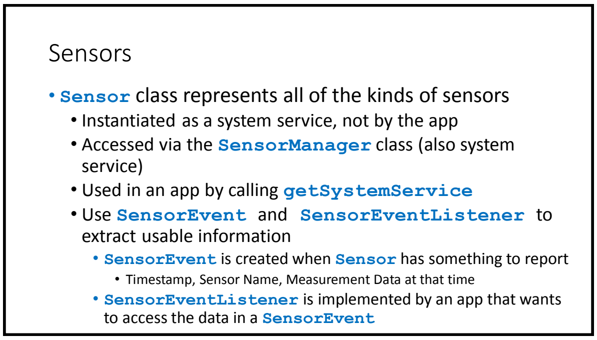
Hardware and Sensors in Android

- Currently, Android platform supports about 12 sensors, including motion, environmental, and position sensors, but not all device vendors include them all on any given physical device
- As part of your code, you should always make sure that a sensor exists before you attempt to make use of it.

Available Sensors

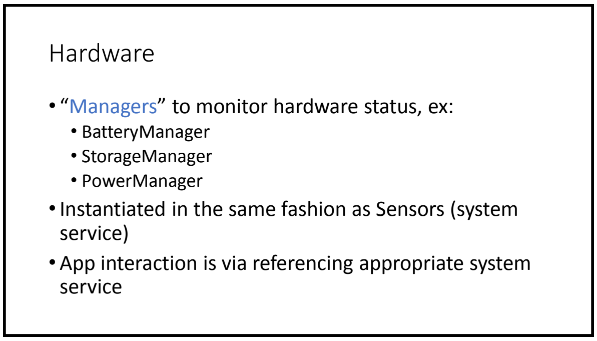
- Temperature
- Relative Humidity
- Atmospheric Pressure
- Magnetic field
- Light Level
- Rotation/Movement detection
- etc

Again, whether you can actually USE all of these sensors (and more) in your code depends on the device manufacturer



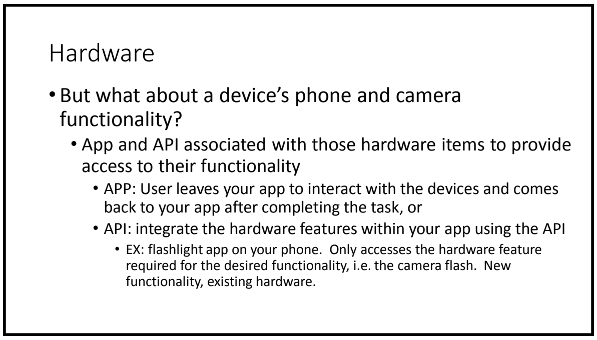
Sensors

- `Sensor` class represents all of the kinds of sensors
  - Instantiated as a system service, not by the app
  - Accessed via the `SensorManager` class (also system service)
  - Used in an app by calling `getSystemService`
  - Use `SensorEvent` and `SensorEventListener` to extract usable information
    - `SensorEvent` is created when `Sensor` has something to report
      - Timestamp, Sensor Name, Measurement Data at that time
    - `SensorEventListener` is implemented by an app that wants to access the data in a `SensorEvent`



Hardware

- “Managers” to monitor hardware status, ex:
  - BatteryManager
  - StorageManager
  - PowerManager
- Instantiated in the same fashion as Sensors (system service)
- App interaction is via referencing appropriate system service

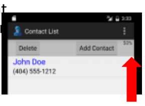


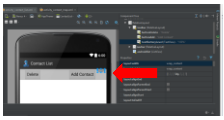
Hardware

- But what about a device’s phone and camera functionality?
  - App and API associated with those hardware items to provide access to their functionality
    - APP: User leaves your app to interact with the devices and comes back to your app after completing the task, or
    - API: integrate the hardware features within your app using the API
      - EX: flashlight app on your phone. Only accesses the hardware feature required for the desired functionality, i.e. the camera flash. New functionality, existing hardware.

Ch 8 app (expected functionality)

- Add “battery monitor” to upper right using a TextView as a place-holder o activity_contact_list.xml
- Add BroadcastReceiver to ContactListActivity.java to populate the placeholder TextView

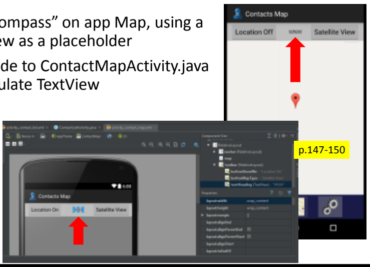




p.146-7

Ch 8 app (expected functionality)

- Add “compass” on app Map, using a TextView as a placeholder
- Add code to ContactMapActivity.java to populate TextView



p.147-150

Ch 8 app (expected functionality)

- Phone actions (press and hold, to bring up phone app)
- Use intent to hop to the phone app in ContactActivity.java file (callContact method)

```
private void callContact(String phoneNumber) {
   Intent intent = new Intent(Intent.ACTION_CALL);
   intent.setData(Uri.parse("tel:" + phoneNumber));
   startActivity(intent);
}
```

Ch 8 app (expected functionality)

p.157-9

- Camera activities
  - Can use in the same manner as phone app is accessed
  - Can use API to directly access hardware
    - Dependent on device hardware

Not covered in chapter

New Team Work

- Start working on your user stories (or you can use cases if you prefer – but you must stick to either one or the other).
- You must work together as a group! Each individual person on your team must complete at least ONE user story (or use case).

Coming up Next -

- Please review Chapter 7 & 8 for Wednesday's lab ☺