**455-555 Week 7 Session B Notes**

Working with Lab 4; working Alert Dialogs, LoginActivity

Android Self Review of Components

**Exercise:** (Time permitting)

-ViewFlipper- Dealing with images, animations

-LogCat (check PowerPoint)

**[ Lab 4 logic - working the LoginActivity, Alert Dialogs ]**

**Sample Login screen construction**

**Graphical user interface, application

Description automatically generated**

**Sample XML Layout follows…**

<EditText  
 android:id="@+id/textUserName"  
 android:layout\_width="140dp"  
 android:layout\_height="50dp"  
 android:layout\_marginStart="140dp"  
 android:layout\_marginTop="124dp"  
 android:textSize="18sp"  
 android:hint="Enter user name"  
 android:inputType="textPersonName"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
<EditText  
 android:id="@+id/textPassword"  
 android:layout\_width="140dp"  
 android:layout\_height="50dp"  
 android:layout\_marginTop="20dp"  
 android:hint="Enter password"  
 android:textSize="18sp"  
 android:inputType="textPassword"  
 app:layout\_constraintStart\_toStartOf="@+id/textUserName"  
 app:layout\_constraintTop\_toBottomOf="@+id/textUserName" />  
  
<Button  
 android:id="@+id/button"  
 android:layout\_width="136dp"  
 android:layout\_height="57dp"  
 android:layout\_marginTop="56dp"  
 android:onClick="onClick"  
 android:text="Submit"  
 app:layout\_constraintStart\_toStartOf="@+id/textPassword"  
 app:layout\_constraintTop\_toBottomOf="@+id/textPassword" />

**Show data into an** [**alertdialog**](https://developer.android.com/guide/topics/ui/dialogs) **box on Button click event**

**1.**

public class LoginActivity extends AppCompatActivity {  
  
 int c = 3; //for tries  
 //StringBuilder data = new StringBuilder();

**2.**

public void onClick(View view) {

System.*out*.print("J Papa");

// hard code in credentials

String uname = "jpapa", pass = "alpha1";  
 EditText name, word;  
 name = findViewById(R.id.*textUserName*);  
 word = findViewById(R.id.*textPassword*);

//check username and password entered in against credentals

//if credentials are valid redirect user (startActivity(intent)

//else decrement c by 1

//call showMessage() to trigger alert dialog message   
}

**3.**

public void showMessage() {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setCancelable(true);  
 builder.setTitle("Password Tries remaining");  
 builder.setMessage("Tries left => " + c);  
 builder.show();  
}

**[ Self Review of Android Components ]**

### Activities

Activities are said to be the presentation layer of our applications. The UI of our application is built around one or more extensions of the Activity class. By using Fragments and Views, activities set the layout and display the output and also respond to the user’s actions. An activity is implemented as a subclass of class Activity.

### Services

Services are like invisible workers of our app. These components run at the backend, updating your data sources and Activities, triggering Notification, and also broadcast Intents. They also perform some tasks when applications are not active. A service can be used as a subclass of class Service:

public class ServiceName extends Service { }

Content Providers

It is used to manage and persist the application data also typically interacts with the SQL database. They are also responsible for sharing the data beyond the application boundaries. The Content Providers of a particular application can be configured to allow access from other applications, and the Content Providers exposed by other applications can also be configured.

public class contentProviderName extends  ContentProvider {

public void onCreate() {}}

### Broadcast Receivers

They are known to be intent listeners as they enable your application to listen to the Intents that satisfy the matching criteria specified by us. Broadcast Receivers make our application react to any received Intent thereby making them perfect for creating event-driven applications.

### Intents

It is a powerful inter-application message-passing framework. They are extensively used throughout Android. Intents can be used to start and stop Activities and Services, to broadcast messages system-wide or to an explicit Activity, Service or Broadcast Receiver or to request action be performed on a particular piece of data.

### Widgets

These are the small visual application components that you can find on the home screen of the devices. They are a special variation of Broadcast Receivers that allow us to create dynamic, interactive application components for users to embed on their Home Screen.

### Notifications

Notifications are the application alerts that are used to draw the user’s attention to some particular app event without stealing focus or interrupting the current activity of the user. They are generally used to grab user’s attention when the application is not visible or active, particularly from within a Service or Broadcast Receiver. Examples: E-mail popups, Messenger popups, etc.

**[ ViewFlipper exercise ]**

Working a simple ViewFlipper with auto image transitions! Unpack the Week 7 ViewFlipper.zip files to a desired folder.

1. Create a project in AS called with an Empty Views Activity. Name your project **ViewFlipper**.

2. Extract the **fruits.zip** file from your folder. Copy all the fruit files and paste them into your drawables folder.

3. Override your MainActivity file with code + imports from your MainActivity file in your folder.

4. Copy in the downloaded activity\_main.xml file from your folder and override your existing xml markup in your project.

5. Run app to test!

6. On to modding!

* Include for the first image to appear to be fruits.jpg, but only have it appear once on start up.
* Include a button to allow for the user to manually flip thru each image at will.
* Results can be seen in the included ViewFlipper(modded).zip inside your demo zip file.

**[ Running Emulator inside IDE ]**

To run the emulator in Android Studio, make sure you're using Android Studio 4.1 or higher with version 30.0.10 or higher of the Android Emulator, then follow these steps:

1. Click File > Settings > Tools > Emulator (or Android Studio > Preferences > Tools > Emulator on macOS), then select Launch in a tool window and click OK.
2. If the Emulator window didn't automatically appear, open it by clicking View > Tool Windows > Emulator.
3. Start your virtual device using the AVD Manager or by targeting it when running your app.

Limitations  
Currently, you can't use the emulator's extended controls when it's running in a tool window. If your development workflow depends heavily on the extended controls, continue to use the Android Emulator as a standalone application