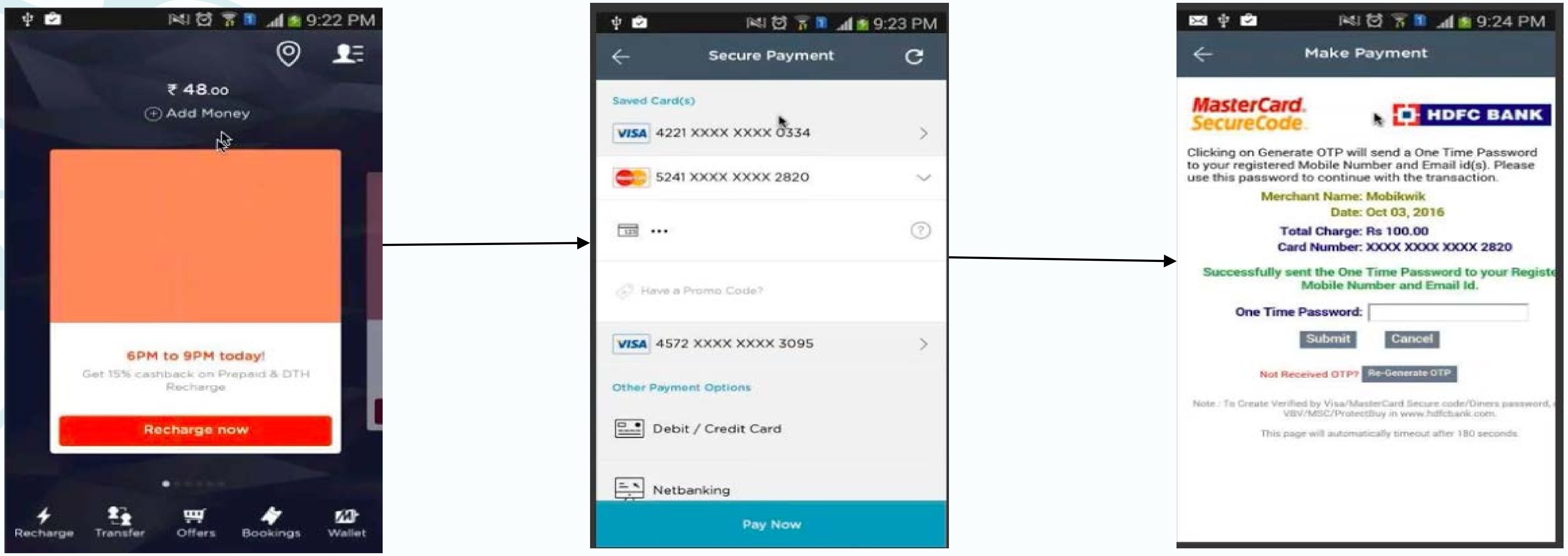
Lesson – 7 WebView & HTML

Agenda

- Introduction
- WebView Operations
- Hands on Example
- Methods in WebViewClient class
- How create your own HTML Page

Introduction

- WebView is one of the UI component in Android, which is used to display webpages in our application.
- If you want to deliver a web application (or just a web page) as a part of a client application, you can do it using WebView
- The WebView class is an extension of Android's View class that allows you to display web pages as a part of your activity layout.
- Sample Scenario example : An Android app called MobikWik is an mobile recharge app, once you click on the Pay Now button, it will take the Bank Webpage.



Clicking on Generate OTP will send a One Time Password to your registered Mobile Number and Email id(s). Please use this password to continue with the transaction.

Merchant Name: Mobikwik
Date: Oct 03, 2016
Total Charge: Rs 100.00
Card Number: XXXX XXXX XXXX 2820

Successfully sent the One Time Password to your Registe
Mobile Number and Email Id.

Note : To Create Verified by Visa/MasterCard Secure code/Diners password,
VBV/MSC/ProtectBuy in www.hdfcbank.com.

This page will automatically timeout after 180 seconds.

Work with WebView

Adding the Widget : To add a WebView to your Application, simply include the <WebView> element in your activity layout.

```
<WebView android:id="@+id/webview"/>
```

Loading Content Via URL : To load a web page in the WebView, use loadUrl().

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```

However, we also have to make one change to AndroidManifest.xml, adding a line where we request permission to access the Internet:

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

Reading Resource : https://developer.android.com/guide/webapps/index.html

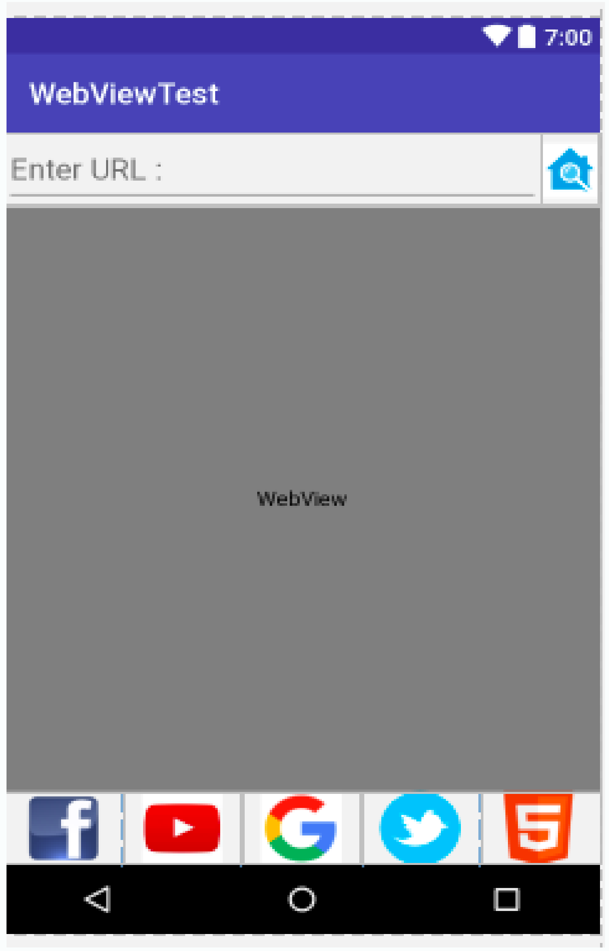
WebView Operations

- By using Webview we can perform 4 operations.

1. call browser
2. integrate browser
3. display .HTML files
4. we can provide a communication between HTML UI & Android Activity.

Hands on example

Problem : Type the expected URL in the EditText UI and click the search button to load the webpage in the WebView component. Below the WebView there are five Images to open the specified URL mostly preferred by the users. Click that image to load the web page in the WebView component.



activity_main.xml

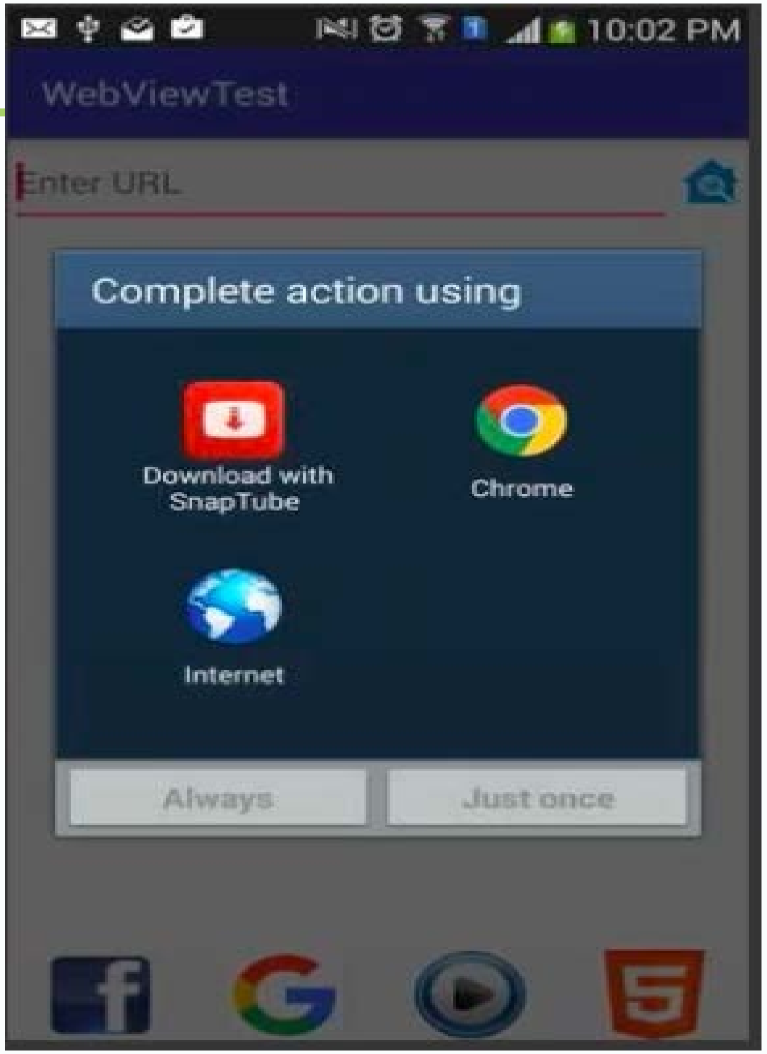
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="0.1"
    android:orientation="horizontal">
    <EditText
      android:id="@+id/et1"
      android:layout_width="0dp"
      android:layout_height="match_parent"
      android:layout_weight="0.9"
      android:textSize="20dp"
      android:hint="Enter URL : "/>
    <ImageView
      android:id="@+id/srch"
      android:layout_width="0dp"
      android:layout_height="match_parent"
      android:layout_weight="0.1"
      android:onClick="test"
      android:src="@drawable/search_logo" />
  </LinearLayout>
<WebView
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="0.8"
   android:id="@+id/wview1">
</WebView>
<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="0dp"
   android:layout_weight="0.1"
   android:orientation="horizontal">
  <ImageView
     android:layout_width="0dp"
     android:layout_height="match_parent"
     android:layout_weight="0.2"
     android:src="@drawable/fb_icon"
     android:onClick="test"
     android:id="@+id/fb"/>
  <ImageView
    android:layout_width="0dp"
    android:layout_height="match_parent"
    android:layout_weight="0.2"
    android:src="@drawable/youtube_logo"
    android:onClick="test"
    android:id="@+id/youtube"/>
<ImageView
     android:layout_width="0dp"
     android:layout_height="match_parent"
     android:layout_weight="0.2"
     android:src="@drawable/g_logo"
     android:onClick="test"
     android:id="@+id/google"/>
  <ImageView
     android:layout_width="0dp"
     android:layout_height="match_parent"
     android:layout_weight="0.2"
     android:src="@drawable/twitter_logo"
     android:onClick="test"
     android:id="@+id/twitter/>
  <ImageView
     android:layout_width="0dp"
     android:layout_height="match_parent"
     android:layout_weight="0.2"
     android:src="@drawable/html_logo"
     android:onClick="test"
     android:id="@+id/html/>
 </LinearLayout>
</LinearLayout>
```

MainActivity.java

```java
public class MainActivity extends AppCompatActivity {
   // Decalre object for the WebView
   WebView wview;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      // Configure the WebView id from XML to Java
      wview = (WebView) findViewById(R.id.wview1);
   }
   // test method decide action based on the image view selected to load the Webpage in WebView UI
   public void test(View v){
      switch (v.getId()){
         case  R.id.srch :
            // Configure the EditText id from XML to Java
            EditText et1 = (EditText) findViewById(R.id.et1);
            // Load the EditText URL to the WebView UI, it is mendatory to mention http:// either here or at the runtime in EditText
            wview.loadUrl("http://"+et1.getText().toString());
            break;
         case  R.id.fb : wview.loadUrl("http://facebook.com");
            break;
         case  R.id.youtube : wview.loadUrl("http://youtube.com");
            break;
         case  R.id.google: wview.loadUrl("http://google.com");
            break;
         case  R.id.twitter : wview.loadUrl("http://twitter.com");break;
         case  R.id.html : break;
      }
   }
}
```

MainActivity.java

– Once you run the app if you click the facebook, youtube, google and twitter image, it will produce the following screen, because just we loaded the URL, but we didn't integrate the browser.



Handling Page Navigation

- When the user clicks a link from a web page in your **WebView**, the default behavior is for Android to launch an application that handles URLs.
- Usually, the default web browser opens and loads the destination URL. However, you can override this behavior for your **WebView**, so links open within your **WebView.**
- You can then allow the user to navigate backward and forward through their web page history that's maintained by your **WebView**
- To open links clicked by the user, simply provide a **WebViewClient** for your **WebView** using **setWebViewClient**.

For example:

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebViewClient(new WebViewClient());
```

Methods in WebViewClient class

1. void onPageStarted () - Notify the host application that a page has started loading.
2. boolean shouldOverrideUrlLoading() - Give the host application a chance to take over the control when a new url is about to be loaded in the current WebView.
3. void onPageFinished () - Notify the host application that a page has finished loading.

For more information WebViewClient refer :

https://developer.android.com/reference/android/webkit/WebViewClient.html

MainActivity.java Modified

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // Configure the WebView id from XML to Java
    wview = (WebView) findViewById(R.id.wview1);
    // Integrate the browser with WebView
    /* We are just dislaying Toast message in below three methods,
    but in real time you can write your pre, post execution operations in your logic
    by overriding WebViewClient methods*/
    wview.setWebViewClient(new WebViewClient(){
        @Override
        public void onPageStarted(WebView view, String url, Bitmap favicon) {
            super.onPageStarted(view, url, favicon);
            Toast.makeText(getApplicationContext(), " Page loading is started",Toast.LENGTH_LONG).show();
        }
        @Override
        public void onPageFinished(WebView view, String url) {
            super.onPageFinished(view, url);
            Toast.makeText(getApplicationContext(),"On Page finished",Toast.LENGTH_LONG).show();
        }
    });
}
```

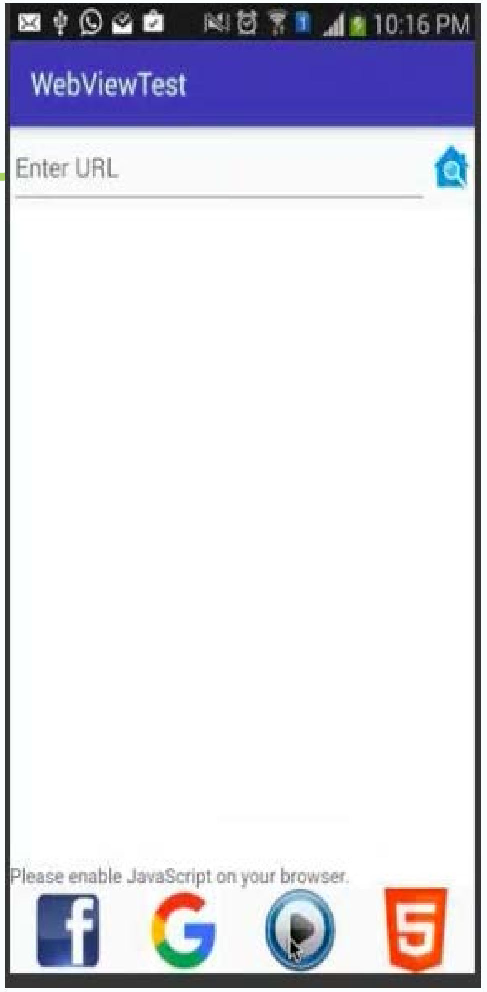
Supporting JavaScript :

- Now, you may be tempted to replace the URL with something else that relies upon JavaScript. You will find that such pages do not work especially well by default(see the image).
- That is because, by default, JavaScript is turned off in WebView widgets.
- If you want to enable JavaScript, call the below method on the WebView instance.

getSettings().setJavaScriptEnabled(true);

- At this point, any JavaScript referenced by your Web page should work normally.
- Sets whether the WebView should use its built-in zoom mechanisms.
- The built-in zoom mechanisms comprise on-screen zoom controls, which are displayed over the WebView's content, and the use of a pinch gesture to control zooming. To enable by giving

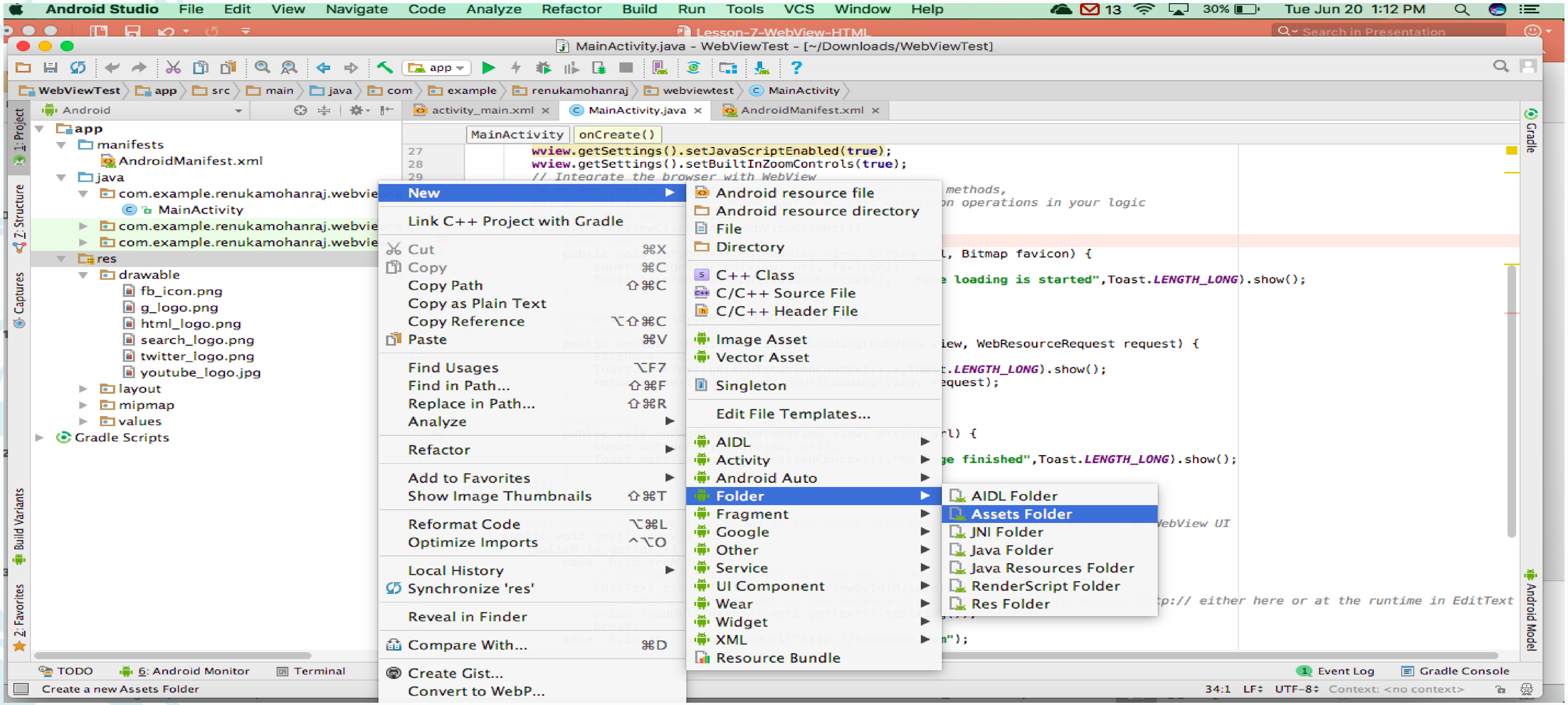
wview.getSettings().setBuiltInZoomControls(**true**);



Please enable JavaScript on your browser.

Creating own HTML Page

– Step 1 : Create an assets folder in your res directory as per the given screen shot.

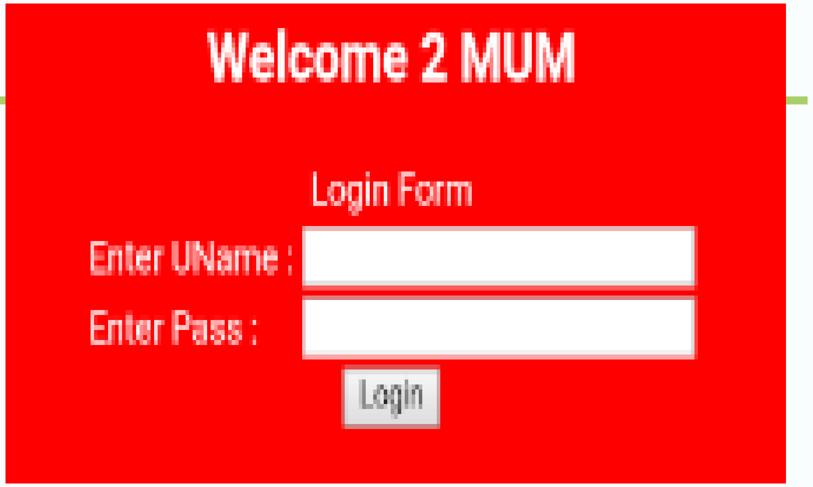


```
wview.getSettings().setJavaScriptEnabled(true);
wview.getSettings().setBuiltInZoomControls(true);
// Integrate the browser with WebView
methods,
on operations in your logic
l, Bitmap favicon) {
e loading is started",Toast.LENGTH_LONG).show();
iew, WebResourceRequest request) {
t.LENGTH_LONG).show();
equest);
rl) {
ge finished",Toast.LENGTH_LONG).show();
WebView UI
tp:// either here or at the runtime in EditText
n");
```

Creating own HTML Page

– Step – 2 : Create an HTML under assets folder. By Right Click assets→New→File. Then login.html as a file name.

–
```
<html>
<body bgcolor="#FF0000" text="FFFFFF">
<head>
</head>
<center>
  <h2> Welcome 2 MUM </h2>
  <table>
    <tr><td colspan='2' align="center">Login Form </td></tr>
    <tr><td>Enter UName : </td> <td> <input type="text" id="name"/></td></tr>
    <tr><td>Enter Pass : </td> <td> <input type="password" id="pass"/></td></tr>
    <tr><td colspan='2' align="center"> <input type="button" value="Login"/> </td></tr>
  </table>
</center>
</body>
</html>
```



Creating own HTML Page

- Step – 3 : Load login.html in MainActivity.java using
  case R.id.***html*** :
  wview.loadUrl("file:///android_asset/login.html"); break;

Now the requirement is to get the username and password to the Android activity.

Using JavaScript interface we can provide the communication between HTML UI & Android activity by calling

wview.addJavaScriptInterface(class_object, interface_name);

Example : wview.addJavaScriptInterface(this, “myinterface”);

Using the second parameter interface_name we can communicate with the specified class methods and variables from JavaScript.

JavaScript logic

– Step 4 : Once the user click the login button, need of JavaScript logic in HTML.

<tr><td colspan='2' align="center"> <input type="button" value="Login" onclick="login()"/> </td></tr>

Write an JavaScript code inside the <head> tag.

Reason : The reason behind this is as the Head gets loaded before the body. Any dynamic javascript code that gets executed in the body on load will execute correctly. Also to speed up the process.

The Document interface represents the entire HTML or XML document. Conceptually, it is the root of the document tree, and provides the primary access to the document's data.

Element getElementById (String elementId) - Returns the Element that has an ID attribute with the given value. If no such element exists, this returns null .

To retrieve the value invoke getElementById(“ide”).value

JavaScript logic

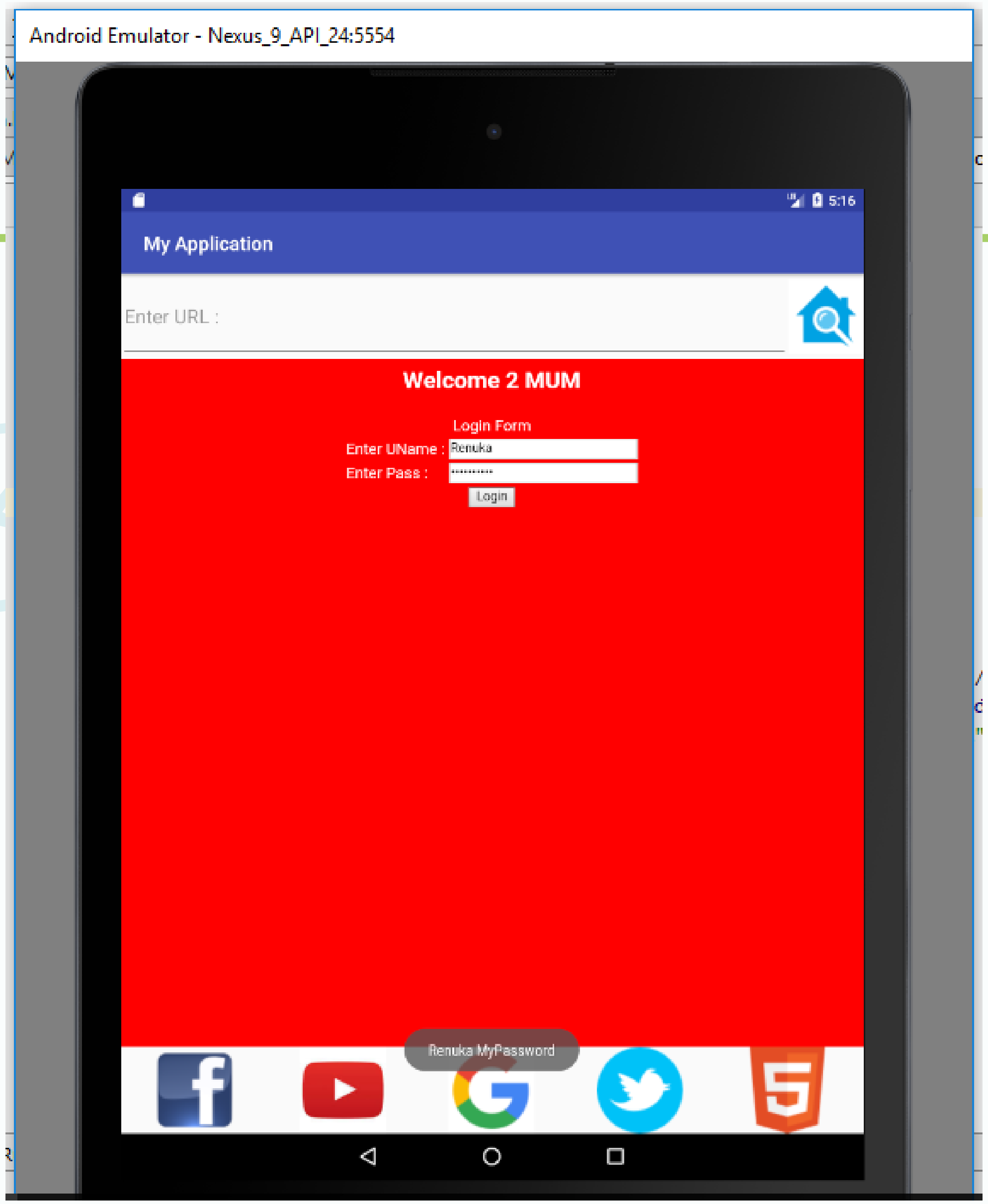
login.html

```
<head>
   <script language="JavaScript">
         function login(){
         var name=document.getElementById('name').value;
         var pass=document.getElementById('pass').value;
         myinterface.displayMsg(name,pass);
         }
      </script>
</head>
```

MainActivity.java - add the below method

```
@JavascriptInterface  // Annotation that allows exposing methods to JavaScript.
public void displyMsg(String name,String pass){
   Toast.makeText(getApplicationContext(),name + " " + pass , Toast.LENGTH_LONG).show();
}
```

HTML Click - Output



Android Emulator - Nexus_9_API_24:5554

My Application

Enter URL :

Welcome 2 MUM

Login Form

Enter UName : Renuka

Enter Pass : ••••••••••

Login

Renuka MyPassword