CASTLE LIBRARY USER MANUAL

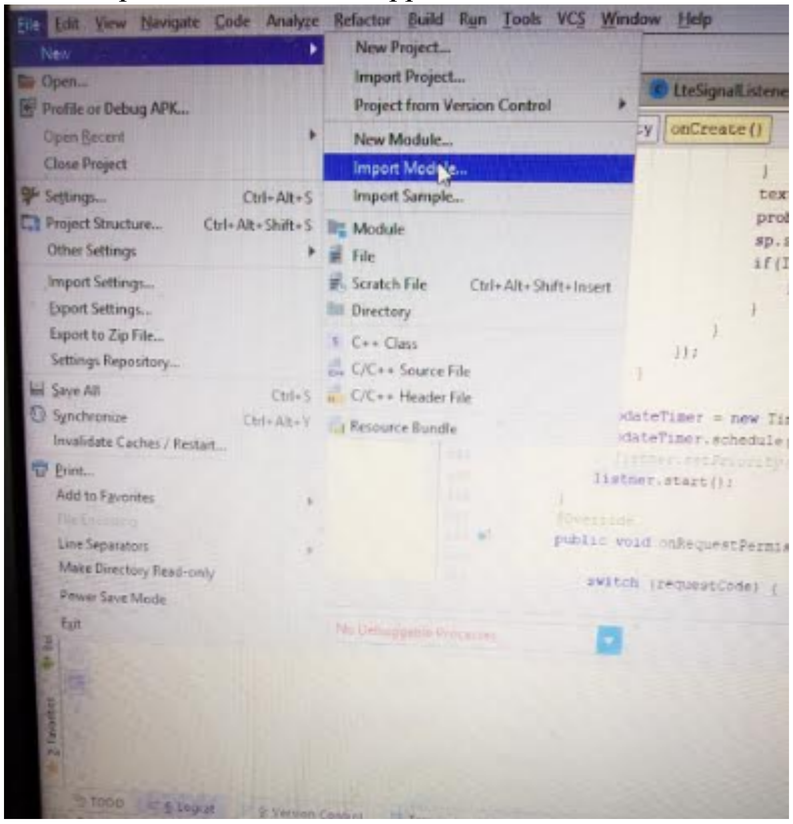
Castle library has following API's and description and usage of each API is covered in this manual

Installation of library:

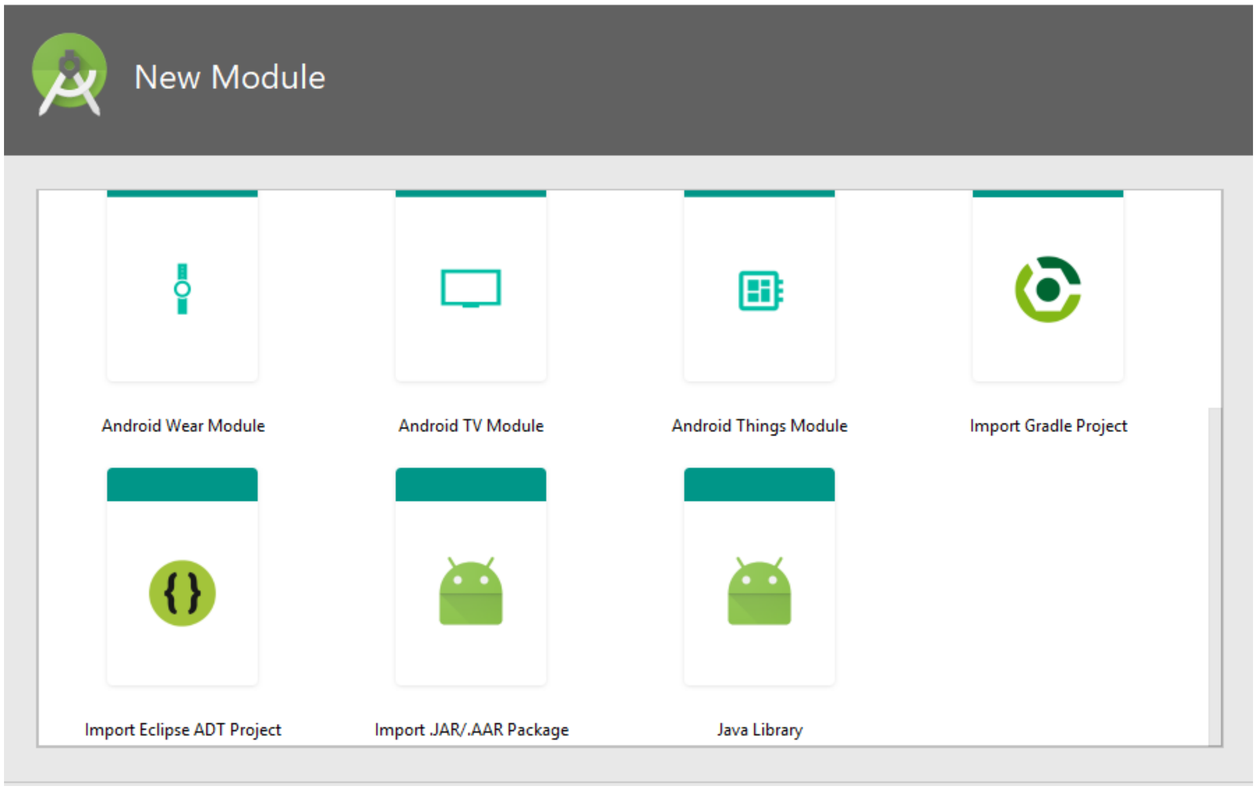
Download the CASTLE_LIBRARY from https://github.com/cu-pscr/CASTLE_LIBRARY.git and create the .aar file module and load this module onto application that you build.

Step 1: Load the library onto the application

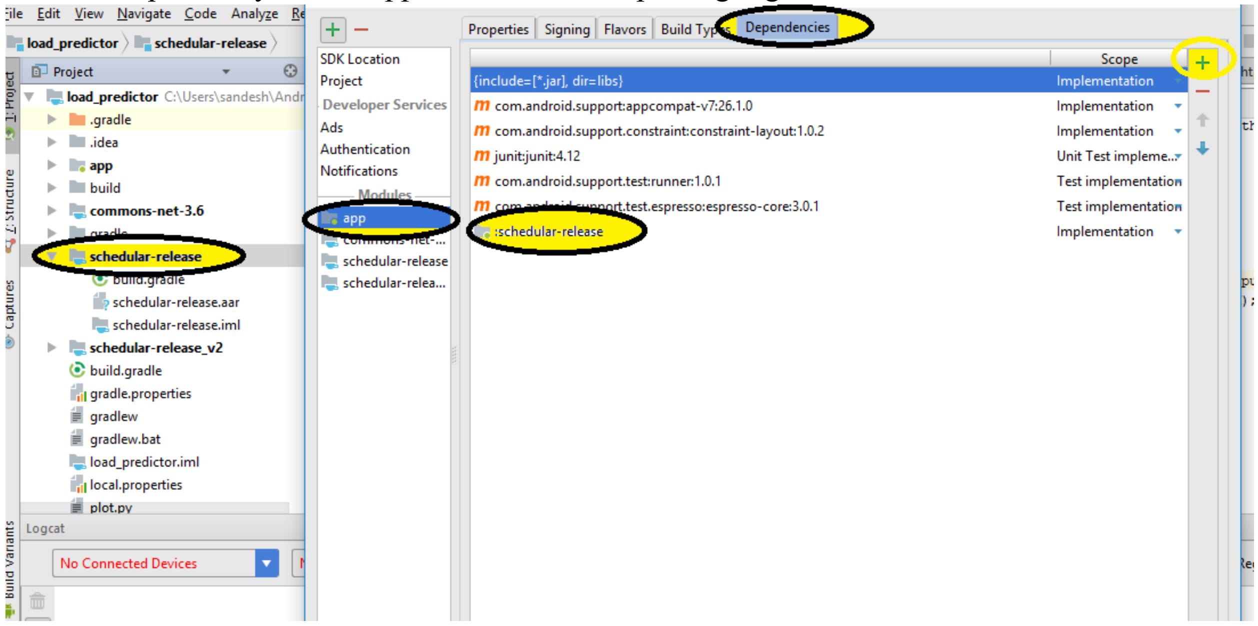
Open android application and keep the built library .aar file ready. Once project is opened, create a new import module for the application.



```
dateTimer = new Ti
dateTimer.schedule(
listner.start();
public void onRequestPermis
switch (requestCode) {
```



Once library is imported, right click onto library, select library management, click on the app and add it as dependency for the application. Each step is highlighted below.



	Scope
{include=[*.jar], dir=libs}	Implementation
com.android.support:appcompat-v7:26.1.0	Implementation
com.android.support.constraint:constraint-layout:1.0.2	Implementation
junit:junit:4.12	Unit Test impleme...
com.android.support.test:runner:1.0.1	Test implementation
com.android.support.test.espresso:espresso-core:3.0.1	Test implementation
:schedular-release	Implementation

Now use each of the library function in the application you need.

Description of library:

The library has following important public functions that could be used in your application

1. `castle_predict_class_long( )`
2. `castle_predict_class_short()`
3. `castle_compute_load_percentage()`
4. `castle_schedule( )`

predict class long avoids the sensing error and short version gives the instantaneous predicted class. It is always better to use long version than the short. Compute_load_precentage will computes and give the actual load percentage that is needed. Schedule function will schedule if the downloading can be done or not.

Usage of library:

Once you have installed the application using the library and corresponding permissions, your application should have complete access to the external storage and network access. Hence make sure you have taken care of it in the application part. It might take three/four times to get permission. Hence re-start application three times at least, so it has permissions for all the flags you have set in the application.

Also copy the machine learning look up file from the git link to the external storage of your mobile device. https://github.com/cu-pscr/CASTLE_data.git

See below box which specifies the permission tags you should add in your manifest as well as in your main activity, so your application will have access to the external storage and Network to sense media.

```
<uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.READ_PHONE_STATE" />
<uses-permission
android:name="com.android.alarm.permission.SET_ALARM"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE"/>
```

After this, make use of threads to call the API's. The library is dependent on the sensing of media and getting the RSRP, RSRQ and SNR values. Hence shared Preferences should be created and called through castle initializing function called castle_scheduler.init(). Check the code snippet below how you should call the init function.

```
final SharedPreferences
sharedPreferences=getApplicationContext().getSharedPreferences("SNR",0);
final SharedPreferences.Editor editor=sharedPreferences.edit();
Thread initialise=new Thread(new Runnable() {
    @Override
    public void run() {
        cs.init(sharedPreferences,astar,b,ddl);//8000*1000
    }
});
```

Where the astar,b and ddl are the variables values that should be computed using equation from the paper.

This shared preference value should be written using the sensing of values:

```
final TelephonyManager
mTelephonyManager=(TelephonyManager)this.getSystemService(Context.TELEPHONY_SERVICE)
;
final PhoneStateListener mPhoneStateListener=new PhoneStateListener(){
    public void onSignalStrengthsChanged(SignalStrength signalStrength) {
        Log.d("castle_library","signal strength changed");
        super.onSignalStrengthsChanged(signalStrength);
        Pattern SPACE_STR = Pattern.compile(" ");
        String[] splitSignals = SPACE_STR.split(signalStrength.toString());
        if (splitSignals.length < 3) {
            splitSignals = signalStrength.toString().split("[ .,|:]+");
        }
        String s_rsrp = splitSignals[9];
        String s_rsrq = splitSignals[10];
        String s_snr = splitSignals[11];
        String s_cqi = splitSignals[12];
        int snr=Integer.valueOf(s_snr);
        Log.d("snr sensed1:\t",""+snr);
        editor.putInt("snr_value",snr);
        boolean edit_ok=editor.commit();
        while(!edit_ok){
            edit_ok=editor.commit();
        }
    }
};
```

Will help you to sense the media.
A separate thread should be created and installed and used to call the sensing thread. Like showed below

```
Thread sensing= new Thread(new Runnable() {
    @Override
    public void run() {
        while(true) {
            mTelephonyManager.listen(mPhoneStateListener,
PhoneStateListener.LISTEN_SIGNAL_STRENGTHS);
            mTelephonyManager.listen(mPhoneStateListener,
PhoneStateListener.LISTEN_NONE);
            try{
                sleep(1000);
            }catch (Exception e){
                e.printStackTrace();
```

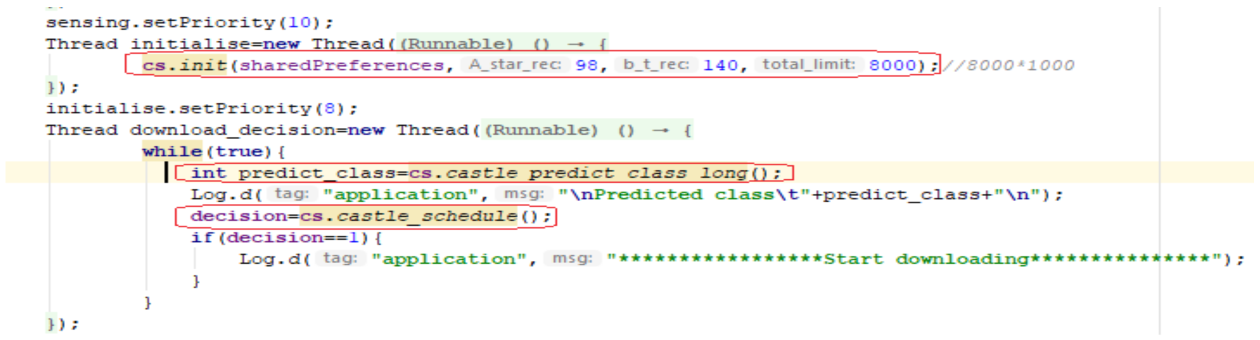
```
                    }
                }
            }
 });
```

In our experimental set up we used ftp operations to trteive files from FTP server. Hence once you have download schedule permission you can download files using common-net libraries.

http://apache.mirrors.ionfish.org//commons/net/binaries/commons-net-3.6-bin.tar.gz

Example

We have created the example application which uses the castle library by using few lines of code:



```
sensing.setPriority(10);
Thread initialise=new Thread((Runnable) () → {
        cs.init(sharedPreferences, A_star_rec: 98, b_t_rec: 140, total_limit: 8000);//8000*1000
});
initialise.setPriority(8);
Thread download_decision=new Thread((Runnable) () → {
        while(true){
            int predict_class=cs.castle_predict_class_long();
            Log.d( tag: "application", msg: "\nPredicted class\t"+predict_class+"\n");
            decision=cs.castle_schedule();
            if(decision==1){
                Log.d( tag: "application", msg: "*****************Start downloading***************");
            }
        }
});
```

Find the same application in https://github.com/cu-pscr/CASTLE_example.git