CS390MB: Mobile Health Sensing and Monitoring

Schedule of Classes

The schedule below shows a rough week-level breakdown of the class. There may be some changes as the course proceeds, but we will try to roughly follow this sequence. There are eight assignments which will be released here, as well as around six multiple choice homeworks. The reading that you need for this class are all in the directory below. There are eight chapters that we will try to finish, not necessarily in-order.

Download All Chapters

Client Data Visualization I

- ### Assignment Description and Goals

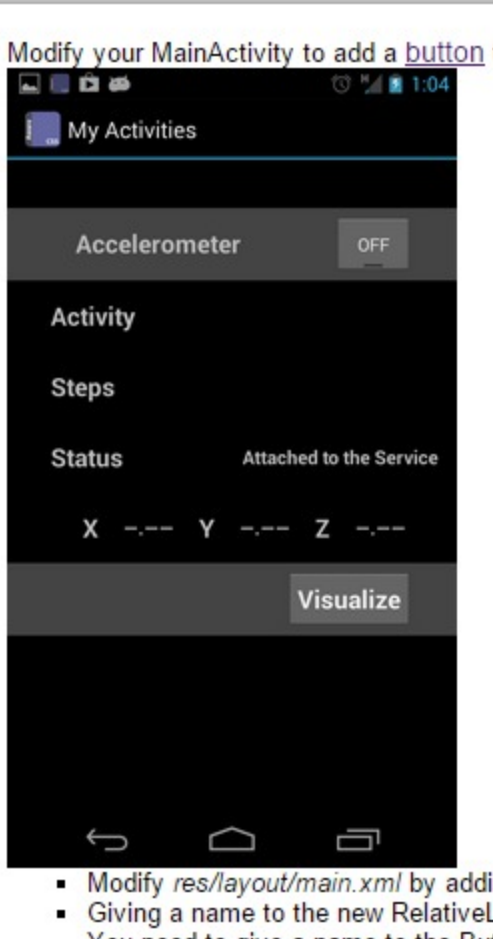
  So far, you have developed classifiers for step counting and activity recognition. In this assignment, you will visualize these various data streams, including features, live on the android phone. You are encouraged to be creative with this assignment. Feel free to create your own views to visualize the data. However, at a minimum your finished app should be able to do the following:

  - Ability to choose which streams to visualize. Steps, Activity and Activity Features such as acceleration values in X, Y and Z axes.
  - Ability to view the history of the streams the user picks
  - Once the user selects the streams to visualize, the app should remember them

  The following steps are simply guidelines that will help you along the way. The code given have comments in them that will also help you. This project is likely to take you a while especially since you will be playing with lots of different Android APIs that you have not encountered before. So **start early!** You will very likely need to read the Android documentation online from time to time to find out more information on various APIs. As always, clarifications can be posted on Piazza.

- ### Assignment Details

1. Modify your MainActivity to add a button to launch the visualizer.



   - Modify *res/layout/main.xml* by adding a Button in a new RelativeLayout underneath the AccelLayout RelativeLayout element
   - Giving a name to the new RelativeLayout is not necessary but is good practice
   - You need to give a name to the Button so that you can attach an *OnClickEventListener*
   - Add code to inflate and attach an *OnClickEventListener* to the button in *MainActivity.java*

```
vizButton.setOnClickListener(new OnClickListener() {
                        @Override
                        public void onClick(View v) {
                                Intent intent = new Intent(getApplicationContext(),ContextActivity.class);
                                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_SINGLE_TOP);
                                startActivity(intent);
                        }
                });
```

2. Create a new ListActivity (*ContextActivity.java*) to show the graphs you want to visualize.

   - You can use this as a starting point. This won't compile until you add missing components.
   - Download WidgetBase.java and ContextImageWidget.java
   - You will need to complete *ContextImageWidget.java* so that it draws correctly, as we discussed in class.
   - Your widgets need to receive data from the background service. So, follow the communication protocol implemented in *MainActivity.java*. In particular, Make sure you 1) create a connection to the ContextService in *onResume()* if not already created, and that you register as a client once the connection is established; 2) unregister as client when *onPause* is called and unbind from the service. (Recall that before an activity is displayed, its *onResume()* method is called, and before a view disappears, its *onPause()* is called.)
   - Add a menu item to the activity so that you can launch a new Activity to pick which streams to visualize. More information can be found here. You need to create a new xml file for the new menu item *res/menu/context_menu.xml*:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
    <item android:id="@+id/action_pick"
          android:icon="@drawable/ic_menu_refresh"
          android:title="@string/action_pick"/>

</menu>
```

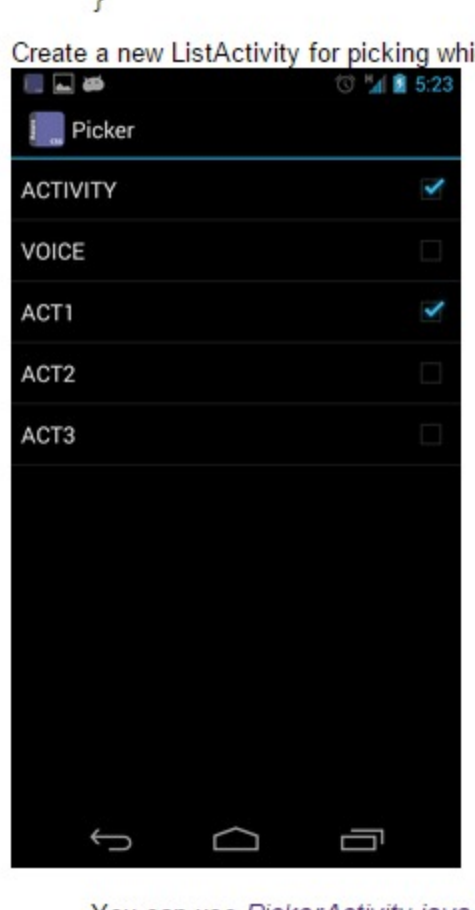
   Inflate the menu bar in your activity by overriding

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu items for use in the action bar
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.context_menu, menu);
        return super.onCreateOptionsMenu(menu);
    }
```

   You have to launch the picker activity once the button is clicked. Follow the example used to launch *ContextActivity* from *MainActivity* to implement this

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle presses on the action bar items
    switch (item.getItemId()) {
        case R.id.action_pick:
            //TODO:: launch picker activity
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

3. Create a new ListActivity for picking which streams to visualize



   - You can use *PickerActivity.java* file as a starting point
   - You need to finish the *onItemClick()* method. You will have to store the selection made in this activity so that you can use the selection to populate the ContextActivity. Refer to the ListView documentation to figure out how to 1) obtain the selected items, and 2) preselect items when the view loads, if items have been previously set by the user. In particular check out *getCheckedItemIds* and *setItemChecked*. To keep things simple, you may create a public static variable in Context_Service to store the selection.
4. Write a new *ContinuousContextImageWidget.java* that extends *ContextImageWidget.java* to work for continuous valued streams as discussed in class.
5. Now that you have everything working for visualizing Activity, add to *ContextActivity.STREAMS* a few features (at least 3) from your activity classifier. Remember, you will need to modify your *Context_Service* so that it publishes the features you decide to include. This means extending the communication interface between the service and the activity.
6. Submission - Upload your project in a zipped file to moodle by Friday, Oct 31 midnight. Please also demonstrate your app to one of the TAs.
7. UPDATE: icon files. ic_menu_refresh.png stat.png walk.png drive.png