```
 1  package edu.seaaddicts.brockbutler.scheduler;
 2
 3  import java.util.ArrayList;
 4  import java.util.Calendar;
 5
 6  import android.app.Activity;
 7  import android.app.AlertDialog;
 8  import android.content.Context;
 9  import android.content.DialogInterface;
10  import android.content.Intent;
11  import android.graphics.Paint;
12  import android.os.Bundle;
13  import android.util.Log;
14  import android.view.Menu;
15  import android.view.MenuItem;
16  import android.view.View;
17  import android.view.View.OnClickListener;
18  import android.view.ViewGroup;
19  import android.widget.AdapterView;
20  import android.widget.ArrayAdapter;
21  import android.widget.CheckBox;
22  import android.widget.CompoundButton;
23  import android.widget.CompoundButton.OnCheckedChangeListener;
24  import android.widget.ImageButton;
25  import android.widget.LinearLayout;
26  import android.widget.ListView;
27  import android.widget.TextView;
28  import android.widget.Toast;
29  import edu.seaaddicts.brockbutler.R;
30  import edu.seaaddicts.brockbutler.animation.ExpandAnimation;
31  import edu.seaaddicts.brockbutler.coursemanager.Course;
32  import edu.seaaddicts.brockbutler.coursemanager.CourseHandler;
33  import edu.seaaddicts.brockbutler.coursemanager.Offering;
34  import edu.seaaddicts.brockbutler.coursemanager.OfferingTime;
35  import edu.seaaddicts.brockbutler.help.HelpActivity;
36
37  public class SchedulerActivity extends Activity {
38    private static final String TAG = "SchedulerActivity";
39
40    private static final int VISIBLE = 0;
41    private static final int GONE = 8;
42
43    private ArrayList<Course> mRegisteredCoursesList = null;
44    private CourseHandler mCourseHandle = null;
45    private ListView mRegisteredCoursesListView = null;
46    ArrayAdapter<String> mListAdaptor;
47    View mView;
48    int mCurTaskTextViewId;
49    int mCurTaskCheckBoxId;
50    int mCurTaskNum;
51    ArrayList<Task> mCurTasks;
52
53    @Override
54    protected void onCreate(Bundle savedInstanceState) {
55      super.onCreate(savedInstanceState);
56      setContentView(R.layout.activity_scheduler);
57      mCourseHandle = new CourseHandler(this);
58      populateCoursesLayout();
59    }
60
61    @Override
62    protected void onResume() {
63      super.onResume();
64      populateCoursesLayout();
65    }
66
67    /**
68     *
69     */
70    private void populateCoursesLayout() {
71      TextView tvNoCourses = (TextView) findViewById(R.id.tv_no_courses);
```

```
72        mRegisteredCoursesListView = (ListView) findViewById(R.id.sched_list);
73        mRegisteredCoursesList = mCourseHandle.getRegisteredCourses();
74        if (mRegisteredCoursesList.size() == 0) {
75          // There are no registered courses so set message.
76          mRegisteredCoursesListView.setVisibility(GONE);
77          tvNoCourses.setVisibility(VISIBLE);
78        } else {
79          // We have registered courses so populate ListView.
80          // Creating the list adapter and populating the list
81          mListAdaptor = new CustomListAdapter(this, R.layout.sched_list_item);
82          for (int i = 0; i < mRegisteredCoursesList.size(); i++)
83            mListAdaptor.add(mRegisteredCoursesList.get(i).mSubject + " "
84                + mRegisteredCoursesList.get(i).mCode);
85          mRegisteredCoursesListView.setAdapter(mListAdaptor);
86          tvNoCourses.setVisibility(GONE);
87          mRegisteredCoursesListView.setVisibility(VISIBLE);
88          mRegisteredCoursesListView
89              .setOnItemClickListener(new AdapterView.OnItemClickListener() {
90                public void onItemClick(AdapterView<?> parent,
91                    final View view, int position, long id) {
92                  showHideToolbar(view, position);
93                }
94              });
95          registerForContextMenu(mRegisteredCoursesListView);
96        }
97      }
98
99      private void populateCourseView(View view, int position) {
100
101       mView = view;
102       // Get current course info.
103       String offering = "";
104       ArrayList<Offering> offs = mRegisteredCoursesList.get(position).mOfferings;
105       ArrayList<OfferingTime> offTimes;
106       ArrayList<Task> tasks = mRegisteredCoursesList.get(position).mTasks;
107       Log.d(TAG, "NUMBER OF TASKS: " + tasks.size());
108
109       // Set Instructor
110       ((TextView) view.findViewById(R.id.tv_prof_name))
111           .setText(mRegisteredCoursesList.get(position).mInstructor);
112
113       // Add Offerings registered for.
114       for (int i = 0; i < offs.size(); i++) {
115         String what = offs.get(i).mType.substring(0, 3).trim();
116         offTimes = offs.get(i).mOfferingTimes;
117         offering = "";
118
119         // Loop through OfferingTimes for each Offering to populate
120         for (int j = 0; j < offTimes.size(); j++) {
121           offering += offTimes.get(j).mDay + " "
122               + offTimes.get(j).mStartTime + " - "
123               + offTimes.get(j).mEndTime + " @ "
124               + offTimes.get(j).mLocation + "\n";
125
126           // Check for type of Offering and add as appropriate
127           if (what.equalsIgnoreCase("lec")) {
128             TextView tv = ((TextView) view
129                 .findViewById(R.id.tv_lecture));
130             tv.setText(offering);
131           }
132
133           // ...a lab?
134           else if (what.equalsIgnoreCase("lab"))
135             ((TextView) view.findViewById(R.id.tv_lab))
136                 .setText(offering);
137
138           // ..a tutorial?
139           else if (what.equalsIgnoreCase("tut"))
140             ((TextView) view.findViewById(R.id.tv_tutorial))
141                 .setText(offering);
142
```

```
143           // ...a seminar?
144           else if (what.equalsIgnoreCase("sem"))
145             ((TextView) view.findViewById(R.id.tv_seminar))
146                 .setText(offering);
147         }
148       }
149
150       /*
151        * Add Tasks to view
152        */
153       if (tasks.size() > 0) {
154         mCurTasks = tasks;
155         // Set visibility for Tasks
156         view.findViewById(R.id.sched_tasks_title).setVisibility(VISIBLE);
157         view.findViewById(R.id.sched_tasks).setVisibility(VISIBLE);
158         ((LinearLayout) view.findViewById(R.id.sched_tasks))
159             .removeAllViews();
160         for (int j = 0; j < tasks.size(); j++) {
161           mCurTaskNum = j;
162           // LinearLayout for each task
163           LinearLayout ll = new LinearLayout(this);
164
165           // TextView for task info
166           TextView tv = new TextView(this);
167
168           ll.addView(tv);
169
170           // Each TextView will need an ID when added to the layout
171           mCurTaskTextViewId = 1000+j;
172           tv.setId(mCurTaskTextViewId);
173
174           // Cosmetics
175           tv.setPadding(20, 20, 30, 20);
176           tv.setLayoutParams(new LinearLayout.LayoutParams(425,
177               LinearLayout.LayoutParams.WRAP_CONTENT));
178
179           // Set the TextView
180           tv.setText(tasks.get(j).mName + "\n\tDue: " + tasks.get(j).mDueDate);
181
182           // Checkbox for Task status
183           CheckBox cb = new CheckBox(this);
184
185           // Buttons for Modify/Delete
186           ImageButton ib_mod = new ImageButton(this);
187           ImageButton ib_del = new ImageButton(this);
188           ib_mod.setFocusable(false);
189           ib_mod.setFocusableInTouchMode(false);
190           ib_mod.setBackgroundResource(android.R.drawable.ic_dialog_info);
191           final Task edit = mCurTasks.get(mCurTaskNum);
192           ib_mod.setOnClickListener(new OnClickListener() {
193
194             public void onClick(View v) {
195               modifyTask(edit);
196             }
197           });
198           ib_del.setFocusable(false);
199           ib_del.setFocusableInTouchMode(false);
200           ib_del.setBackgroundResource(android.R.drawable.ic_delete);
201           ib_del.setOnClickListener(new OnClickListener() {
202
203             public void onClick(View v) {
204               displayTaskRemoveDialog(edit);
205             }
206           });
207           ll.addView(ib_del);
208           ll.addView(ib_mod);
209
210
211           // Each TextView will need an ID when added to the layout
212           mCurTaskCheckBoxId = 1000 + j + 1;
213           cb.setId(mCurTaskCheckBoxId);
```

```
214           final Task currentTask = tasks.get(j);
215           final View fView = mView;
216           cb.setOnCheckedChangeListener(new OnCheckedChangeListener() {
217
218             public void onCheckedChanged(CompoundButton buttonView,
219                 boolean isChecked) {
220               Log.d(TAG, "mCurTaskTextViewId: " + (buttonView.getId()-1));
221               TextView tv = ((TextView) fView.findViewById(buttonView.getId()-1));
222               if (isChecked) {
223                 tv.setPaintFlags(tv.getPaintFlags()
224                     | Paint.STRIKE_THRU_TEXT_FLAG);
225                 currentTask.mIsDone = 1;
226                 mCourseHandle.addTask(currentTask);
227               } else {
228                 tv.setPaintFlags(0);
229                 currentTask.mIsDone = 0;
230                 mCourseHandle.addTask(currentTask);
231               }
232             }
233           });
234           cb.setPadding(20, 0, 0, 0);
235           cb.setFocusable(false);
236           cb.setFocusableInTouchMode(false);
237           ll.addView(cb);
238           ((LinearLayout) view.findViewById(R.id.sched_tasks)).addView(ll);
239           cb.setChecked(currentTask.mIsDone != 0);  //HUZZAH!!!
240         }
241       }
242
243       float mark  = mCourseHandle.getMark(mRegisteredCoursesList.get(position));
244       float base  = mCourseHandle.getBase(mRegisteredCoursesList.get(position));
245       float total=0;
246       if(base != 0){
247         total= (mark/base)*100;
248       }
249
250       String grade= ""+(int)mark+"/"+(int)base+" = "+ (int)total+"%";
251       ((TextView) view.findViewById(R.id.course_grade_grade))
252       .setText(grade);
253
254       /*
255        * Hide class type if none available
256        */
257       if (((TextView) view.findViewById(R.id.tv_lecture)).getText()
258           .toString().equalsIgnoreCase("none"))
259         view.findViewById(R.id.row_lec).setVisibility(GONE);
260       if (((TextView) view.findViewById(R.id.tv_lab)).getText().toString()
261           .equalsIgnoreCase("none"))
262         view.findViewById(R.id.row_lab).setVisibility(GONE);
263       if (((TextView) view.findViewById(R.id.tv_tutorial)).getText()
264           .toString().equalsIgnoreCase("none"))
265         view.findViewById(R.id.row_tut).setVisibility(GONE);
266       if (((TextView) view.findViewById(R.id.tv_seminar)).getText()
267           .toString().equalsIgnoreCase("none"))
268         view.findViewById(R.id.row_sem).setVisibility(GONE);
269     }
270
271     private void displayTaskRemoveDialog(final Task t) {
272       AlertDialog.Builder editalert = new AlertDialog.Builder(this);
273
274       editalert.setTitle("Delete Task?");
275       editalert.setMessage("Are you sure you wish to delete this task?");
276
277       editalert.setPositiveButton("Yes",
278           new DialogInterface.OnClickListener() {
279             public void onClick(DialogInterface dialog, int whichButton) {
280               mCourseHandle.removeTask(t);
281               populateCoursesLayout();
282             }
283           });
284       editalert.setNegativeButton("No",
```

```
285           new DialogInterface.OnClickListener() {
286             public void onClick(DialogInterface dialog, int whichButton) {
287               // do nothing
288             }
289           });
290
291       editalert.show();
292     }
293
294     private void showHideToolbar(View view, int position) {
295       View toolbar = view.findViewById(R.id.sched_toolbar);
296       populateCourseView(view, position);
297
298       // Creating the expand animation for the item
299       ExpandAnimation expandAni = new ExpandAnimation(toolbar,
300           ExpandAnimation.ANIMATE_SHORT);
301
302       // Start the animation on the toolbar
303       toolbar.startAnimation(expandAni);
304
305     }
306
307     public void showHelp(MenuItem item) {
308       Intent intent = new Intent(SchedulerActivity.this, HelpActivity.class);
309       Bundle bundle = new Bundle();
310       bundle.putString("activity", "scheduler");
311       intent.putExtras(bundle);
312       startActivity(intent);
313     }
314
315     public void modifyTask(Task task){
316       Intent intent = new Intent(SchedulerActivity.this, ModifyTaskActivity.class);
317       Bundle bundle = new Bundle();
318       bundle.putInt("assign", task.mAssign);
319       bundle.putString("title",task.mName);
320       bundle.putFloat("weight",task.mWeight);
321       bundle.putFloat("base",task.mBase);
322       bundle.putFloat("mark",task.mMark);
323       bundle.putString("course",task.mSubj + " " + task.mCode);
324       bundle.putInt("priority",task.mPriority);
325       bundle.putInt("is_done",task.mIsDone);
326       if (task.mDueDate.equals(getResources().getString(R.string.activity_add_task_date
          ))){
327         final Calendar cal = Calendar.getInstance();
328         bundle.putInt("year",cal.get(Calendar.YEAR));
329         bundle.putInt("month",cal.get(Calendar.MONTH));
330         bundle.putInt("day",cal.get(Calendar.DAY_OF_MONTH));
331       }
332       else {
333         int year = Integer.parseInt(task.mDueDate.substring(0,4));
334         int month = Integer.parseInt(task.mDueDate.substring(5,7));
335         int day = Integer.parseInt(task.mDueDate.substring(8));
336         bundle.putInt("year",year);
337         bundle.putInt("month",month);
338         bundle.putInt("day",day);
339       }
340       intent.putExtras(bundle);
341       startActivity(intent);
342     }
343
344     public void removeTask(Task task){
345       //REMOVE THE TASK
346     }
347
348     @Override
349     public boolean onCreateOptionsMenu(Menu menu) {
350       // Inflate the menu; this adds items to the action bar if it is present.
351       getMenuInflater().inflate(R.menu.activity_scheduler, menu);
352       return true;
353     }
354
```

```
355     @SuppressWarnings("unused")
356     private void sendEmail() {
357       Intent i = new Intent(Intent.ACTION_SEND);
358       i.setType("message/rfc822");
359       try {
360         startActivity(Intent.createChooser(i, "Send mail..."));
361       } catch (android.content.ActivityNotFoundException ex) {
362         Toast.makeText(SchedulerActivity.this,
363             "There are no email clients installed.", Toast.LENGTH_SHORT)
364             .show();
365       }
366     }
367
368     public void addTask(MenuItem item) {
369       Intent i = new Intent(SchedulerActivity.this, AddTaskActivity.class);
370       startActivity(i);
371     }
372
373     /**
374      * A simple implementation of list adapter.
375      */
376     class CustomListAdapter extends ArrayAdapter<String> {
377
378       public CustomListAdapter(Context context, int textViewResourceId) {
379         super(context, textViewResourceId);
380       }
381
382       @Override
383       public View getView(int position, View convertView, ViewGroup parent) {
384
385         if (convertView == null) {
386           convertView = getLayoutInflater().inflate(
387               R.layout.sched_list_item, null);
388         }
389
390         ((TextView) convertView.findViewById(R.id.sched_list_item_title))
391             .setText(getItem(position));
392
393         // Resets the toolbar to be closed
394         View toolbar = convertView.findViewById(R.id.sched_toolbar);
395         ((LinearLayout.LayoutParams) toolbar.getLayoutParams()).bottomMargin = -(toolbar
396             .getHeight());
397         toolbar.setVisibility(View.GONE);
398
399         return convertView;
400       }
401     }
402
403   }
404
```