```
 1  package edu.seaaddicts.brockbutler.coursemanager;
 2
 3  import java.util.ArrayList;
 4
 5  import android.app.Activity;
 6  import android.app.ProgressDialog;
 7  import android.content.Context;
 8  import android.content.Intent;
 9  import android.net.Uri;
10  import android.os.Bundle;
11  import android.os.Handler;
12  import android.util.Log;
13  import android.view.ContextMenu;
14  import android.view.ContextMenu.ContextMenuInfo;
15  import android.view.View.OnClickListener;
16  import android.view.Gravity;
17  import android.view.Menu;
18  import android.view.MenuItem;
19  import android.view.View;
20  import android.view.ViewGroup;
21  import android.widget.AdapterView;
22  import android.widget.ArrayAdapter;
23  import android.widget.Button;
24  import android.widget.LinearLayout;
25  import android.widget.ListView;
26  import android.widget.TextView;
27  import android.widget.Toast;
28  import edu.seaaddicts.brockbutler.R;
29  import edu.seaaddicts.brockbutler.animation.ExpandAnimation;
30  import edu.seaaddicts.brockbutler.help.HelpActivity;
31
32  public class CourseManagerActivity extends Activity {
33    public static final int CODE_COURSE_MODIFIED = 0;
34    public static final int CODE_COURSE_UNMODIFIED = 1;
35    public static final int CODE_ADD_COURSE = 2;
36
37    public static final String CODE_COURSE_SUBJECT = "csubj";
38    public static final String CODE_COURSE_CODE = "ccode";
39    public static final String CODE_COURSE_DESC = "cdesc";
40    public static final String CODE_COURSE_INSTRUCTOR = "cinstruct";
41    public static final String CODE_COURSE_INSTRUCTOR_EMAIL = "cinstructemail";
42    public static final String CODE_COURSE_OFFERINGS = "coffs";
43
44    private static final String TAG = "CourseManagerActivity";
45
46    private static final int VISIBLE = 0;
47    private static final int GONE = 8;
48
49    private ArrayList<Course> mRegisteredCoursesList;
50    private CourseHandler mCourseHandle = null;
51    private ListView mRegisteredCoursesListView = null;
52
53    @Override
54    protected void onCreate(Bundle savedInstanceState) {
55      super.onCreate(savedInstanceState);
56      setContentView(R.layout.activity_coursemanager);
57      mCourseHandle = new CourseHandler(this.getApplicationContext());
58
59      if (mCourseHandle.getSize() < 1) {
60        updateCourseDatabaseFromRegistrar();
61      }
62    }
63
64    @Override
65    protected void onResume() {
66      super.onResume();
67      populateCoursesLayout();
68    }
69
70    /**
71     * Populates the ListView with registered classes and brief details, i.e.
```

```
72       * instructor name and class times.
73       */
74      private void populateCoursesLayout() {
75        TextView tvNoCourses = (TextView) findViewById(R.id.tv_no_courses);
76        mRegisteredCoursesList = mCourseHandle.getRegisteredCourses();
77        mRegisteredCoursesListView = (ListView) findViewById(R.id.course_manager_list);
78        if (mRegisteredCoursesList.size() == 0) {
79          // There are no registered courses so set message.
80          mRegisteredCoursesListView.setVisibility(GONE);
81          tvNoCourses.setVisibility(VISIBLE);
82        } else {
83          // We have registered courses so populate ListView.
84          // Creating the list adapter and populating the list
85          ArrayAdapter<String> listAdapter = new CustomListAdapter(this,
86              R.layout.course_list_item);
87
88          for (int i = 0; i < mRegisteredCoursesList.size(); i++) {
89            listAdapter.add(mRegisteredCoursesList.get(i).mSubject + " "
90                + mRegisteredCoursesList.get(i).mCode);
91
92            for (int j = 0; j < mRegisteredCoursesList.get(i).mOfferings
93                .size(); j++)
94              Log.d(TAG, "Offerings: "
95                  + mRegisteredCoursesList.get(i).mOfferings.get(j));
96
97            Log.d(TAG, "# Offerings: "
98                + mRegisteredCoursesList.get(i).mOfferings.size());
99          }
100         mRegisteredCoursesListView.setAdapter(listAdapter);
101         tvNoCourses.setVisibility(GONE);
102         mRegisteredCoursesListView.setVisibility(VISIBLE);
103         mRegisteredCoursesListView
104             .setOnItemClickListener(new AdapterView.OnItemClickListener() {
105               public void onItemClick(AdapterView<?> parent,
106                   final View view, int position, long id) {
107                 showHideToolbar(view, position);
108               }
109             });
110         registerForContextMenu(mRegisteredCoursesListView);
111       }
112     }
113
114     public void showHelp(MenuItem item) {
115       Intent intent = new Intent(CourseManagerActivity.this,
116           HelpActivity.class);
117       Bundle bundle = new Bundle();
118       bundle.putString("activity", "coursemanager");
119       intent.putExtras(bundle);
120       startActivity(intent);
121     }
122
123     @Override
124     public void onCreateContextMenu(ContextMenu menu, View v,
125         ContextMenuInfo menuInfo) {
126       if (v.getId() == R.id.course_manager_list) {
127         //AdapterView.AdapterContextMenuInfo info =
          (AdapterView.AdapterContextMenuInfo) menuInfo;
128         String[] menuItems = getResources().getStringArray(
129             R.array.course_manager_context_menu);
130         for (int i = 0; i < menuItems.length; i++) {
131           menu.add(Menu.NONE, i, i, menuItems[i]);
132         }
133       }
134     }
135
136     /**
137      * Determines which MenuItem was selected and acts appropriately depending
138      * on choice.
139      */
140     @Override
141     public boolean onContextItemSelected(MenuItem item) {
```

```
142       AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
          item
143           .getMenuInfo();
144       int menuItemIndex = item.getItemId();
145       Log.d(TAG, "" + menuItemIndex);
146       Course thisCourse = mRegisteredCoursesList.get(info.position);
147       switch (menuItemIndex) {
148       case 0:
149
150         // Start Intent with Course as Extra
151         Intent i = new Intent(CourseManagerActivity.this,
152             ModifyCourseActivity.class);
153         i.putExtra(CODE_COURSE_SUBJECT, thisCourse.mSubject);
154         i.putExtra(CODE_COURSE_CODE, thisCourse.mCode);
155         startActivity(i);
156         break;
157       case 1:
158           Course c = mRegisteredCoursesList.get(info.position);
159           mCourseHandle.removeCourse(c);
160       }
161       populateCoursesLayout();
162       return true;
163     }
164
165     @Override
166     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
167       if (resultCode == RESULT_OK) {
168         switch (requestCode) {
169         case (CODE_ADD_COURSE):
170           // Course c = (Course)
171           // data.getSerializableExtra(CODE_COURSE_OBJECT);
172           // Toast.makeText(getApplicationContext(),
173           // c.mSubject + " " + c.mCode + " added.",
174           // Toast.LENGTH_LONG).show();
175           break;
176         default:
177           break;
178         }
179       }
180       if (resultCode == RESULT_OK && requestCode == CODE_COURSE_MODIFIED) {
181         if (data.hasExtra("returnKey1")) {
182           Toast.makeText(this, data.getExtras().getString("returnKey1"),
183               Toast.LENGTH_SHORT).show();
184         }
185       }
186     }
187
188     /**
189      * Shows/hides verbose description of course.
190      *
191      * @param view
192      *            The selected view to hide/show.
193      */
194     private void showHideToolbar(View view, int position) {
195       View toolbar = view.findViewById(R.id.course_manager_toolbar);
196
197       // Get current course info.
198       String subj = mRegisteredCoursesList.get(position).mSubject;
199       String code = mRegisteredCoursesList.get(position).mCode;
200       String offering = "";
201       ArrayList<Offering> offs = mRegisteredCoursesList.get(position).mOfferings;
202       ArrayList<OfferingTime> offTimes;
203
204       // Creating the expand animation for the item
205       ExpandAnimation expandAni = new ExpandAnimation(toolbar,
206           ExpandAnimation.ANIMATE_SHORT);
207
208       // Start the animation on the toolbar
209       toolbar.startAnimation(expandAni);
210
211       ((TextView) view.findViewById(R.id.tv_prof_name))
```

```
212           .setText(mRegisteredCoursesList.get(position).mInstructor);
213       final String e= mRegisteredCoursesList.get(position).mInstructor_email;
214       ((Button) view.findViewById(R.id.prof_email_button))
215       .setOnClickListener(new OnClickListener() {
216         public void onClick(View v) {
217           sendEmail(e);
218         }
219       });
220
221       Log.d(TAG,"Number of Offerings for " + subj + " " + code + ": " + offs.size());
222
223       // Add Offerings registered for.
224       for (int i = 0; i < offs.size(); i++) {
225         String what = offs.get(i).mType.substring(0, 3).trim();
226
227         offTimes = offs.get(i).mOfferingTimes;
228
229         Log.d(TAG, "Offering Type: " + what + ", # " + offTimes.size());
230
231         offering = "";
232
233         // Loop through OfferingTimes for each Offering to populate
234         for (int j = 0; j < offTimes.size(); j++) {
235           offering += offTimes.get(j).mDay + " "
236               + offTimes.get(j).mStartTime + " - "
237               + offTimes.get(j).mEndTime + " @ "
238               + offTimes.get(j).mLocation + "\n";
239
240           // Check for type of Offering and add as appropriate
241           if (what.equalsIgnoreCase("lec")) {
242             TextView tv = ((TextView) view
243                 .findViewById(R.id.tv_lecture));
244             tv.setText(offering);
245           }
246
247           else if (what.equalsIgnoreCase("lab"))
248             ((TextView) view.findViewById(R.id.tv_lab))
249                 .setText(offering);
250
251           else if (what.equalsIgnoreCase("tut"))
252             ((TextView) view.findViewById(R.id.tv_tutorial))
253                 .setText(offering);
254
255           else if (what.equalsIgnoreCase("sem"))
256             ((TextView) view.findViewById(R.id.tv_seminar))
257                 .setText(offering);
258         }
259       }
260
261       float mark  = mCourseHandle.getMark(mRegisteredCoursesList.get(position));
262       float base  = mCourseHandle.getBase(mRegisteredCoursesList.get(position));
263       float total=0;
264       if(base != 0){
265         total= (mark/base)*100;
266       }
267
268       String grade= ""+(int)mark+"/"+(int)base+" = "+ (int)total+"%";
269       ((TextView) view.findViewById(R.id.course_grade_grade))
270       .setText(grade);
271
272       /*
273        * Hide class type if none available
274        */
275       if (((TextView) view.findViewById(R.id.tv_lecture)).getText()
276           .toString().equalsIgnoreCase("none"))
277         view.findViewById(R.id.row_lec).setVisibility(GONE);
278       if (((TextView) view.findViewById(R.id.tv_lab)).getText()
279           .toString().equalsIgnoreCase("none"))
280         view.findViewById(R.id.row_lab).setVisibility(GONE);
281       if (((TextView) view.findViewById(R.id.tv_tutorial)).getText()
282           .toString().equalsIgnoreCase("none"))
```

```
283         view.findViewById(R.id.row_tut).setVisibility(GONE);
284       if (((TextView) view.findViewById(R.id.tv_seminar)).getText()
285           .toString().equalsIgnoreCase("none"))
286         view.findViewById(R.id.row_sem).setVisibility(GONE);
287     }
288
289     @Override
290     public boolean onCreateOptionsMenu(Menu menu) {
291       // Inflate the menu; this adds items to the action bar if it is present.
292       getMenuInflater().inflate(R.menu.activity_coursemanager, menu);
293       return true;
294     }
295
296     /**
297      * A simple implementation of list adapter to populate ListView with
298      * courses.
299      */
300     class CustomListAdapter extends ArrayAdapter<String> {
301
302       public CustomListAdapter(Context context, int textViewResourceId) {
303         super(context, textViewResourceId);
304       }
305
306       @Override
307       public View getView(int position, View convertView, ViewGroup parent) {
308
309         if (convertView == null) {
310           convertView = getLayoutInflater().inflate(
311               R.layout.course_list_item, null);
312         }
313
314         ((TextView) convertView.findViewById(R.id.course_list_item_title))
315             .setText(getItem(position));
316
317         // Resets the toolbar to be closed
318         View toolbar = convertView
319             .findViewById(R.id.course_manager_toolbar);
320         ((LinearLayout.LayoutParams) toolbar.getLayoutParams()).bottomMargin = -50;
321         toolbar.setVisibility(View.GONE);
322         return convertView;
323       }
324     }
325
326     /**
327      * Launches AddCourseActivity as intent.
328      *
329      * @param menu
330      *            MenuItem selected.
331      */
332     public void addCourse(MenuItem menu) {
333       Intent i = new Intent(CourseManagerActivity.this,
334           AddCourseActivity.class);
335       startActivity(i);
336     }
337
338     /**
339      * Allows user to manually fetch course calendar offerings from Registrar
340      *
341      * @param item
342      */
343     public void updateMaster(MenuItem item) {
344       updateCourseDatabaseFromRegistrar();
345     }
346
347     private void sendEmail(String instr_email) {
348       Intent i = new Intent(Intent.ACTION_SENDTO);
349       i.setType("message/rfc822");
350       i.setData(Uri.parse("mailto:"+instr_email));
351       try {
352         startActivity(Intent.createChooser(i, "Send mail..."));
353       } catch (android.content.ActivityNotFoundException ex) {
```

```
354         Toast.makeText(CourseManagerActivity.this,
355             "There are no email clients installed.", Toast.LENGTH_SHORT)
356             .show();
357       }
358     }
359
360     /**
361      * Updates the course calendar offerings master table. Is called at first
362      * run (if table does not exist) and manually when user wishes to check for
363      * updates. Progress bar to prevent hanging on main thread.
364      */
365     private void updateCourseDatabaseFromRegistrar() {
366       final Handler handler = new Handler();
367       final ProgressDialog progressDialog;
368
369       TextView title = new TextView(CourseManagerActivity.this);
370       title.setText(R.string.loading_courses_registrar);
371       title.setGravity(Gravity.FILL);
372
373       TextView msg = new TextView(CourseManagerActivity.this);
374       msg.setText(R.string.loading_courses_registrar_msg);
375       msg.setGravity(Gravity.FILL);
376
377       progressDialog = ProgressDialog.show(this, "Course Timetable Update",
378           "Updating course timetable from registrar. Please be patient.");
379
380       Thread thread = new Thread() {
381         public void run() {
382           mCourseHandle.updateAll();
383           // this will handle the post task.
384           // it will run when the time consuming task get finished
385           handler.post(new Runnable() {
386             public void run() {
387
388               // Update your UI or
389               // do any Post job after the time consuming task
390               // remember to dismiss the progress dialog here.
391               // updateUI();
392               progressDialog.dismiss();
393               populateCoursesLayout();
394             }
395           });
396         }
397       };
398       thread.start();
399     }
400   }
401
```