```java
package edu.seaaddicts.brockbutler.coursemanager;

import java.util.ArrayList;

import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Gravity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import edu.seaaddicts.brockbutler.R;
import edu.seaaddicts.brockbutler.animation.ExpandAnimation;
import edu.seaaddicts.brockbutler.help.HelpActivity;

public class CourseManagerActivity extends Activity {
  public static final int CODE_COURSE_MODIFIED = 0;
  public static final int CODE_COURSE_UNMODIFIED = 1;
  public static final int CODE_ADD_COURSE = 2;

  public static final String CODE_COURSE_SUBJECT = "csubj";
  public static final String CODE_COURSE_CODE = "ccode";
  public static final String CODE_COURSE_DESC = "cdesc";
  public static final String CODE_COURSE_INSTRUCTOR = "cinstruct";
  public static final String CODE_COURSE_INSTRUCTOR_EMAIL = "cinstructemail";
  public static final String CODE_COURSE_OFFERINGS = "coffs";

  private static final String TAG = "CourseManagerActivity";

  private static final int VISIBLE = 0;
  private static final int GONE = 8;

  private ArrayList<Course> mRegisteredCoursesList;
  private CourseHandler mCourseHandle = null;
  private ListView mRegisteredCoursesListView = null;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_coursemanager);
    mCourseHandle = new CourseHandler(this.getApplicationContext());

    if (mCourseHandle.getSize() < 1) {
      updateCourseDatabaseFromRegistrar();
    }
  }

  @Override
  protected void onResume() {
    super.onResume();
    populateCoursesLayout();
  }

  /**
   * Populates the ListView with registered classes and brief details, i.e.
   * instructor name and class times.
   */
  private void populateCoursesLayout() {
    TextView tvNoCourses = (TextView) findViewById(R.id.tv_no_courses);
    mRegisteredCoursesList = mCourseHandle.getRegisteredCourses();
    mRegisteredCoursesListView = (ListView) findViewById(R.id.course_manager_list);
    if (mRegisteredCoursesList.size() == 0) {
      // There are no registered courses so set message.
      mRegisteredCoursesListView.setVisibility(GONE);
      tvNoCourses.setVisibility(VISIBLE);
    } else {
      // We have registered courses so populate ListView.
      // Creating the list adapter and populating the list
      ArrayAdapter<String> listAdapter = new CustomListAdapter(this,
          R.layout.course_list_item);

      for (int i = 0; i < mRegisteredCoursesList.size(); i++) {
        listAdapter.add(mRegisteredCoursesList.get(i).mSubject + " "
            + mRegisteredCoursesList.get(i).mCode);

        for (int j = 0; j < mRegisteredCoursesList.get(i).mOfferings
            .size(); j++)
          Log.d(TAG, "Offerings: "
              + mRegisteredCoursesList.get(i).mOfferings.get(j));

        Log.d(TAG, "# Offerings: "
            + mRegisteredCoursesList.get(i).mOfferings.size());
      }
      mRegisteredCoursesListView.setAdapter(listAdapter);
      tvNoCourses.setVisibility(GONE);
      mRegisteredCoursesListView.setVisibility(VISIBLE);
      mRegisteredCoursesListView
          .setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> parent,
                final View view, int position, long id) {
              showHideToolbar(view, position);
            }
          });
      registerForContextMenu(mRegisteredCoursesListView);
    }
  }

  public void showHelp(MenuItem item) {
    Intent intent = new Intent(CourseManagerActivity.this,
        HelpActivity.class);
    Bundle bundle = new Bundle();
    bundle.putString("activity", "coursemanager");
    intent.putExtras(bundle);
    startActivity(intent);
  }

  @Override
  public void onCreateContextMenu(ContextMenu menu, View v,
      ContextMenuInfo menuInfo) {
    if (v.getId() == R.id.course_manager_list) {
      //AdapterView.AdapterContextMenuInfo info =
      (AdapterView.AdapterContextMenuInfo) menuInfo;
      String[] menuItems = getResources().getStringArray(
          R.array.course_manager_context_menu);
      for (int i = 0; i < menuItems.length; i++) {
        menu.add(Menu.NONE, i, i, menuItems[i]);
      }
    }
  }

  /**
   * Determines which MenuItem was selected and acts appropriately depending
   * on choice.
   */
  @Override
  public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info = (AdapterView.AdapterContextMenuInfo)
    item
        .getMenuInfo();
    int menuItemIndex = item.getItemId();
    Log.d(TAG, "" + menuItemIndex);
    Course thisCourse = mRegisteredCoursesList.get(info.position);
    switch (menuItemIndex) {
    case 0:

      // Start Intent with Course as Extra
      Intent i = new Intent(CourseManagerActivity.this,
          ModifyCourseActivity.class);
      i.putExtra(CODE_COURSE_SUBJECT, thisCourse.mSubject);
      i.putExtra(CODE_COURSE_CODE, thisCourse.mCode);
      startActivity(i);
      break;
    case 1:
        Course c = mRegisteredCoursesList.get(info.position);
        mCourseHandle.removeCourse(c);
    }
    populateCoursesLayout();
    return true;
  }

  @Override
  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK) {
      switch (requestCode) {
      case (CODE_ADD_COURSE):
        // Course c = (Course)
        // data.getSerializableExtra(CODE_COURSE_OBJECT);
        // Toast.makeText(getApplicationContext(),
        // c.mSubject + " " + c.mCode + " added.",
        // Toast.LENGTH_LONG).show();
        break;
      default:
        break;
      }
    }
    if (resultCode == RESULT_OK && requestCode == CODE_COURSE_MODIFIED) {
      if (data.hasExtra("returnKey1")) {
        Toast.makeText(this, data.getExtras().getString("returnKey1"),
            Toast.LENGTH_SHORT).show();
      }
    }
  }

  /**
   * Shows/hides verbose description of course.
   *
   * @param view
   *           The selected view to hide/show.
   */
  private void showHideToolbar(View view, int position) {
    View toolbar = view.findViewById(R.id.course_manager_toolbar);

    // Get current course info.
    String subj = mRegisteredCoursesList.get(position).mSubject;
    String code = mRegisteredCoursesList.get(position).mCode;
    String offering = "";
    ArrayList<Offering> offs = mRegisteredCoursesList.get(position).mOfferings;
    ArrayList<OfferingTime> offTimes;

    // Creating the expand animation for the item
    ExpandAnimation expandAni = new ExpandAnimation(toolbar,
        ExpandAnimation.ANIMATE_SHORT);

    // Start the animation on the toolbar
    toolbar.startAnimation(expandAni);

    ((TextView) view.findViewById(R.id.tv_prof_name))
        .setText(mRegisteredCoursesList.get(position).mInstructor);

    Log.d(TAG,
        "Number of Offerings for " + subj + " " + code + ": "
            + offs.size());

    // Add Offerings registered for.
    for (int i = 0; i < offs.size(); i++) {
      String what = offs.get(i).mType.substring(0, 3).trim();

      offTimes = offs.get(i).mOfferingTimes;

      Log.d(TAG, "Offering Type: " + what + ", # " + offTimes.size());

      offering = "";

      // Loop through OfferingTimes for each Offering to populate
      for (int j = 0; j < offTimes.size(); j++) {
        offering += offTimes.get(j).mDay + " "
            + offTimes.get(j).mStartTime + " - "
            + offTimes.get(j).mEndTime + " @ "
            + offTimes.get(j).mLocation + "\n";

        // Check for type of Offering and add as appropriate
        if (what.equalsIgnoreCase("lec")) {
          TextView tv = ((TextView) view
              .findViewById(R.id.tv_lecture));
          tv.setText(offering);
        }

        else if (what.equalsIgnoreCase("lab"))
          ((TextView) view.findViewById(R.id.tv_lab))
              .setText(offering);

        else if (what.equalsIgnoreCase("tut"))
          ((TextView) view.findViewById(R.id.tv_tutorial))
              .setText(offering);

        else if (what.equalsIgnoreCase("sem"))
          ((TextView) view.findViewById(R.id.tv_seminar))
              .setText(offering);
      }
    }

    float mark  = mCourseHandle.getMark(mRegisteredCoursesList.get(position));
    float base  = mCourseHandle.getBase(mRegisteredCoursesList.get(position));
    float total=0;
    if(base != 0){
      total= (mark/base)*100;
    }

    String grade= ""+(int)mark+"/"+(int)base+" = "+ (int)total+"%";
    ((TextView) view.findViewById(R.id.course_grade_grade))
    .setText(grade);

    /*
     * Hide class type if none available
     */
    if (((TextView) view.findViewById(R.id.tv_lecture)).getText()
        .toString().equalsIgnoreCase("none"))
      view.findViewById(R.id.row_lec).setVisibility(GONE);
    if (((TextView) view.findViewById(R.id.tv_lab)).getText()
        .toString().equalsIgnoreCase("none"))
      view.findViewById(R.id.row_lab).setVisibility(GONE);
    if (((TextView) view.findViewById(R.id.tv_tutorial)).getText()
        .toString().equalsIgnoreCase("none"))
      view.findViewById(R.id.row_tut).setVisibility(GONE);
    if (((TextView) view.findViewById(R.id.tv_seminar)).getText()
        .toString().equalsIgnoreCase("none"))
      view.findViewById(R.id.row_sem).setVisibility(GONE);
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
```

```java
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.activity_coursemanager, menu);
    return true;
  }

  /**
   * A simple implementation of list adapter to populate ListView with
   * courses.
   */
  class CustomListAdapter extends ArrayAdapter<String> {

    public CustomListAdapter(Context context, int textViewResourceId) {
      super(context, textViewResourceId);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

      if (convertView == null) {
        convertView = getLayoutInflater().inflate(
            R.layout.course_list_item, null);
      }

      ((TextView) convertView.findViewById(R.id.course_list_item_title))
          .setText(getItem(position));

      // Resets the toolbar to be closed
      View toolbar = convertView
          .findViewById(R.id.course_manager_toolbar);
      ((LinearLayout.LayoutParams) toolbar.getLayoutParams()).bottomMargin = -50;
      toolbar.setVisibility(View.GONE);
      return convertView;
    }
  }

  /**
   * Launches AddCourseActivity as intent.
   *
   * @param menu
   *            MenuItem selected.
   */
  public void addCourse(MenuItem menu) {
    Intent i = new Intent(CourseManagerActivity.this,
        AddCourseActivity.class);
    startActivity(i);
  }

  /**
   * Allows user to manually fetch course calendar offerings from Registrar
   *
   * @param item
   */
  public void updateMaster(MenuItem item) {
    updateCourseDatabaseFromRegistrar();
  }

  /**
   * Updates the course calendar offerings master table. Is called at first
   * run (if table does not exist) and manually when user wishes to check for
   * updates. Progress bar to prevent hanging on main thread.
   */
  private void updateCourseDatabaseFromRegistrar() {
    final Handler handler = new Handler();
    final ProgressDialog progressDialog;

    TextView title = new TextView(CourseManagerActivity.this);
    title.setText(R.string.loading_courses_registrar);
    title.setGravity(Gravity.FILL);

    TextView msg = new TextView(CourseManagerActivity.this);
    msg.setText(R.string.loading_courses_registrar_msg);
```

```
      msg.setGravity(Gravity.FILL);

      progressDialog = ProgressDialog.show(this, "Course Timetable Update",
          "Updating course timetable from registrar. Please be patient.");

      Thread thread = new Thread() {
        public void run() {
          mCourseHandle.updateAll();
          // this will handle the post task.
          // it will run when the time consuming task get finished
          handler.post(new Runnable() {
            public void run() {

              // Update your UI or
              // do any Post job after the time consuming task
              // remember to dismiss the progress dialog here.
              // updateUI();
              progressDialog.dismiss();
              populateCoursesLayout();
            }
          });
        }
      };
      thread.start();
    }
  }

```