

TheMainActivity.java

```
1. package com.example.baltu.myapplication.MainMenu;
2. import android.app.Activity;
3. import android.content.Context;
4. import android.content.Intent;
5. import android.content.SharedPreferences;
6. import android.database.sqlite.SQLiteDatabase;
7. import android.database.sqlite.SQLiteException;
8. import android.database.sqlite.SQLiteOpenHelper;
9. import android.graphics.Point;
10. import android.net.ConnectivityManager;
11. import android.net.NetworkInfo;
12. import android.os.Bundle;
13. import android.util.Log;
14. import android.view.Display;
15. import android.view.View;
16. import android.view.WindowManager;
17. import android.view.inputmethod.InputMethodManager;
18. import android.widget.EditText;
19. import android.widget.Toast;
20. import android.widget.ViewSwitcher;
21. import com.example.baltu.myapplication.Buses_Schedules;
22. import com.example.baltu.myapplication.Academic_Calendar;
23. import com.example.baltu.myapplication.DB_Constructor.Local_DB_Helper;
24. import com.example.baltu.myapplication.DB_Constructor.User_DB_Helper;
25. import com.example.baltu.myapplication.GPA_Calculator;
26. import com.example.baltu.myapplication.Information_Activity.InfoListActivity;
27. import com.example.baltu.myapplication.News.News_Activity;
28. import com.example.baltu.myapplication.R;
29. import com.example.baltu.myapplication.Settings.Settings_activity;
30. import com.example.baltu.myapplication.Map.University_Map;
31. import com.example.baltu.myapplication.Map.Offline_Map_Activity;
32. import com.example.baltu.myapplication.TimeTables.All_TimeTables_Activity;
33. import com.example.tomek.notepad.MainActivity;
34. import com.scottyab.aescrypt.AESCrypt;
35. import java.io.File;
36. import java.io.FileOutputStream;
37. import java.io.IOException;
38. import java.io.InputStream;
39. import java.io.OutputStream;
40. import java.security.GeneralSecurityException;
41. /*Unfortunately The code was rushed and implemented by only one student* so the focus was on t
   he functionality and no focus on the documentation*/
42. public class TheMainActivity extends Activity {
43.     Point sizes = new Point();
44.     SQLiteDatabase userdatabase;
45.     SQLiteDatabase localdatabase;
46.     Integer Tries = 3;
47.     static String DB_PATH = "/data/data/com.example.baltu.myapplication/databases/";
48.     public static final String DB_FILENAME = "LocalDb.db";
49.     protected void onCreate(Bundle savedInstanceState) {
50.         super.onCreate(savedInstanceState);
51.         setContentView(R.layout.mainmenu2);
52.         findViewById(R.id.img3).setOnClickListener(new View.OnClickListener() {@
53.             Override public void onClick(View v) {
54.                 Intent intents2 = new Intent(TheMainActivity.this, Academic_Calendar.class);
55.                 startActivity(intents2);
56.             }
57.         });
58.         findViewById(R.id.img5).setOnClickListener(new View.OnClickListener() {@
```

```

59.         Override public void onClick(View v) {
60.             Intent calenderintents = new Intent(TheMainActivity.this, Buses_Schedules.class
s);
61.             startActivity(calenderintents);
62.         }
63.     });
64.     findViewById(R.id.img6).setOnClickListener(new View.OnClickListener() {@
65.         Override public void onClick(View v) {
66.             if (isNetworkAvailable()) {
67.                 Intent NewsIntent = new Intent(TheMainActivity.this, News_Activity.class);
//News
68.                 startActivity(NewsIntent);
69.             } else Toast.makeText(TheMainActivity.this, "Not Connected To the Internet", T
oast.LENGTH_SHORT).show();
70.         }
71.     });
72.     findViewById(R.id.img7).setOnClickListener(new View.OnClickListener() {@
73.         Override public void onClick(View v) {
74.             Intent NotesIntent = new Intent(TheMainActivity.this, MainActivity.class); //N
otes
75.             startActivity(NotesIntent);
76.         }
77.     });
78.     findViewById(R.id.img8).setOnClickListener(new View.OnClickListener() {@
79.         Override public void onClick(View v) {
80.             Intent InformationIntent = new Intent(TheMainActivity.this, InfoListActivity.c
lass);
81.             startActivity(InformationIntent);
82.         }
83.     });
84.     findViewById(R.id.img11).setOnClickListener(new View.OnClickListener() {@
85.         Override public void onClick(View v) {
86.             Intent SettingsIntent = new Intent(TheMainActivity.this, Settings_activity.cla
ss);
87.             startActivity(SettingsIntent);
88.         }
89.     });
90.     findViewById(R.id.img2).setOnClickListener(new View.OnClickListener() {@
91.         Override public void onClick(View v) {
92.             Intent TimetablesIntent = new Intent(TheMainActivity.this, All_TimeTables_Acti
vity.class);
93.             startActivity(TimetablesIntent);
94.         }
95.     });
96.     findViewById(R.id.img9).setOnClickListener(new View.OnClickListener() {@
97.         Override public void onClick(View v) {
98.             Intent gpaIntent = new Intent(TheMainActivity.this, GPA_Calculator.class);
99.             startActivity(gpaIntent);
100.        }
101.    });
102.    WindowManager wm = (WindowManager) this.getSystemService(Context.WINDOW_SERVICE
);
103.    Display display = wm.getDefaultDisplay();
104.    display.getSize(sizes);
105.    Log.d("MainMenue", "onCreate");
106.    final SharedPreferences prefs = getSharedPreferences(getString(R.string.prefere
nce_file_key), MODE_PRIVATE);
107.    boolean key = prefs.getBoolean("pass_on", false);
108.    final String password_string = prefs.getString("Password_String", "");
109.    final ViewSwitcher vs = (ViewSwitcher) findViewById(R.id.Switcher);
110.    if (!key) {

```

```

111.         Log.d("Mainmen", "no pass");
112.         vs.showNext();
113.         if (checkDataBase()) {
114.             SQLiteOpenHelper openhelper = new User_DB_Helper(this);
115.             userdatabase = openhelper.getWritableDatabase();
116.             userdatabase.close();
117.             SQLiteOpenHelper localdbhhelper = new Local_DB_Helper(this);
118.             localdatabase = localdbhhelper.getWritableDatabase();
119.             localdatabase.close();
120.             Toast.makeText(this, "not copied", Toast.LENGTH_SHORT).show();
121.         } else {
122.             SQLiteOpenHelper openhelper = new User_DB_Helper(this);
123.             userdatabase = openhelper.getWritableDatabase();
124.             userdatabase.close();
125.             SQLiteOpenHelper localdbhhelper = new Local_DB_Helper(this);
126.             localdatabase = localdbhhelper.getWritableDatabase();
127.             localdatabase.close();
128.             try {
129.                 copyDataBase();
130.                 Toast.makeText(this, "copied", Toast.LENGTH_SHORT).show();
131.             } catch (IOException e) {
132.                 e.printStackTrace();
133.                 Toast.makeText(this, "Error", Toast.LENGTH_SHORT).show();
134.             }
135.         } //Toast.makeText(this, "Created DB", Toast.LENGTH_SHORT).show();
136.     } else {
137.         findViewById(R.id.unlock_bt).setOnClickListener(new View.OnClickListener()
138.         {
139.             @Override public void onClick(View v) {
140.                 EditText pass = (EditText) findViewById(R.id.password_box);
141.                 if (!pass.getText().toString().equals("")) {
142.                     try {
143.                         if (AESCrypt.encrypt("CanTheWorldUnderstand?", "TuNiSbYnIgh
144. t" + pass.getText().toString()).equals(password_string)) {
145.                             SharedPreferences.Editor PrefE = prefs.edit();
146.                             PrefE.putBoolean("Logged", true);
147.                             PrefE.commit();
148.                             Log.i("logged", "true");
149.                             View view = TheMainActivity.this.getCurrentFocus();
150.                             if (view != null) {
151.                                 InputMethodManager imm = (InputMethodManager) getSy
152. stemService(Context.INPUT_METHOD_SERVICE);
153.                                 imm.hideSoftInputFromWindow(view.getWindowToken(),
154. 0);
155.                             }
156.                             pass.setText("");
157.                             vs.showNext();
158.                         } else {
159.                             View view = TheMainActivity.this.getCurrentFocus();
160.                             if (view != null) {
161.                                 InputMethodManager imm = (InputMethodManager) getSy
162. stemService(Context.INPUT_METHOD_SERVICE);
163.                                 imm.hideSoftInputFromWindow(view.getWindowToken(),
164. 0);
165.                             }
166.                             Tries--;
167.                             if (Tries == 0) finish();
168.                             else {
169.                                 Toast.makeText(TheMainActivity.this, " Wrong Pas
170. sword" + "\nYou Have " + Integer.toString(Tries) + " More Tries", Toast.LENGTH_SHORT).show();

```

```

164.                pass.setText("");
165.            }
166.        }
167.        } catch (GeneralSecurityException e) {
168.            e.printStackTrace();
169.        }
170.    } else Toast.makeText(TheMainActivity.this, "Enter Password", Toast
.LENGTH_SHORT).show();
171.    }
172.    });
173.    }
174.    }@
175.    Override protected void onStart() {
176.        super.onStart();
177.        EditText pass = (EditText) findViewById(R.id.password_box);
178.        pass.setText("");
179.    }
180.    public void clickonMap(View view) {
181.        if (isNetworkAvailable()) {
182.            Intent intf = new Intent(this, University_Map.class);
183.            intf.putExtra("type", 0);
184.            startActivity(intf);
185.        } else {
186.            Toast.makeText(this, "connect to the internet for an interactive map", Toas
t.LENGTH_LONG).show();
187.            Intent offlineMap = new Intent(TheMainActivity.this, Offline_Map_Activity.c
lass);
188.            startActivity(offlineMap);
189.        }
190.    }
191.    private boolean isNetworkAvailable() {
192.        ConnectivityManager connectivityManager = (ConnectivityManager) getSystemServic
e(Context.CONNECTIVITY_SERVICE);
193.        NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
194.        return activeNetworkInfo != null && activeNetworkInfo.isConnected();
195.    }
196.    private void copyDataBase() throws IOException {
197.        try {
198.            InputStream mInputStream = this.getAssets().open("firstdb.db");
199.            String outFileName = DB_PATH + DB_FILENAME;
200.            OutputStream mOutputStream = new FileOutputStream(outFileName);
201.            byte[] buffer = new byte[1024];
202.            int length;
203.            while ((length = mInputStream.read(buffer)) > 0) {
204.                mOutputStream.write(buffer, 0, length);
205.            }
206.            mOutputStream.flush();
207.            mOutputStream.close();
208.            mInputStream.close();
209.        } catch (Exception e) {
210.            e.printStackTrace();
211.        }
212.    }
213.    private boolean checkDataBase() {
214.        try {
215.            final String mPath = DB_PATH + DB_FILENAME;
216.            final File file = new File(mPath);
217.            if (file.exists()) return true;
218.            else return false;
219.        } catch (SQLiteException e) {
220.            e.printStackTrace();

```

```

221.         return false;
222.     }
223. }
224. }

```

All_TimeTables_Activity.java

```

1. package com.example.baltu.myapplication.TimeTables;
2. import android.app.Activity;
3. import android.content.Intent;
4. import android.os.Bundle;
5. import android.support.annotation.Nullable;
6. import android.support.v7.widget.RecyclerView;
7. import android.view.View;
8. import android.widget.AdapterView;
9. import android.widget.AdapterView;
10. import android.widget.AdapterView;
11. import android.widget.AdapterView;
12. import android.widget.AdapterView;
13. import android.widget.AdapterView;
14. import android.widget.AdapterView;
15. import android.widget.AdapterView;
16. import android.widget.AdapterView;
17. import android.widget.AdapterView;
18. import android.widget.AdapterView;
19. import android.widget.AdapterView;
20. import android.widget.AdapterView;
21. import android.widget.AdapterView;
22. import android.widget.AdapterView;
23. import android.widget.AdapterView;
24. import android.widget.AdapterView;
25. import android.widget.AdapterView;
26. import android.widget.AdapterView;
27. import android.widget.AdapterView;
28. import android.widget.AdapterView;
29. import android.widget.AdapterView;
30. import android.widget.AdapterView;
31. import android.widget.AdapterView;
32. import android.widget.AdapterView;
33. import android.widget.AdapterView;
34. import android.widget.AdapterView;
35. import android.widget.AdapterView;
36. import android.widget.AdapterView;
37. import android.widget.AdapterView;
38. import android.widget.AdapterView;
39. import android.widget.AdapterView;
40. import android.widget.AdapterView;
41. import android.widget.AdapterView;
42. import android.widget.AdapterView;
43. import android.widget.AdapterView;
44. import android.widget.AdapterView;
45. import android.widget.AdapterView;
46. import android.widget.AdapterView;
47. import android.widget.AdapterView;
48. import android.widget.AdapterView;

```

```

49.         final TabHost uvo = (TabHost) findViewById(R.id.upcomingvsold2);
50.         uvo.setup();
51.         final TabHost.TabSpec spec11 = uvo.newTabSpec("Upcoming").setContent(R.id.tab11).setIndicator("Upcoming");
52.         uvo.addTab(spec11);
53.         TabHost.TabSpec spec21 = uvo.newTabSpec("Old").setContent(R.id.tab21).setIndicator("Old");
54.         uvo.addTab(spec21);
55.         findViewById(R.id.addnewbt).setOnClickListener(new View.OnClickListener() {
56.             Override public void onClick(View v) {
57.                 if (tabletypes.getSelectedItemPosition() == 0) {
58.                     Intent calenderintents = new Intent(All_TimeTables_Activity.this, Courses_Viewer_Editor_Activity.class);
59.                     calenderintents.putExtra("day", th.getCurrentTab());
60.                     calenderintents.putExtra("name", "");
61.                     startActivity(calenderintents);
62.                 } else if (tabletypes.getSelectedItemPosition() == 1) {
63.                     Intent newtask = new Intent(All_TimeTables_Activity.this, Tasks_Viewer_Editor_Activity.class);
64.                     startActivity(newtask);
65.                 } else if (tabletypes.getSelectedItemPosition() == 2) {
66.                     Intent newexam = new Intent(All_TimeTables_Activity.this, Exams_Viewer_Editor_Activity.class);
67.                     startActivity(newexam);
68.                 }
69.             }
70.         });
71.         thisdaycourses = mDatasource.getcourses(th.getCurrentTab());
72.         courseslistsetup(thisdaycourses);
73.         th.setOnTabChangeListener(new TabHost.OnTabChangeListener() {
74.             Override public void onTabChanged(String tabId) {
75.                 thisdaycourses = mDatasource.getcourses(th.getCurrentTab());
76.                 courseslistsetup(thisdaycourses);
77.             }
78.         });
79.         uvo.setOnTabChangeListener(new TabHost.OnTabChangeListener() {
80.             Override public void onTabChanged(String tabId) {
81.                 if (tabletypes.getSelectedItemPosition() == 1) {
82.                     allTasks = mDatasource.GetTasks(uvo.getCurrentTab());
83.                     Taskslistsetup(allTasks);
84.                 } else if (tabletypes.getSelectedItemPosition() == 2) {
85.                     allExams = mDatasource.GetExams(uvo.getCurrentTab());
86.                     Examslistsetup(allExams);
87.                 }
88.             }
89.         });
90.         tabletypes.setOnItemClickListener(new AdapterView.OnItemClickListener() {
91.             Override public void onItemClick(AdapterView <? > parent, View view, int position, long id) {
92.                 View hs = findViewById(R.id.hscrollc);
93.                 View upvol = findViewById(R.id.upcomingvsold);
94.                 if (position != 0) {
95.                     hs.setVisibility(View.GONE);
96.                     upvol.setVisibility(View.VISIBLE);
97.                 } else {
98.                     hs.setVisibility(View.VISIBLE);
99.                     thisdaycourses = mDatasource.getcourses(th.getCurrentTab());
100.                    courseslistsetup(thisdaycourses);
101.                    upvol.setVisibility(View.GONE);
102.                }
103.                if (position == 1) {

```

```

104.                allTasks = mDatasource.GetTasks(uvo.getCurrentTab());
105.                Taskslistsetup(allTasks);
106.            } else if (position == 2) {
107.                allExams = mDatasource.GetExams(uvo.getCurrentTab());
108.                Examslistsetup(allExams);
109.            }
110.        }@
111.        Override public void onNothingSelected(AdapterView <? > parent) {
112.            /*long tm=new Date().getTime();                Random rand = new Random
(tm);                int r = rand.nextInt(256);                int g = rand.nextInt(256);
                int b = rand.nextInt(256);                th.getCurrentTabView().setBackgroundCo
lor(Color.argb(100,r,g,b));                th.getCurrentView().setBackgroundColor(Color.argb(5
0,r,g,b));*/
113.        }
114.    });
115.    }@
116.    Override protected void onPause() {
117.        super.onPause();
118.        mDatasource.close();
119.    }@
120.    Override protected void onResume() {
121.        super.onResume();
122.        TabHost th = (TabHost) findViewById(R.id.tabhost);
123.        final TabHost uvo = (TabHost) findViewById(R.id.upcomingvsold2);
124.        mDatasource.open();
125.        Spinner tabletypes = (Spinner) findViewById(R.id.tablestype);
126.        if (tabletypes.getSelectedItemPosition() == 0) {
127.            thisdaycourses = mDatasource.getcourses(th.getCurrentTab());
128.            courseslistsetup(thisdaycourses);
129.        } else if (tabletypes.getSelectedItemPosition() == 1) {
130.            allTasks = mDatasource.GetTasks(uvo.getCurrentTab());
131.            Taskslistsetup(allTasks);
132.        } else {
133.            allExams = mDatasource.GetExams(uvo.getCurrentTab());
134.            Examslistsetup(allExams);
135.        }
136.    }
137.    private void courseslistsetup(List < TimeTable_Classes > x) {
138.        View v = findViewById(R.id.crecycler);
139.        RecyclerView crn = (RecyclerView) v;
140.        CoursesItemAdapter cadpter = new CoursesItemAdapter(this, x);
141.        crn.setAdapter(cadpter);
142.    }
143.    private void Taskslistsetup(List < Tasks_Data > x) {
144.        View v = findViewById(R.id.crecycler); //assert (v!=null);
145.        RecyclerView crn = (RecyclerView) v;
146.        TasksItemsAdapter cadpter = new TasksItemsAdapter(this, x);
147.        crn.setAdapter(cadpter);
148.    }
149.    private void Examslistsetup(List < Exam_class > x) {
150.        View v = findViewById(R.id.crecycler); //assert (v!=null);
151.        RecyclerView crn = (RecyclerView) v;
152.        ExamsItemsAdapter cadpter = new ExamsItemsAdapter(this, x);
153.        crn.setAdapter(cadpter);
154.    }
155.    }

```

Courses_Viewer_Editor_Activity.java

```

1. package com.example.baltu.myapplication.TimeTables;
2. import android.app.Activity;
3. import android.app.Dialog;

```



```

4. import android.app.AlertDialog;
5. import android.app.TimePickerDialog;
6. import android.content.DialogInterface;
7. import android.database.sqlite.SQLiteException;
8. import android.os.Bundle;
9. import android.support.annotation.Nullable;
10. import android.support.v7.app.AlertDialog;
11. import android.view.View;
12. import android.widget.AdapterView;
13. import android.widget.Spinner;
14. import android.widget.TextView;
15. import android.widget.TimePicker;
16. import android.widget.Toast;
17. import android.widget.ViewSwitcher;
18. import com.example.baltu.myapplication.DataTypes.TimeTable_Classes;
19. import com.example.baltu.myapplication.Data_Provider.UserDB_Provider;
20. import com.example.baltu.myapplication.R;
21. import java.text.DateFormat;
22. import java.text.ParseException;
23. import java.text.SimpleDateFormat;
24. import java.util.Calendar;
25. /** * Created by Baltu on 2017-05-07. */
26. public class Courses_ViewEditor_Activity extends Activity {
27.     private UserDB_Provider mDatasource;
28.     private String Id;
29.     private TextView vnoc;
30.     private TextView DofW;
31.     private TextView vstoc;
32.     private TextView vetoc;
33.     private TextView vnoteofc;
34.     private Spinner tabletypes;
35.     private TextView noc; //name of course
36.     private static TextView sth; //start time hours
37.     private static TextView eth; //end time hours
38.     private TextView nnc;
39.     private ViewSwitcher vsc;
40.     private static DateFormat timeformat = new SimpleDateFormat("HH:mm");@
41.     Override protected void onCreate(@Nullable Bundle savedInstanceState) {
42.         super.onCreate(savedInstanceState);
43.         setContentView(R.layout.new_course2);
44.         vsc = (ViewSwitcher) findViewById(R.id.ViewSwitcherCourses);
45.         mDatasource = new UserDB_Provider(this);
46.         mDatasource.open();
47.         vnoc = (TextView) findViewById(R.id.name); //V name of course
48.         DofW = (TextView) findViewById(R.id.DayOfTheWeek); //V Day of week
49.         vstoc = (TextView) findViewById(R.id.starttimeee); //V start time
50.         vetoc = (TextView) findViewById(R.id.endtimeee); //V end time
51.         vnoteofc = (TextView) findViewById(R.id.notesnewcourseee); //V Notes
52.         tabletypes = (Spinner) findViewById(R.id.daysofweek);
53.         noc = (TextView) findViewById(R.id.editText2); //E name
54.         sth = (TextView) findViewById(R.id.starttime); //E start time
55.         eth = (TextView) findViewById(R.id.endtime); //E end time
56.         nnc = (TextView) findViewById(R.id.notesnewcourse); //E Notes
57.         ArrayAdapter < CharSequence > items = ArrayAdapter.createFromResource(this, R.array.Da
ys_of_Week, android.R.layout.simple_spinner_item);
58.         items.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
59.         tabletypes.setAdapter(items);
60.         if (getIntent().getStringExtra("id") == null) {
61.             vsc.showNext();
62.         } else {
63.             vnoc.setText(getIntent().getStringExtra("name"));

```



```

64.         noc.setText(getIntent().getStringExtra("name"));
65.         DofW.setText(items.getItem(getIntent().getIntExtra("day", 0)));
66.         tabletypes.setSelection(getIntent().getIntExtra("day", 0));
67.         vstoc.setText(getIntent().getStringExtra("start"));
68.         sth.setText(getIntent().getStringExtra("start"));
69.         vetoc.setText(getIntent().getStringExtra("end"));
70.         eth.setText(getIntent().getStringExtra("end"));
71.         vnoteofc.setText(getIntent().getStringExtra("note"));
72.         nnc.setText(getIntent().getStringExtra("note"));
73.     }
74.     findViewById(R.id.button5).setOnClickListener(new View.OnClickListener() {@
75.         Override public void onClick(View v) {
76.             vsc.showNext();
77.         }
78.     });
79.     findViewById(R.id.button6).setOnClickListener(new View.OnClickListener() {@
80.         Override public void onClick(View v) {
81.             new AlertDialog.Builder(Courses_Viewer_Editor_Activity.this).setTitle("Delete
            ").setMessage("Are you sure you want to delete " + vnoc.getText().toString() + "?").setIcon(an
            droid.R.drawable.ic_dialog_alert).setPositiveButton(android.R.string.yes, new DialogInterface.
            OnClickListener() {
82.                 public void onClick(DialogInterface dialog, int whichButton) {
83.                     if (mDatasource.deletecourse(getIntent().getStringExtra("id"))) {
84.                         Toast.makeText(Courses_Viewer_Editor_Activity.this, "Deleted", Toa
            st.LENGTH_LONG).show();
85.                         finish();
86.                     } else {
87.                         Toast.makeText(Courses_Viewer_Editor_Activity.this, "Error", Toast
            .LENGTH_LONG).show();
88.                         finish();
89.                     }
90.                 }
91.             }).setNegativeButton(android.R.string.no, null).show();
92.         }
93.     });
94.     findViewById(R.id.starttimelayout).setOnClickListener(new View.OnClickListener() {@
95.         Override public void onClick(View v) {
96.             Bundle g = new Bundle();
97.             g.putString("time", sth.getText().toString());
98.             DialogFragment df = new timepicker();
99.             df.setArguments(g);
100.            df.show(getFragmentManager(), "timepicker");
101.        }
102.    });
103.    findViewById(R.id.endtimelayout2).setOnClickListener(new View.OnClickListener()
    {@
104.        Override public void onClick(View v) {
105.            Bundle g = new Bundle();
106.            g.putString("time", eth.getText().toString());
107.            DialogFragment df = new timepicker();
108.            df.setArguments(g);
109.            df.show(getFragmentManager(), "timepicker1");
110.        }
111.    });
112.    findViewById(R.id.savecourse).setOnClickListener(new View.OnClickListener() {@
113.        Override public void onClick(View v) {
114.            TimeTable_Classes gh;
115.            gh = new TimeTable_Classes(getIntent().getStringExtra("id"), noc.getTex
            t().toString(), sth.getText().toString(), eth.getText().toString(), tabletypes.getSelectedIte
            mPosition(), nnc.getText().toString());

```

```

116.         if (noc.getText().toString().equals("") || (sth.getText().toString().eq
117. uals("00:00") && eth.getText().toString().equals("00:00"))) {
118.             Toast.makeText(Courses_Viewer_Editor_Activity.this, "Error: Complet
119. e your input", Toast.LENGTH_SHORT).show();
120.         } else if (getIntent().getStringExtra("id") == null) {
121.             try {
122.                 mDatasource.addnewcourse(gh);
123.             } catch (SQLiteException e) {
124.                 e.printStackTrace();
125.             } finally {
126.                 Toast.makeText(Courses_Viewer_Editor_Activity.this, "Saved", To
127. ast.LENGTH_LONG).show();
128.             }
129.             finish();
130.         } else {
131.             try {
132.                 mDatasource.updatecourse(gh);
133.             } catch (SQLiteException e) {
134.                 e.printStackTrace();
135.             } finally {
136.                 Toast.makeText(Courses_Viewer_Editor_Activity.this, "Updated",
137. Toast.LENGTH_SHORT).show();
138.             }
139.             finish();
140.         }
141.     }
142. });
143. }
144. }@
145. Override protected void onPause() {
146.     super.onPause();
147.     mDatasource.close();
148. }
149. public static class timepicker extends DialogFragment implements TimePickerDialog.O
150. nTimeSetListener {
151.     @Override public Dialog onCreateDialog(Bundle savedInstanceState) {
152.         final Calendar c = Calendar.getInstance();
153.         final int Hours = c.get(Calendar.HOUR_OF_DAY);
154.         int Minutes = c.get(Calendar.MINUTE);
155.         if (sth.getText().toString().equals("00:00") && eth.getText().toString().eq
156. uals("00:00")) {
157.             return new TimePickerDialog(getActivity(), this, Hours, Minutes, true);
158.         } else {
159.             DateFormat df = new SimpleDateFormat("HH:mm");
160.             try {
161.                 return new TimePickerDialog(getActivity(), this, df.parse(getArgume
162. nts().getString("time", "00:00")).getHours(), df.parse(getArguments().getString("time", "00:00
163. ").getMinutes(), true);
164.             } catch (ParseException e) {
165.                 e.printStackTrace();
166.                 return new TimePickerDialog(getActivity(), this, Hours, Minutes, tr
167. ue);
168.             }
169.         }
170.     }
171. }@
172. Override public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
173.     TextView sth = (TextView) getActivity().findViewById(R.id.starttime);
174.     TextView eth = (TextView) getActivity().findViewById(R.id.endtime);
175.     if (getTag() == "timepicker") {
176.         sth.setText(String.valueOf(hourOfDay) + ":" + String.valueOf(minute));

```

```

166.         if (eth.getText().toString().equals("00:00")) eth.setText(String.valueOf(
f((hourOfDay + 2) % 24) + ":" + String.valueOf(minute)));
167.     } else {
168.         eth.setText(String.valueOf(hourOfDay) + ":" + String.valueOf(minute));
169.
170.         if (sth.getText().toString().equals("00:00")) sth.setText(String.valueOf(
f((hourOfDay + 22) % 24) + ":" + String.valueOf(minute)));
171.     }
172. }
173. }

```

Exams_Viewer_Editor_Activity.java

```

1. package com.example.baltu.myapplication.TimeTables;
2. import android.content.DialogInterface;
3. import android.content.Intent;
4. import android.database.sqlite.SQLiteException;
5. import android.os.Bundle;
6. import android.provider.CalendarContract;
7. import android.support.annotation.Nullable;
8. import android.support.v4.app.FragmentActivity;
9. import android.support.v7.app.AlertDialog;
10. import android.view.View;
11. import android.widget.TextView;
12. import android.widget.Toast;
13. import android.widget.ViewFlipper;
14. import com.example.baltu.myapplication.DataTypes.Exam_class;
15. import com.example.baltu.myapplication.Data_Provider.UserDB_Provider;
16. import com.example.baltu.myapplication.R;
17. import com.github.jjjobs.slidedatetimepicker.SlideDateTimeListener;
18. import com.github.jjjobs.slidedatetimepicker.SlideDateTimePicker;
19. import java.text.DateFormat;
20. import java.text.ParseException;
21. import java.text.SimpleDateFormat;
22. import java.util.Date;
23. /** * Created by Baltu on 2017-05-09. */
24. public class Exams_Viewer_Editor_Activity extends FragmentActivity {
25.     UserDB_Provider msource;
26.     Override protected void onCreate(@Nullable Bundle savedInstanceState) {
27.         super.onCreate(savedInstanceState);
28.         setContentView(R.layout.new_exam);
29.         final ViewFlipper vf = (ViewFlipper) findViewById(R.id.viewFlipper);
30.         final String id = getIntent().getStringExtra("id");
31.         final TextView cname2 = (TextView) findViewById(R.id.newtaskcn2); //view task name
32.         final TextView cdesc2 = (TextView) findViewById(R.id.newtaskscd2); // new task descrip
tion
33.         final TextView Datet2 = (TextView) findViewById(R.id.newtaskdate2); //new task date
34.         final TextView ntn2 = (TextView) findViewById(R.id.newtasksnote2); //new task note
35.         final DateFormat all2 = new SimpleDateFormat("dd/MM/yyyy hh:mm aaa");
36.         findViewById(R.id.Editebuttun2).setOnClickListener(new View.OnClickListener() {@
37.             Override public void onClick(View v) {
38.                 vf.showNext();
39.             }
40.         });
41.         msource = new UserDB_Provider(this);
42.         msource.open();
43.         final Date nowd = new Date();
44.         final TextView cname = (TextView) findViewById(R.id.newtaskcn); //new task name
45.         final TextView cdesc = (TextView) findViewById(R.id.newtaskscd); // new task descripti
on
46.         final TextView Timet = (TextView) findViewById(R.id.newtasktime); //new task time

```

```

47.     final TextView Datet = (TextView) findViewById(R.id.newtaskdate); //new task date
48.     final TextView ntn = (TextView) findViewById(R.id.newtasksnote); //new task note
49.     final DateFormat tt = new SimpleDateFormat("hh:mm aaa");
50.     final DateFormat dd = new SimpleDateFormat("dd/MM/yyyy");
51.     final DateFormat all = new SimpleDateFormat("dd/MM/yyyyhh:mm aaa");
52.     Timet.setText(tt.format(nowd));
53.     Datet.setText(dd.format(nowd));
54.     if (id != null) {
55.         vf.showNext();
56.         cname2.setText(getIntent().getStringExtra("name"));
57.         cname.setText(getIntent().getStringExtra("name"));
58.         cdesc2.setText(getIntent().getStringExtra("desc"));
59.         cdesc.setText(getIntent().getStringExtra("desc"));
60.         Datet2.setText(all2.format(new Date(getIntent().getLongExtra("date", 0))));
61.         Datet.setText(dd.format(new Date(getIntent().getLongExtra("date", 0))));
62.         Timet.setText(tt.format(new Date(getIntent().getLongExtra("date", 0))));
63.         ntn2.setText(getIntent().getStringExtra("notes"));
64.         ntn.setText(getIntent().getStringExtra("notes"));
65.     }
66.     findViewById(R.id.newtasktime).setOnClickListener(new View.OnClickListener() {@
67.         Override public void onClick(View v) {
68.             SlideDateTimeListener listner = new SlideDateTimeListener() {@
69.                 Override public void onDateTimeSet(Date date) {
70.                     Timet.setText(tt.format(date));
71.                     Datet.setText(dd.format(date));
72.                 }
73.             };
74.             Date nowtime = new Date();
75.             try {
76.                 nowtime = all.parse(Datet.getText().toString() + Timet.getText().toString(
77. ));
78.             } catch (ParseException e) {
79.                 e.printStackTrace();
80.             }
81.             new SlideDatePicker.Builder(getSupportFragmentManager()).setListener(listner).setInitialDate(nowtime).setMaxDate(new Date(nowtime.getTime() + (long) 31536000000.0)).setMinDate(new Date(nowtime.getTime() - (long) 31536000000.0)).setCurrentd_or_t(1).build().show();
82.         }
83.     });
84.     findViewById(R.id.alarm).setOnClickListener(new View.OnClickListener() {@
85.         Override public void onClick(View v) {
86.             Intent intent = new Intent(Intent.ACTION_INSERT);
87.             intent.setType("vnd.android.cursor.item/event");
88.             intent.putExtra(CalendarContract.Events.TITLE, cname2.getText().toString());
89.             intent.putExtra(CalendarContract.Events.DESRIPTION, cdesc2.getText().toString());
90.         }
91.     });
92.     Date taskdate = new Date(); // Setting dates
93.     try {
94.         taskdate = all2.parse(Datet2.getText().toString());
95.     } catch (ParseException e) {
96.         e.printStackTrace();
97.     }
98.     intent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, taskdate.getTime());
99.     intent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME, taskdate.getTime());
100.    startActivity(intent); // make it a full day event // intent.putExtra(CalendarContract.EXTRA_EVENT_ALL_DAY, true); // make it a recurring Event //intent.putExtra(CalendarContract.Events.RRULE, "FREQ=WEEKLY;COUNT=11;WKST=SU;BYDAY=TU,TH"); // Making it private and shown as busy // intent.putExtra(CalendarContract.Events.ACCESS_LEVEL, CalendarContract.Events.AC

```

```

CESS_PRIVATE); // intent.putExtra(CalendarContract.Events.AVAILABILITY, CalendarContract.Event
s.AVAILABILITY_BUSY);
98.    }
99.    });
100.    findViewById(R.id.newtaskdate).setOnClickListener(new View.OnClickListener() {@
101.        Override public void onClick(View v) {
102.            SlideDateTimeListener listner = new SlideDateTimeListener() {@
103.                Override public void onDateTimeSet(Date date) {
104.                    Timet.setText(tt.format(date));
105.                    Datet.setText(dd.format(date));
106.                }
107.            };
108.            Date nowtime = new Date();
109.            try {
110.                nowtime = all.parse(Datet.getText().toString() + Timet.getText().to
String());
111.            } catch (ParseException e) {
112.                e.printStackTrace();
113.            }
114.            new SlideDateTimePicker.Builder(getSupportFragmentManager()).setListene
r(listner).setInitialDate(nowtime).setMaxDate(new Date(nowtime.getTime() + (long) 315360000000
.0)).setMinDate(new Date(nowtime.getTime() - (long) 315360000000.0)).setCurrentd_or_t(0).build
().show();
115.        }
116.    });
117.    findViewById(R.id.newtasksaveb).setOnClickListener(new View.OnClickListener() {
118.        @Override public void onClick(View v) {
119.            Exam_class gh = new Exam_class();
120.            Date isitold = new Date();
121.            try {
122.                isitold = all.parse(Datet.getText().toString() + Timet.getText().to
String());
123.            } catch (ParseException e) {
124.                e.printStackTrace();
125.            }
126.            if (!cname.getText().toString().isEmpty() && isitold.getTime() > new Da
te().getTime()) {
127.                gh.setCourse(cname.getText().toString());
128.                gh.setExam_Date(isitold.getTime());
129.                gh.setDiscription(cdsc.getText().toString());
130.                gh.setNotes(ntn.getText().toString());
131.                if (id == null) {
132.                    try {
133.                        msource.addnewExam(gh);
134.                    } catch (SQLException e) {
135.                        e.printStackTrace();
136.                    }
137.                    Toast.makeText(Exams_Viewer_Editor_Activity.this, "saved", Toas
t.LENGTH_LONG).show();
138.                    finish();
139.                } else {
140.                    gh.setID(id);
141.                    try {
142.                        msource.updateExam(gh);
143.                    } catch (SQLException e) {
144.                        e.printStackTrace();
145.                    }
146.                    Toast.makeText(Exams_Viewer_Editor_Activity.this, "updated", To
ast.LENGTH_LONG).show();

```

```

147.             finish();
148.         }
149.     } else {
150.         Toast.makeText(Exams_Viewer_Editor_Activity.this, "Error\ncheck you
r Inputs", Toast.LENGTH_LONG).show();
151.     }
152. }
153. });
154. findViewById(R.id.deleteb).setOnClickListener(new View.OnClickListener() {@
155.     Override public void onClick(View v) {
156.         new AlertDialog.Builder(Exams_Viewer_Editor_Activity.this).setTitle("De
lete ").setMessage("Are you sure you want to delete " + cname2.getText().toString() + "?").set
Icon(android.R.drawable.ic_dialog_alert).setPositiveButton(android.R.string.yes, new DialogInt
erface.OnClickListener() {
157.             public void onClick(DialogInterface dialog, int whichButton) {
158.                 if (msource.DeleteExam(id) == 1) {
159.                     Toast.makeText(Exams_Viewer_Editor_Activity.this, "Deleted"
, Toast.LENGTH_LONG).show();
160.                     finish();
161.                 } else {
162.                     Toast.makeText(Exams_Viewer_Editor_Activity.this, "Error",
Toast.LENGTH_LONG).show();
163.                     finish();
164.                 }
165.             }
166.         }).setNegativeButton(android.R.string.no, null).show();
167.     }
168. });
169. }@
170. Override protected void onPause() {
171.     super.onPause();
172.     msource.close();
173. }
174. }

```

Tasks_Viewer_Editor_Activity.java

```

1. package com.example.baltu.myapplication.TimeTables;
2. import android.content.DialogInterface;
3. import android.content.Intent;
4. import android.database.sqlite.SQLiteException;
5. import android.os.Bundle;
6. import android.provider.CalendarContract;
7. import android.support.annotation.Nullable;
8. import android.support.v4.app.FragmentActivity;
9. import android.support.v7.app.AlertDialog;
10. import android.view.View;
11. import android.widget.TextView;
12. import android.widget.Toast;
13. import android.widget.ViewFlipper;
14. import com.example.baltu.myapplication.DataTypes.Tasks_Data;
15. import com.example.baltu.myapplication.Data_Provider.UserDB_Provider;
16. import com.example.baltu.myapplication.R;
17. import com.github.jjjobs.slidedatetimerpicker.SlideDateTimeListener;
18. import com.github.jjjobs.slidedatetimerpicker.SlideDateTimePicker;
19. import java.text.DateFormat;
20. import java.text.ParseException;
21. import java.text.SimpleDateFormat;
22. import java.util.Date;
23. /** * Created by Baltu on 2017-05-09. */
24. public class Tasks_Viewer_Editor_Activity extends FragmentActivity {
25.     UserDB_Provider msource;@

```

```

26. Override protected void onCreate(@Nullable Bundle savedInstanceState) {
27.     super.onCreate(savedInstanceState);
28.     setContentView(R.layout.newtasks);
29.     final ViewFlipper vf = (ViewFlipper) findViewById(R.id.viewFlipper);
30.     final String id = getIntent().getStringExtra("id");
31.     final TextView cname2 = (TextView) findViewById(R.id.newtaskcn2); //view task name
32.     final TextView cdesc2 = (TextView) findViewById(R.id.newtaskscd2); // new task descrip
tion
33.     final TextView Datet2 = (TextView) findViewById(R.id.newtaskdate2); //new task date
34.     final TextView ntn2 = (TextView) findViewById(R.id.newtasksnote2); //new task note
35.     final DateFormat all2 = new SimpleDateFormat("dd/MM/yyyy hh:mm aaa");
36.     findViewById(R.id.Editebuttun2).setOnClickListener(new View.OnClickListener() {@
37.         Override public void onClick(View v) {
38.             vf.showNext();
39.         }
40.     });
41.     findViewById(R.id.alarm).setOnClickListener(new View.OnClickListener() {@
42.         Override public void onClick(View v) {
43.             Intent intent = new Intent(Intent.ACTION_INSERT);
44.             intent.setType("vnd.android.cursor.item/event");
45.             intent.putExtra(CalendarContract.Events.TITLE, cname2.getText().toString());
46.             intent.putExtra(CalendarContract.Events.DESRIPTION, cdesc2.getText().toString
());
47.             Date taskdate = new Date(); // Setting dates
48.             try {
49.                 taskdate = all2.parse(Datet2.getText().toString());
50.             } catch (ParseException e) {
51.                 e.printStackTrace();
52.             }
53.             intent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, taskdate.getTime());
54.             intent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME, taskdate.getTime());
55.             startActivity(intent); // make it a full day event // intent.putExtra(Calendar
Contract.EXTRA_EVENT_ALL_DAY, true); // make it a recurring Event //intent.putExtra(CalendarCo
ntract.Events.RRULE, "FREQ=WEEKLY;COUNT=11;WKST=SU;BYDAY=TU,TH"); // Making it private and sho
wn as busy // intent.putExtra(CalendarContract.Events.ACCESS_LEVEL, CalendarContract.Events.AC
CESS_PRIVATE); // intent.putExtra(CalendarContract.Events.AVAILABILITY, CalendarContract.Event
s.AVAILABILITY_BUSY);
56.         }
57.     });
58.     msource = new UserDB_Provider(this);
59.     msource.open();
60.     final Date nowd = new Date();
61.     final TextView cname = (TextView) findViewById(R.id.newtaskcn); //new task name
62.     final TextView cdesc = (TextView) findViewById(R.id.newtaskscd); // new task descripti
on
63.     final TextView Timet = (TextView) findViewById(R.id.newtaskdate); //new task time
64.     final TextView Datet = (TextView) findViewById(R.id.newtasktime); //new task date
65.     final TextView ntn = (TextView) findViewById(R.id.newtasksnote); //new task note
66.     final DateFormat tt = new SimpleDateFormat("hh:mm aaa");
67.     final DateFormat dd = new SimpleDateFormat("dd/MM/yyyy");
68.     final DateFormat all = new SimpleDateFormat("dd/MM/yyyyhh:mm aaa");
69.     Timet.setText(tt.format(nowd));
70.     Datet.setText(dd.format(nowd));
71.     if (id != null) {
72.         vf.showNext();
73.         cdesc2.setText(getIntent().getStringExtra("desc"));
74.         cdesc.setText(getIntent().getStringExtra("desc"));
75.         Datet2.setText(all2.format(new Date(getIntent().getLongExtra("date", 0))));
76.         Datet.setText(dd.format(new Date(getIntent().getLongExtra("date", 0))));
77.         Timet.setText(tt.format(new Date(getIntent().getLongExtra("date", 0))));

```



```

78.         ntn2.setText(getIntent().getStringExtra("notes"));
79.         ntn.setText(getIntent().getStringExtra("notes"));
80.         cname2.setText(getIntent().getStringExtra("name"));
81.         cname.setText(getIntent().getStringExtra("name"));
82.     }
83.     findViewById(R.id.newtasktime).setOnClickListener(new View.OnClickListener() {@
84.         Override public void onClick(View v) {
85.             SlideDateTimeListener listner = new SlideDateTimeListener() {@
86.                 Override public void onDateTimeSet(Date date) {
87.                     Timet.setText(tt.format(date));
88.                     Datet.setText(dd.format(date));
89.                 }
90.             };
91.             Date nowtime = new Date();
92.             try {
93.                 nowtime = all.parse(Datet.getText().toString() + Timet.getText().toString(
94.             ));
95.             } catch (ParseException e) {
96.                 e.printStackTrace();
97.             }
98.             new SlideDateTimePicker.Builder(getSupportFragmentManager()).setListener(listner).setInitialDate(nowtime).setMaxDate(new Date(nowtime.getTime() + (long) 315360000000.0)).setMinDate(new Date(nowtime.getTime() - (long) 315360000000.0)).setCurrentd_or_t(0).build().show();
99.         }
100.    });
101.    findViewById(R.id.newtaskdate).setOnClickListener(new View.OnClickListener() {@
102.        Override public void onClick(View v) {
103.            SlideDateTimeListener listner = new SlideDateTimeListener() {@
104.                Override public void onDateTimeSet(Date date) {
105.                    Timet.setText(tt.format(date));
106.                    Datet.setText(dd.format(date));
107.                }
108.            };
109.            Date nowtime = new Date();
110.            try {
111.                nowtime = all.parse(Datet.getText().toString() + Timet.getText().to
112.            String());
113.            } catch (ParseException e) {
114.                e.printStackTrace();
115.            }
116.            new SlideDateTimePicker.Builder(getSupportFragmentManager()).setListene
117.            r(listner).setInitialDate(nowtime).setMaxDate(new Date(nowtime.getTime() + (long) 315360000000
118.            .0)).setMinDate(new Date(nowtime.getTime() - (long) 315360000000.0)).setCurrentd_or_t(1).build
119.            ().show();
120.        }
121.    });
122.    findViewById(R.id.newtasksaveb).setOnClickListener(new View.OnClickListener() {
123.        @
124.        Override public void onClick(View v) {
125.            Tasks_Data gh = new Tasks_Data();
126.            Date isitold = new Date();
127.            try {
128.                isitold = all.parse(Datet.getText().toString() + Timet.getText().to
129.            String());
130.            } catch (ParseException e) {
131.                e.printStackTrace();
132.            }
133.            if (!cname.getText().toString().isEmpty() && isitold.getTime() > new Da
134.            te().getTime()) {

```

```

127.                gh.setDeadline_Date(isitold.getTime());
128.                gh.setDiscription(cdesc.getText().toString());
129.                gh.setNotes(ntn.getText().toString());
130.                gh.setCourse(cname.getText().toString());
131.                if (id == null) {
132.                    try {
133.                        msource.addnewTask(gh);
134.                    } catch (SQLiteException e) {
135.                        e.printStackTrace();
136.                    }
137.                    Toast.makeText(Tasks_Viewwer_Editor_Activity.this, "saved", Toas
138.                        t.LENGTH_LONG).show();
139.                    finish();
140.                } else {
141.                    gh.setID(id);
142.                    try {
143.                        msource.updateTask(gh);
144.                    } catch (SQLiteException e) {
145.                        e.printStackTrace();
146.                    }
147.                    Toast.makeText(Tasks_Viewwer_Editor_Activity.this, "updated", To
148.                        ast.LENGTH_LONG).show();
149.                    finish();
150.                } else {
151.                    Toast.makeText(Tasks_Viewwer_Editor_Activity.this, "Error\ncheck you
152.                        r Inputs", Toast.LENGTH_LONG).show();
153.                }
154.            });
155.            findViewById(R.id.deleteb).setOnClickListener(new View.OnClickListener() {@
156.                Override public void onClick(View v) {
157.                    new AlertDialog.Builder(Tasks_Viewwer_Editor_Activity.this).setTitle("De
158.                        lete ").setMessage("Are you sure you want to delete " + cname2.getText().toString() + "?").set
159.                        Icon(android.R.drawable.ic_dialog_alert).setPositiveButton(android.R.string.yes, new DialogInt
160.                        erface.OnClickListener() {
161.                            public void onClick(DialogInterface dialog, int whichButton) {
162.                                if (msource.DeleteTask(id) == 1) {
163.                                    Toast.makeText(Tasks_Viewwer_Editor_Activity.this, "Deleted"
164.                                        , Toast.LENGTH_LONG).show();
165.                                    finish();
166.                                } else {
167.                                    Toast.makeText(Tasks_Viewwer_Editor_Activity.this, "Error",
168.                                        Toast.LENGTH_LONG).show();
169.                                    finish();
170.                                }
171.                            }
172.                        }).setNegativeButton(android.R.string.no, null).show();
173.                }
174.            });
175.        }@
176.        Override protected void onPause() {
177.            super.onPause();
178.            msource.close();
179.        }
180.    }

```

Academic_Calendar.java

```

1. package com.example.baltu.myapplication;
2. import android.app.Activity;
3. import android.app.Fragment;

```

```

4. import android.os.Bundle;
5. import android.support.design.widget.FloatingActionButton;
6. import android.support.design.widget.Snackbar;
7. import android.support.v7.app.AppCompatActivity;
8. import android.support.v7.widget.RecyclerView;
9. import android.support.v7.widget.Toolbar;
10. import android.view.View;
11. import com.example.baltu.myapplication.DataTypes.Calender_Data;
12. import com.example.baltu.myapplication.Data_Provider.Adapters.CalendarItemsAdapter;
13. import com.example.baltu.myapplication.Data_Provider.Local_DB_Main_Provider;
14. import java.util.List;
15. public class Academic_Calendar extends Activity {
16.     Local_DB_Main_Provider mSource;
17.     List < Calender_Data > allevents;@
18.     Override protected void onCreate(Bundle savedInstanceState) {
19.         super.onCreate(savedInstanceState);
20.         mSource = new Local_DB_Main_Provider(this);
21.         setContentView(R.layout.activity_calender);
22.         RecyclerView crv = (RecyclerView) findViewById(R.id.calendar_recycler);
23.         allevents = mSource.GetCalendarItems();
24.         CalendarItemsAdapter cda = new CalendarItemsAdapter(this, allevents);
25.         crv.setAdapter(cda);
26.     }
27. }

```

University_Map.java

```

1. package com.example.baltu.myapplication.Map;
2. import android.content.Intent;
3. import android.support.design.widget.FloatingActionButton;
4. import android.support.v4.app.FragmentActivity;
5. import android.os.Bundle;
6. import android.support.v4.widget.DrawerLayout;
7. import android.view.View;
8. import android.widget.AdapterView;
9. import android.widget.ArrayAdapter;
10. import android.widget.ListView;
11. import com.example.baltu.myapplication.DataTypes.University_Locations;
12. import com.example.baltu.myapplication.Data_Provider.Local_DB_Main_Provider;
13. import com.example.baltu.myapplication.Data_Provider.JsonHelper;
14. import com.example.baltu.myapplication.R;
15. import com.google.android.gms.maps.CameraUpdateFactory;
16. import com.google.android.gms.maps.GoogleMap;
17. import com.google.android.gms.maps.OnMapReadyCallback;
18. import com.google.android.gms.maps.SupportMapFragment;
19. import com.google.android.gms.maps.model.LatLng;
20. import com.google.android.gms.maps.model.LatLngBounds;
21. import com.google.android.gms.maps.model.MarkerOptions;
22. import com.google.maps.android.data.Feature;
23. import com.google.maps.android.data.kml.KmlLayer;
24. import org.xmlpull.v1.XmlPullParserException;
25. import java.io.IOException;
26. import java.util.ArrayList;
27. import java.util.HashMap;
28. import java.util.List;
29. import java.util.Map;
30. public class University_Map extends FragmentActivity implements OnMapReadyCallback {
31.     private Map < String, Places_on_map > all_places = new HashMap < > ();
32.     private Map < String, University_Locations > all_locations = new HashMap < > ();
33.     private static final String File_Name = "mapplaces.json";
34.     private GoogleMap mMap;
35.     static final int MY_REQUEST_CODE = 5584;

```

```

36.     private String[] mPlanetTitles;
37.     private DrawerLayout mDrawerLayout;
38.     private ListView mDrawerList;
39.     private FloatingActionButton myfab;
40.     private Local_DB_Main_Provider mSource;;@
41.     Override protected void onCreate(Bundle savedInstanceState) {
42.         super.onCreate(savedInstanceState);
43.         setContentView(R.layout.activity_un__map);
44.         mSource = new Local_DB_Main_Provider(this);
45.         final List < University_Locations > AllLocations = mSource.GetMapLocations();
46.         mPlanetTitles = getResources().getStringArray(R.array.Credits);
47.         mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
48.         mDrawerList = (ListView) findViewById(R.id.left_drawer);
49.         myfab = (FloatingActionButton) findViewById(R.id.floatingActionButtonmap);
50.         mDrawerList.setAdapter(new ArrayAdapter < String > (this, android.R.layout.simple_
list_item_1, mPlanetTitles));
51.         mDrawerList.setOnItemClickListener(new AdapterView.OnItemClickListener() {@
52.             Override public void onItemClick(AdapterView <? > parent, View view, int posit
ion, long id) {
53.                 University_Locations item = AllLocations.get(position);
54.                 Intent loc_info = new Intent(University_Map.this, Locations_on_Map_Details
.class);
55.                 loc_info.putExtra("Location", item.getShortName());
56.                 loc_info.putExtra("Full_Name", item.getLongName());
57.                 loc_info.putExtra("url", item.getImage_Url());
58.                 loc_info.putExtra("Info", item.getInformation());
59.                 loc_info.putExtra("Longt", item.getLongitude());
60.                 loc_info.putExtra("Lat", item.getLatitude());
61.                 mDrawerLayout.closeDrawers();
62.                 startActivityForResult(loc_info, MY_REQUEST_CODE);
63.             }
64.         });
65.         myfab.setOnClickListener(new View.OnClickListener() {@
66.             Override public void onClick(View v) {
67.                 mDrawerLayout.openDrawer(mDrawerList);
68.             }
69.         });
70.         List < String > ListString = new ArrayList < > ();
71.         if (AllLocations != null) {
72.             for (University_Locations n: AllLocations) {
73.                 ListString.add(n.getLongName());
74.                 all_Locations.put(n.getShortName(), n);
75.             }
76.             mDrawerList.setAdapter(new ArrayAdapter < String > (this, android.R.layout.sim
ple_list_item_1, ListString));
77.         }
78.         List < Places_on_map > myplaces = JsonHelper.Importer(University_Map.this);
79.         if (myplaces != null) {
80.             for (Places_on_map pl: myplaces) {
81.                 all_places.put(pl.getPlace_Name(), pl);
82.             }
83.         } // Obtain the SupportMapFragment and get notified when the map is ready to be us
ed.
84.         SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager().
findFragmentById(R.id.map);
85.         mapFragment.getMapAsync(this);
86.     }
87.     /**      * Manipulates the map once available.      * This callback is triggered when th
e map is ready to be used.      * This is where we can add markers or lines, add listeners or m
ove the camera. In this case,      * we just add a marker near Sydney, Australia.      * If Goog
le Play services is not installed on the device, the user will be prompted to install      * it

```

inside the SupportMapFragment. This method will only be triggered once the user has installed Google Play services and returned to the app. */

```
88.     @
89.     Override public void onMapReady(GoogleMap googleMap) {
90.         mMap = googleMap;
91.         int maptype = getIntent().getIntExtra("type", 0);
92.         try {
93.             if (maptype == 0) {
94.                 KmlLayer layer = new KmlLayer(mMap, R.raw.campusmap, getApplicationContext());
95.                 layer.addLayerToMap();
96.                 layer.setOnFeatureClickListener(new KmlLayer.OnFeatureClickListener() {@
97.                     Override public void onFeatureClick(Feature feature) {
98.                         String namec = null;
99.                         try {
100.                             namec = feature.getProperty("name").toString();
101.                         } catch (Exception e) {
102.                             e.printStackTrace();
103.                         }
104.                         if (all_Locations.containsKey(namec)) {
105.                             Intent loc_info = new Intent(University_Map.this, Locations
_on_Map_Details.class);
106.                             loc_info.putExtra("Location", all_Locations.get(namec).getS
hortName());
107.                             loc_info.putExtra("Full_Name", all_Locations.get(namec).get
LongName());
108.                             loc_info.putExtra("url", all_Locations.get(namec).getImage_
Url());
109.                             loc_info.putExtra("Info", all_Locations.get(namec).getInfor
mation());
110.                             loc_info.putExtra("Longt", all_Locations.get(namec).getLong
itude());
111.                             loc_info.putExtra("Lat", all_Locations.get(namec).getLatitu
de());
112.                             startActivityForResult(loc_info, MY_REQUEST_CODE);
113.                         } else if (all_places.containsKey(namec)) {
114.                             Intent loc_info = new Intent(University_Map.this, Locations
_on_Map_Details.class);
115.                             loc_info.putExtra("Location", namec);
116.                             loc_info.putExtra("Full_Name", all_places.get(namec).getFul
l_Name());
117.                             loc_info.putExtra("imagename", all_places.get(namec).getIma
ge_Name());
118.                             startActivityForResult(loc_info, MY_REQUEST_CODE);
119.                         }
120.                     }
121.                 });
122.             } else {
123.                 int line = getIntent().getIntExtra("line", 0);
124.                 switch (line) {
125.                     case 0:
126.                         KmlLayer layer = new KmlLayer(mMap, R.raw.emubusall, getApplica
tionContext());
127.                         layer.addLayerToMap();
128.                         layer.setOnFeatureClickListener(new KmlLayer.OnFeatureClickList
ener() {@
129.                             Override public void onFeatureClick(Feature feature) {
130.                                 String namec = feature.getProperty("name").toString();
131.                             }
132.                         });
133.                     }
134.                 }
135.             }
136.         }
137.     }
138. }
```

```

133.                 break;
134.             case 1:
135.                 KmlLayer layer2 = new KmlLayer(mMap, R.raw.emubus1, getApplicationContext());
136.                 layer2.addLayerToMap();
137.                 layer2.setOnFeatureClickListener(new KmlLayer.OnFeatureClickListener() {
138.                     Override public void onFeatureClick(Feature feature) {
139.                         String namec = feature.getProperty("name").toString();
140.                     }
141.                 });
142.                 break;
143.             case 2:
144.                 KmlLayer layer3 = new KmlLayer(mMap, R.raw.emubus2, getApplicationContext());
145.                 layer3.addLayerToMap();
146.                 layer3.setOnFeatureClickListener(new KmlLayer.OnFeatureClickListener() {
147.                     Override public void onFeatureClick(Feature feature) {
148.                         String namec = feature.getProperty("name").toString();
149.                     }
150.                 });
151.                 break;
152.             case 3:
153.                 KmlLayer layer4 = new KmlLayer(mMap, R.raw.emubus3, getApplicationContext());
154.                 layer4.addLayerToMap();
155.                 layer4.setOnFeatureClickListener(new KmlLayer.OnFeatureClickListener() {
156.                     Override public void onFeatureClick(Feature feature) {
157.                         String namec = feature.getProperty("name").toString();
158.                     }
159.                 });
160.                 break;
161.             case 4:
162.                 KmlLayer layer5 = new KmlLayer(mMap, R.raw.emubus4, getApplicationContext());
163.                 layer5.addLayerToMap();
164.                 layer5.setOnFeatureClickListener(new KmlLayer.OnFeatureClickListener() {
165.                     Override public void onFeatureClick(Feature feature) {
166.                         String namec = feature.getProperty("name").toString();
167.                     }
168.                 });
169.                 break;
170.             case 5:
171.                 KmlLayer layer6 = new KmlLayer(mMap, R.raw.emubus5, getApplicationContext());
172.                 layer6.addLayerToMap();
173.                 break;
174.             }
175.         }
176.     } catch (XmlPullParserException e) {
177.         e.printStackTrace();
178.     } catch (IOException e) {
179.         e.printStackTrace();
180.     }

```

```

181.         LatLng emu = new LatLng(35.14328, 33.9102722);
182.         mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(emu, 16));
183.         mMap.setMinZoomPreference(14);
184.         LatLng emua = new LatLng(35.0749442, 33.8914845); //SW
185.         LatLng emub = new LatLng(35.2309717, 33.9522488); //NE
186.         LatLngBounds emuc = new LatLngBounds(emua, emub);
187.         mMap.setLatLngBoundsForCameraTarget(emuc);
188.     }@
189.     Override protected void onActivityResult(int requestCode, int resultCode, Intent data) {
190.         super.onActivityResult(requestCode, resultCode, data);
191.         if (requestCode == resultCode) {
192.             MarkerOptions g = new MarkerOptions();
193.             g.position(new LatLng(data.getDoubleExtra("Lati", 33.9085024), data.getDoubleExtra("longt", 35.1460493)));
194.             g.title(data.getStringExtra("Title"));
195.             mMap.addMarker(g);
196.             mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(g.getPosition(), 16));
197.         }
198.     }
199. }

```

Offline_Map_Activity.java

```

1. package com.example.baltu.myapplication.Map;
2. import android.app.Activity;
3. import android.os.Bundle;
4. import android.support.annotation.Nullable;
5. import android.support.v7.app.AppCompatActivity;
6. import com.example.baltu.myapplication.R;
7. import com.github.chrisbanes.photoview.PhotoView;
8. /** * Created by Baltu on 2017-04-05. */
9. public class Offline_Map_Activity extends Activity {@
10.     Override protected void onCreate(@Nullable Bundle savedInstanceState) {
11.         super.onCreate(savedInstanceState);
12.         setContentView(R.layout.offlinemap);
13.         PhotoView gh = (PhotoView) findViewById(R.id.zoomablemap);
14.         gh.setMaximumScale(6);
15.         gh.setMediumScale((float) 3);
16.     }
17. }
18. Places_on_map.java package com.example.baltu.myapplication.Map;
19. /** * Created by Baltu on 2017-04-03. */
20. public class Places_on_map {
21.     private String Place_Name;
22.     private String Full_Name;
23.     private String Image_Name;
24.     public String getPlace_Name() {
25.         return Place_Name;
26.     }
27.     public String getFull_Name() {
28.         return Full_Name;
29.     }
30.     public Places_on_map() {}
31.     public void setPlace_Name(String place_Name) {
32.         Place_Name = place_Name;
33.     }
34.     public void setFull_Name(String full_Name) {
35.         Full_Name = full_Name;
36.     }
37.     public String getImage_Name() {
38.         return Image_Name;

```



```

39.     }
40.     public void setImage_Name(String image_Name) {
41.         Image_Name = image_Name;
42.     }
43.     public Places_on_map(String place_Name, String full_Name, String image_Name) {
44.         Place_Name = place_Name;
45.         Full_Name = full_Name;
46.         Image_Name = image_Name;
47.     }
48. }

```

Locations_on_Map_Details.java

```

1. package com.example.baltu.myapplication.Map;
2. import android.app.Activity;
3. import android.app.Instrumentation;
4. import android.content.Intent;
5. import android.graphics.Bitmap;
6. import android.graphics.BitmapFactory;
7. import android.graphics.drawable.Drawable;
8. import android.os.AsyncTask;
9. import android.os.Bundle;
10. import android.support.annotation.Nullable;
11. import android.util.Log;
12. import android.view.Display;
13. import android.view.View;
14. import android.webkit.WebView;
15. import android.widget.ImageView;
16. import android.widget.TextView;
17. import android.widget.Toast;
18. import com.example.baltu.myapplication.R;
19. import java.io.IOException;
20. import java.io.InputStream;
21. /** * Created by Baltu on 2017-04-09. */
22. public class Locations_on_Map_Details extends Activity {
23.     String url;
24.     private String name;
25.     private Double longt;
26.     private Double Lati;
27.     private ImageView mv;
28.     Override protected void onCreate(@Nullable Bundle savedInstanceState) {
29.         super.onCreate(savedInstanceState);
30.         setContentView(R.layout.map_location);
31.         Intent k = getIntent();
32.         name = k.getStringExtra("Full_Name");
33.         url = k.getStringExtra("url");
34.         if (url != null) {
35.             AsyncTask sk = new DownloadImageTask(((ImageView) findViewById(R.id.Location_Image
36.             )))
37.             {
38.                 execute(new String[] {
39.                     url
40.                 });
41.             } else {
42.                 String ImageS = getIntent().getStringExtra("imagenam");
43.                 if (ImageS != null) try {
44.                     InputStream imagei = getAssets().open("images/" + ImageS);
45.                     Drawable imagef = Drawable.createFromStream(imagei, null);
46.                     ((ImageView) findViewById(R.id.Location_Image)).setImageDrawable(imagef);
47.                 } catch (IOException e) {
48.                     e.printStackTrace();
49.                 }
50.             }
51.             }(((TextView) findViewById(R.id.Place_full_name)).setText(name);

```

```

49.         ((TextView) findViewById(R.id.Place_Info)).setText(getIntent().getStringExtra("Info"))
50.     ;
51.     longt = k.getDoubleExtra("Longt", 33.9085024);
52.     Lati = k.getDoubleExtra("Lat", 35.1460493);
53.     findViewById(R.id.Cloase_Location).setOnClickListener(new View.OnClickListener() {@
54.         Override public void onClick(View v) {
55.             finish();
56.         }
57.     });
58.     findViewById(R.id.Location_Location).setOnClickListener(new View.OnClickListener() {@
59.         Override public void onClick(View v) {
60.             Intent data = new Intent();
61.             data.putExtra("longt", longt);
62.             data.putExtra("Lati", Lati);
63.             data.putExtra("Title", name);
64.             setResult(5584, data);
65.             finish();
66.         }
67.     });
68.     private class DownloadImageTask extends AsyncTask < String, Void, Bitmap > {
69.         ImageView mMyImageView;
70.         public DownloadImageTask(ImageView MyImageView) {
71.             this.mMyImageView = MyImageView;
72.         }
73.         protected Bitmap doInBackground(String...urls) {
74.             String urldisplay = urls[0];
75.             Bitmap mIcon11 = null;
76.             try {
77.                 InputStream in = new java.net.URL(urldisplay).openStream();
78.                 mIcon11 = BitmapFactory.decodeStream( in );
79.             } catch (Exception e) {
80.                 Log.e("Error", e.getMessage());
81.                 e.printStackTrace();
82.             }
83.             return mIcon11;
84.         }
85.         protected void onPostExecute(Bitmap result) {
86.             mMyImageView.setImageBitmap(result);
87.         }
88.     }
89. }

```

Buses_Schedules.java

```

1. package com.example.baltu.myapplication;
2. import android.support.v7.app.AppCompatActivity;
3. import android.os.Bundle;
4. import android.support.v7.widget.RecyclerView;
5. import android.view.GestureDetector;
6. import android.view.MotionEvent;
7. import android.view.View;
8. import android.widget.AdapterView;
9. import android.widget.AdapterView;
10. import android.widget.ArrayAdapter;
11. import android.widget.Spinner;
12. import android.widget.TabHost;
13. import android.widget.ViewAnimator;
14. import com.example.baltu.myapplication.Data_Provider.Adapters.BusItemsAdapter;
15. import com.example.baltu.myapplication.DataTypes.Bus_Data;
16. import com.example.baltu.myapplication.Data_Provider.Adapters.BusItemsAdapterweekend;
17. import com.example.baltu.myapplication.Data_Provider.Local_DB_Main_Provider;

```



```

70.         }
71.     }
72. });
73. lefttb.setOnClickListener(new View.OnClickListener() {@
74.     Override public void onClick(View v) {
75.         ani.showPrevious();
76.         anit.showPrevious();
77.     }
78. });
79. righttb.setOnClickListener(new View.OnClickListener() {@
80.     Override public void onClick(View v) {
81.         ani.showNext();
82.         anit.showNext();
83.     }
84. });
85. }
86. private void Buseslistsetup(List < Bus_Data > x, List < Bus_Data > y) {
87.     View v = findViewById(R.id.buscyc1);
88.     View v2 = findViewById(R.id.buscyc2); //assert (v!=null);
89.     RecyclerView crn = (RecyclerView) v;
90.     RecyclerView crn2 = (RecyclerView) v2;
91.     BusItemsAdapter cadpter = new BusItemsAdapter(x);
92.     BusItemsAdapterweekend cadpter2 = new BusItemsAdapterweekend(y);
93.     crn.setAdapter(cadpter);
94.     crn2.setAdapter(cadpter2);
95. }
96. }

```

News_Activity.java

```

1. package com.example.baltu.myapplication.News;
2. import android.os.AsyncTask;
3. import android.support.v7.app.AppCompatActivity;
4. import android.os.Bundle;
5. import android.support.v7.widget.RecyclerView;
6. import android.widget.TabHost;
7. import com.example.baltu.myapplication.DataTypes.News_Class;
8. import com.example.baltu.myapplication.R;
9. import org.xmlpull.v1.XmlPullParserException;
10. import java.io.IOException;
11. import java.io.InputStream;
12. import java.net.HttpURLConnection;
13. import java.net.URL;
14. import java.util.List;
15. public class News_Activity extends AppCompatActivity {
16.     String urlString = "http://ww1.emu.edu.tr/rss/en/21";
17.     String EventsurlString = "http://ww1.emu.edu.tr/rss/en/33";
18.     String AnnurlString = "http://ww1.emu.edu.tr/rss/en/32";
19.     List < News_Class > news;
20.     List < News_Class > Events;
21.     List < News_Class > Announcements;
22.     InputStream stream;
23.     InputStream stream2;
24.     InputStream stream3;@
25.     Override protected void onCreate(Bundle savedInstanceState) {
26.         super.onCreate(savedInstanceState);
27.         setContentView(R.layout.activity_news);
28.         TabHost NTH = (TabHost) findViewById(R.id.news_tab);
29.         NTH.setup();
30.         final TabHost.TabSpec spec1 = NTH.newTabSpec("News").setContent(R.id.tab1).setIndicator("News");
31.         NTH.addTab(spec1);

```

```

32.         TabHost.TabSpec spec2 = NTH.newTabSpec("Events").setContent(R.id.tab2).setIndicator("E
vents");
33.         NTH.addTab(spec2);
34.         TabHost.TabSpec spec3 = NTH.newTabSpec("Announ").setContent(R.id.tab3).setIndicator("A
nnoun.");
35.         NTH.addTab(spec3);
36.         RSSFEED d = new RSSFEED();
37.         d.execute(new String[] {
38.             urlString, EventsurlString, AnnurlString
39.         });
40.         if (news != null) {
41.             RecyclerView nr = (RecyclerView) findViewById(R.id.news_recycler);
42.             NewsItemsAdapter NRA = new NewsItemsAdapter(this, news);
43.             nr.setAdapter(NRA);
44.         }
45.     }
46.     private class RSSFEED extends AsyncTask < String, Void, String > {@
47.         Override protected String doInBackground(String...urls) {
48.             try {
49.                 stream = downloadUrl(urls[0]);
50.                 news = NewsRssParser.parse(stream);
51.                 stream2 = downloadUrl(urls[1]);
52.                 Events = NewsRssParser.parse(stream2);
53.                 stream3 = downloadUrl(urls[2]);
54.                 Announcements = NewsRssParser.parse(stream3);
55.             } catch (XmlPullParserException e) {
56.                 e.printStackTrace();
57.             } catch (IOException e) {
58.                 e.printStackTrace();
59.             }
60.             if (stream != null) {
61.                 try {
62.                     stream.close();
63.                 } catch (IOException e) {
64.                     e.printStackTrace();
65.                 }
66.             }
67.             if (stream2 != null) {
68.                 try {
69.                     stream2.close();
70.                 } catch (IOException e) {
71.                     e.printStackTrace();
72.                 }
73.             }
74.             if (stream3 != null) {
75.                 try {
76.                     stream3.close();
77.                 } catch (IOException e) {
78.                     e.printStackTrace();
79.                 }
80.             }
81.             return "y";
82.         }@
83.         Override protected void onPostExecute(String s) {
84.             super.onPostExecute(s);
85.             if (news != null) {
86.                 RecyclerView nr = (RecyclerView) findViewById(R.id.news_recycler);
87.                 NewsItemsAdapter NRA = new NewsItemsAdapter(News_Activity.this, news);
88.                 nr.setAdapter(NRA);
89.                 RecyclerView nr2 = (RecyclerView) findViewById(R.id.Events_recycler);
90.                 NewsItemsAdapter NRA2 = new NewsItemsAdapter(News_Activity.this, Events);

```

```

91.         nr2.setAdapter(NRA2);
92.         RecyclerView nr3 = (RecyclerView) findViewById(R.id.Anno_recycler);
93.         NewsItemsAdapter NRA3 = new NewsItemsAdapter(News_Activity.this, Announcements)
;
94.         nr3.setAdapter(NRA3);
95.     }
96. }
97. }
98. private InputStream downloadUrl(String urlString) throws IOException {
99.     URL url = new URL(urlString);
100.     HttpURLConnection conn = (HttpURLConnection) url.openConnection();
101.     conn.setReadTimeout(10000 /* milliseconds */ );
102.     conn.setConnectTimeout(15000 /* milliseconds */ );
103.     conn.setRequestMethod("GET");
104.     conn.setDoInput(true); // Starts the query
105.     conn.connect();
106.     return conn.getInputStream();
107. }
108. }

```

NewsRssParser.java

```

1. package com.example.baltu.myapplication.News;
2. import android.util.Xml;
3. import com.example.baltu.myapplication.DataTypes.News_Class;
4. import org.xmlpull.v1.XmlPullParser;
5. import org.xmlpull.v1.XmlPullParserException;
6. import java.io.IOException;
7. import java.io.InputStream;
8. import java.text.DateFormat;
9. import java.text.ParseException;
10. import java.text.SimpleDateFormat;
11. import java.util.ArrayList;
12. import java.util.Date;
13. import java.util.List;
14. /** * Created by Baltu on 2017-05-19. */
15. public class NewsRssParser {
16.     private static final String ns = null;
17.     private static final String ns2 = null;
18.     public static List parse(InputStream in ) throws XmlPullParserException, IOException {
19.         try {
20.             XmlPullParser parser = Xml.newPullParser();
21.             parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false);
22.             parser.setInput(in , null);
23.             parser.nextTag();
24.             return readRSS(parser);
25.         } finally { in .close();
26.         }
27.     }
28.     private static List readRSS(XmlPullParser parser) throws XmlPullParserException, IOExcepti
on {
29.         List entries = new ArrayList();
30.         parser.require(XmlPullParser.START_TAG, ns, "rss");
31.         while (parser.next() != XmlPullParser.END_TAG) {
32.             if (parser.getEventType() != XmlPullParser.START_TAG) {
33.                 continue;
34.             }
35.             String name = parser.getName();
36.             if (name.equals("item")) {
37.                 entries.add(readItem(parser));
38.             } else if (name.equals("channel")) {
39.                 continue;

```

```

40.         } else {
41.             skip(parser);
42.         }
43.     }
44.     return entries;
45. }
46. private static News_Class readItem(XmlPullParser parser) throws XmlPullParserException, IO
Exception {
47.     parser.require(XmlPullParser.START_TAG, ns, "item");
48.     String Title = null;
49.     String Link = null;
50.     String Description = null;
51.     Date Date = null;
52.     while (parser.next() != XmlPullParser.END_TAG) {
53.         if (parser.getEventType() != XmlPullParser.START_TAG) {
54.             continue;
55.         }
56.         String name = parser.getName();
57.         if (name.equals("title")) {
58.             Title = readTitle(parser);
59.         } else if (name.equals("link")) {
60.             Link = readLink(parser);
61.         } else if (name.equals("description")) {
62.             Description = readDes(parser);
63.         } else if (name.equals("pubDate")) {
64.             Date = readDate(parser);
65.         } else {
66.             skip(parser);
67.         }
68.     }
69.     return new News_Class(Title, Link, Description, Date);
70. }
71. private static String readTitle(XmlPullParser parser) throws IOException, XmlPullParserExc
ption {
72.     parser.require(XmlPullParser.START_TAG, ns, "title");
73.     String title = readText(parser);
74.     parser.require(XmlPullParser.END_TAG, ns, "title");
75.     return title;
76. }
77. private static String readLink(XmlPullParser parser) throws IOException, XmlPullParserExce
ption {
78.     parser.require(XmlPullParser.START_TAG, ns, "link");
79.     String title = readText(parser);
80.     parser.require(XmlPullParser.END_TAG, ns, "link");
81.     return title;
82. }
83. private static String readDes(XmlPullParser parser) throws IOException, XmlPullParserExcep
tion {
84.     parser.require(XmlPullParser.START_TAG, ns, "description");
85.     String title = readText(parser);
86.     parser.require(XmlPullParser.END_TAG, ns, "description");
87.     return title;
88. }
89. private static Date readDate(XmlPullParser parser) throws IOException, XmlPullParserExcept
ion {
90.     parser.require(XmlPullParser.START_TAG, ns, "pubDate");
91.     String title = readText(parser);
92.     parser.require(XmlPullParser.END_TAG, ns, "pubDate");
93.     DateFormat df = new SimpleDateFormat("E, dd MMM yyyy hh:mm");
94.     Date dt = new Date();
95.     try {

```



```

96.         dt = df.parse(title);
97.     } catch (ParseException e) {
98.         e.printStackTrace();
99.     }
100.     return dt;
101. }
102.     private static String readText(XmlPullParser parser) throws IOException, XmlPullParserException {
103.         String result = "";
104.         if (parser.next() == XmlPullParser.TEXT) {
105.             result = parser.getText();
106.             parser.nextTag();
107.         }
108.         return result;
109.     }
110.     private static void skip(XmlPullParser parser) throws XmlPullParserException, IOException {
111.         if (parser.getEventType() != XmlPullParser.START_TAG) {
112.             throw new IllegalStateException();
113.         }
114.         int depth = 1;
115.         while (depth != 0) {
116.             switch (parser.next()) {
117.                 case XmlPullParser.END_TAG:
118.                     depth--;
119.                     break;
120.                 case XmlPullParser.START_TAG:
121.                     depth++;
122.                     break;
123.             }
124.         }
125.     }
126. }
127. NewsItemsAdapter.java package com.example.baltu.myapplication.News;
128. import android.content.Context;
129. import android.content.Intent;
130. import android.support.v7.widget.RecyclerView;
131. import android.view.LayoutInflater;
132. import android.view.View;
133. import android.view.ViewGroup;
134. import android.widget.TextView;
135. import com.example.baltu.myapplication.DataTypes.News_Class;
136. import com.example.baltu.myapplication.MyWebViwer;
137. import com.example.baltu.myapplication.R;
138. import java.text.DateFormat;
139. import java.text.SimpleDateFormat;
140. import java.util.Date;
141. import java.util.List;
142. public class NewsItemsAdapter extends RecyclerView.Adapter < NewsItemsAdapter.ViewHolder > {
143.     private List < News_Class > mItems;
144.     private Context mContext;
145.     int f;
146.     public NewsItemsAdapter(Context context, List < News_Class > items) {
147.         this.mContext = context;
148.         this.mItems = items;
149.     }
150.     Override public NewsItemsAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
151.         LayoutInflater inflater = LayoutInflater.from(mContext);
152.         View itemView = inflater.inflate(R.layout.news_layout, parent, false);

```

```

153.         ViewHolder viewHolder = new ViewHolder(itemView);
154.         return viewHolder;
155.     }@
156.     Override public void onBindViewHolder(final NewsItemsAdapter.ViewHolder holder, int
position) {
157.         final News_Class item = mItems.get(position);
158.         holder.Title.setText(item.getTitle());
159.         holder.TDescription.setText(item.getDescription());
160.         Date d = item.getDate();
161.         DateFormat dtd = new SimpleDateFormat("EEE dd MMM yyyy hh:mm");
162.         holder.DATE.setText(dtd.format(d));
163.         holder.mView.setOnClickListener(new View.OnClickListener() {@
164.             Override public void onClick(View v) {
165.                 Intent newsintent = new Intent(holder.mView.getContext(), MyWebViwer.cl
ass);
166.                 newsintent.putExtra("URL", item.getLink());
167.                 holder.mView.getContext().startActivity(newsintent);
168.             }
169.         });
170.     }@
171.     Override public int getItemCount() {
172.         return mItems.size();
173.     }
174.     public static class ViewHolder extends RecyclerView.ViewHolder {
175.         public TextView Title;
176.         public TextView TDescription;
177.         public TextView DATE;
178.         public View mView;
179.         public ViewHolder(View itemView) {
180.             super(itemView);
181.             Title = (TextView) itemView.findViewById(R.id.News_Title);
182.             TDescription = (TextView) itemView.findViewById(R.id.News_Desc);
183.             DATE = (TextView) itemView.findViewById(R.id.News_Date);
184.             mView = itemView;
185.         }
186.     }
187. }

```

InfoListActivity.java

```

1. package com.example.baltu.myapplication.Information_Activity;
2. import android.content.Context;
3. import android.content.Intent;
4. import android.os.Bundle;
5. import android.support.annotation.NonNull;
6. import android.support.v7.app.AppCompatActivity;
7. import android.support.v7.widget.RecyclerView;
8. import android.support.v7.widget.Toolbar;
9. import android.view.LayoutInflater;
10. import android.view.View;
11. import android.view.ViewGroup;
12. import android.widget.TextView;
13. import com.example.baltu.myapplication.DataTypes.InfoContent;
14. import com.example.baltu.myapplication.Information_Activity.InfoDetailActivity;
15. import com.example.baltu.myapplication.Information_Activity.InfoDetailFragment;
16. import com.example.baltu.myapplication.R;
17. import java.util.List;
18. /** * An activity representing a list of Items. This activity * has different presentations fo
r handset and tablet-
size devices. On * handsets, the activity presents a list of items, which when touched, * lead
to a {@link InfoDetailActivity} representing * item details. On tablets, the activity present
s the list of items and * item details side-by-side using two vertical panes. */

```

```

19. public class InfoListActivity extends AppCompatActivity {
20.     /** * Whether or not the activity is in two-
pane mode, i.e. running on a tablet * device. */
21.     private boolean mTwoPane;
22.     @Override protected void onCreate(Bundle savedInstanceState) {
23.         super.onCreate(savedInstanceState);
24.         setContentView(R.layout.activity_info_list);
25.         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
26.         setSupportActionBar(toolbar);
27.         toolbar.setTitle(getTitle());
28.         View recyclerView = findViewById(R.id.info_list);
29.         assert recyclerView != null;
30.         setupRecyclerView((RecyclerView) recyclerView);
31.         if (findViewById(R.id.info_detail_container) != null) { // The detail container view w
ill be present only in the // large-screen layouts (res/values-
w900dp). // If this view is present, then the // activity should be in two-pane mode.
32.             mTwoPane = true;
33.         }
34.     }
35.     private void setupRecyclerView(@NonNull RecyclerView recyclerView) {
36.         recyclerView.setAdapter(new SimpleItemRecyclerViewAdapter(InfoContent.ITEMS));
37.     }
38.     public class SimpleItemRecyclerViewAdapter extends RecyclerView.Adapter < SimpleItemRecycl
erViewHolder > {
39.         private final List < InfoContent.InfoItem > mValues;
40.         public SimpleItemRecyclerViewAdapter(List < InfoContent.InfoItem > items) {
41.             mValues = items;
42.         }
43.         @Override public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
44.             View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.info_list_co
ntent, parent, false);
45.             return new ViewHolder(view);
46.         }
47.         @Override public void onBindViewHolder(final ViewHolder holder, int position) {
48.             holder.mItem = mValues.get(position); //holder.mIdView.setText(mValues.get(positio
n).id);
49.             holder.mContentView.setText(mValues.get(position).content);
50.             holder.mView.setOnClickListener(new View.OnClickListener() {@
51.                 @Override public void onClick(View v) {
52.                     if (mTwoPane) {
53.                         Bundle arguments = new Bundle();
54.                         arguments.putString(InfoDetailFragment.ARG_ITEM_ID, holder.mItem.id);
55.                         InfoDetailFragment fragment = new InfoDetailFragment();
56.                         fragment.setArguments(arguments);
57.                         getSupportFragmentManager().beginTransaction().replace(R.id.info_detai
l_container, fragment).commit();
58.                     } else {
59.                         Context context = v.getContext();
60.                         Intent intent = new Intent(context, InfoDetailActivity.class);
61.                         intent.putExtra(InfoDetailFragment.ARG_ITEM_ID, holder.mItem.id);
62.                         context.startActivity(intent);
63.                     }
64.                 }
65.             });
66.         }
67.         @Override public int getItemCount() {
68.             return mValues.size();
69.         }
70.         public class ViewHolder extends RecyclerView.ViewHolder {
71.             public final View mView; //public final TextView mIdView;

```

```

72.         public final TextView mContentView;
73.         public InfoContent.InfoItem mItem;
74.         public ViewHolder(View view) {
75.             super(view);
76.             mView = view; //mIdView = (TextView) view.findViewById(R.id.id);
77.             mContentView = (TextView) view.findViewById(R.id.content);
78.         }
79.         Override public String toString() {
80.             return super.toString() + " '" + mContentView.getText() + "'";
81.         }
82.     }
83. }
84. }

```

InfoDetailActivity.java

```

1. package com.example.baltu.myapplication.Information_Activity;
2. import android.content.Intent;
3. import android.os.Bundle;
4. import android.support.design.widget.FloatingActionButton;
5. import android.support.design.widget.Snackbar;
6. import android.support.v7.widget.Toolbar;
7. import android.view.View;
8. import android.support.v7.app.AppCompatActivity;
9. import android.support.v7.app.ActionBar;
10. import android.view.MenuItem;
11. import com.example.baltu.myapplication.R;
12. /** * An activity representing a single Info detail screen. This * activity is only used narrow width devices. On tablet-size devices, * item details are presented side-by-side with a list of items * in a {@link InfoListActivity}. */
13. public class InfoDetailActivity extends AppCompatActivity {
14.     Override protected void onCreate(Bundle savedInstanceState) {
15.         super.onCreate(savedInstanceState);
16.         setContentView(R.layout.activity_info_detail);
17.         Toolbar toolbar = (Toolbar) findViewById(R.id.detail_toolbar);
18.         setSupportActionBar(toolbar);
19.         FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
20.         fab.setOnClickListener(new View.OnClickListener() {
21.             Override public void onClick(View view) {
22.                 Snackbar.make(view, "Replace with your own detail action", Snackbar.LENGTH_LONG).setAction("Action", null).show();
23.             }
24.         }); // Show the Up button in the action bar.
25.         ActionBar actionBar = getSupportActionBar();
26.         if (actionBar != null) {
27.             actionBar.setDisplayHomeAsUpEnabled(true);
28.         } // savedInstanceState is non-null when there is fragment state // saved from previous configurations of this activity // (e.g. when rotating the screen from portrait to landscape). // In this case, the fragment will be automatically be re-added // to its container so we don't need to manually add it. // For more information, see the Fragments API guide at: // http://developer.android.com/guide/components/fragments.html //
29.         if (savedInstanceState == null) { // Create the detail fragment and add it to the activity // using a fragment transaction.
30.             Bundle arguments = new Bundle();
31.             arguments.putString(InfoDetailFragment.ARG_ITEM_ID, getIntent().getStringExtra(InfoDetailFragment.ARG_ITEM_ID));
32.             InfoDetailFragment fragment = new InfoDetailFragment();
33.             fragment.setArguments(arguments);
34.             getSupportFragmentManager().beginTransaction().add(R.id.info_detail_container, fragment).commit();

```

```

35.     }
36. }@
37. Override public boolean onOptionsItemSelected(MenuItem item) {
38.     int id = item.getItemId();
39.     if (id == android.R.id.home) { // This ID represents the Home or Up button. In the cas
e of this // activity, the Up button is shown. For // more details, see the Navigation pattern
on Android Design: // // http://developer.android.com/design/patterns/navigation.html#up-vs-
back //
40.         navigateUpTo(new Intent(this, InfoListActivity.class));
41.         return true;
42.     }
43.     return super.onOptionsItemSelected(item);
44. }
45. }

```

InfoDetailFragment.java

```

1. package com.example.baltu.myapplication.Information_Activity;
2. import android.app.Activity;
3. import android.support.design.widget.CollapsingToolbarLayout;
4. import android.os.Bundle;
5. import android.support.v4.app.Fragment;
6. import android.view.LayoutInflater;
7. import android.view.View;
8. import android.view.ViewGroup;
9. import android.widget.TextView;
10. import com.example.baltu.myapplication.DataTypes.InfoContent;
11. import com.example.baltu.myapplication.R;
12. /** * A fragment representing a single Info detail screen. * This fragment is either contained
in a {@link InfoListActivity} * in two-
pane mode (on tablets) or a {@link InfoDetailActivity} * on handsets. */
13. public class InfoDetailFragment extends Fragment {
14.     /** * The fragment argument representing the item ID that this fragment * represen
ts. */
15.     public static final String ARG_ITEM_ID = "item_id";
16.     /** * The dummy content this fragment is presenting. */
17.     private InfoContent.InfoItem mItem;
18.     /** * Mandatory empty constructor for the fragment manager to instantiate the * fr
agment (e.g. upon screen orientation changes). */
19.     public InfoDetailFragment() {}@
20.     Override public void onCreate(Bundle savedInstanceState) {
21.         super.onCreate(savedInstanceState);
22.         if (getArguments().containsKey(ARG_ITEM_ID)) { // Load the dummy content specified by
the fragment // arguments. In a real-
world scenario, use a Loader // to load content from a content provider.
23.             mItem = InfoContent.ITEM_MAP.get(getArguments().getString(ARG_ITEM_ID));
24.             Activity activity = this.getActivity();
25.             CollapsingToolbarLayout appBarLayout = (CollapsingToolbarLayout) activity.findViewById
(R.id.toolbar_layout);
26.             if (appBarLayout != null) {
27.                 appBarLayout.setTitle(mItem.content);
28.             }
29.         }
30.     }@
31.     Override public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle sav
edInstanceState) {
32.         View rootView = inflater.inflate(R.layout.info_detail, container, false); // Show the
dummy content as text in a TextView.
33.         if (mItem != null) {
34.             ((TextView) rootView.findViewById(R.id.info_detail)).setText(mItem.details);
35.         }
36.         return rootView;

```

```
37.     }
38. }
```

GPA_Calculator.java

```
1. package com.example.baltu.myapplication;
2. import android.support.v7.app.AppCompatActivity;
3. import android.os.Bundle;
4. import android.view.View;
5. import android.widget.ArrayAdapter;
6. import android.widget.Spinner;
7. import android.widget.TextView;
8. import android.widget.Toast;
9. import java.text.DecimalFormat;
10. import java.util.ArrayList;
11. import java.util.List;
12. public class GPA_Calculator extends AppCompatActivity {
13.     private Spinner s11;
14.     TextView result;
15.     private Spinner s12;
16.     private Spinner s13;
17.     private Spinner s14;
18.     private Spinner s15;
19.     private Spinner s16;
20.     private Spinner s17;
21.     private Spinner s21;
22.     private Spinner s22;
23.     private Spinner s23;
24.     private Spinner s24;
25.     private Spinner s25;
26.     private Spinner s26;
27.     private Spinner s27;
28.     List < grades > allgrades = new ArrayList < > ();
29.     TextView semsterc;
30.     TextView semstergpa;
31.     TextView totalc;
32.     TextView cgpatext;
33.     TextView newCGPA;@
34.     Override protected void onCreate(Bundle savedInstanceState) {
35.         super.onCreate(savedInstanceState);
36.         setContentView(R.layout.activity_gpa_calculator);
37.         s11 = (Spinner) findViewById(R.id.spinner11);
38.         s12 = (Spinner) findViewById(R.id.spinner12);
39.         s13 = (Spinner) findViewById(R.id.spinner13);
40.         s14 = (Spinner) findViewById(R.id.spinner14);
41.         s15 = (Spinner) findViewById(R.id.spinner15);
42.         s16 = (Spinner) findViewById(R.id.spinner16);
43.         s17 = (Spinner) findViewById(R.id.spinner17);
44.         s21 = (Spinner) findViewById(R.id.spinner21);
45.         s22 = (Spinner) findViewById(R.id.spinner22);
46.         s23 = (Spinner) findViewById(R.id.spinner23);
47.         s24 = (Spinner) findViewById(R.id.spinner24);
48.         s25 = (Spinner) findViewById(R.id.spinner25);
49.         s26 = (Spinner) findViewById(R.id.spinner26);
50.         s27 = (Spinner) findViewById(R.id.spinner27);
51.         result = (TextView) findViewById(R.id.GPAResult);
52.         semsterc = (TextView) findViewById(R.id.semesterCredits);
53.         semstergpa = (TextView) findViewById(R.id.yourGPA);
54.         totalc = (TextView) findViewById(R.id.TotalOldCredits);
55.         cgpatext = (TextView) findViewById(R.id.yourCGPA);
56.         newCGPA = (TextView) findViewById(R.id.new_CGPA);
57.         setallspinners();
```

```

58.         setall2spinners();
59.         View caculate = findViewById(R.id.calculat_bt);
60.         caculate.setOnClickListener(new View.OnClickListener() {@
61.             Override public void onClick(View v) {
62.                 allgrades.clear();
63.                 int allmycredits = 0;
64.                 double allmygrades = 0;
65.                 double gpa = 0;
66.                 if (s11.getSelectedItemPosition() != 0 && s21.getSelectedItemPosition() != 0)
addgrade(s11.getSelectedItemPosition(), s21.getSelectedItemPosition());
67.                 if (s12.getSelectedItemPosition() != 0 && s22.getSelectedItemPosition() != 0)
addgrade(s12.getSelectedItemPosition(), s22.getSelectedItemPosition());
68.                 if (s13.getSelectedItemPosition() != 0 && s23.getSelectedItemPosition() != 0)
addgrade(s13.getSelectedItemPosition(), s23.getSelectedItemPosition());
69.                 if (s14.getSelectedItemPosition() != 0 && s24.getSelectedItemPosition() != 0)
addgrade(s14.getSelectedItemPosition(), s24.getSelectedItemPosition());
70.                 if (s15.getSelectedItemPosition() != 0 && s25.getSelectedItemPosition() != 0)
addgrade(s15.getSelectedItemPosition(), s25.getSelectedItemPosition());
71.                 if (s16.getSelectedItemPosition() != 0 && s26.getSelectedItemPosition() != 0)
addgrade(s16.getSelectedItemPosition(), s26.getSelectedItemPosition());
72.                 if (s17.getSelectedItemPosition() != 0 && s27.getSelectedItemPosition() != 0)
addgrade(s17.getSelectedItemPosition(), s27.getSelectedItemPosition());
73.                 if (allgrades.size() != 0) {
74.                     for (grades item: allgrades) {
75.                         allmycredits += item.getCredit();
76.                         allmygrades += ((item.getGrade() / 4.0) * (double) item.getCredit());
77.                     }
78.                     gpa = (allmygrades / (double) allmycredits) * 4.0;
79.                     DecimalFormat df = new DecimalFormat("0.00");
80.                     result.setText(df.format(gpa));
81.                     semstergpa.setText(df.format(gpa));
82.                     semsterc.setText(Integer.toString(allmycredits));
83.                 } else Toast.makeText(GPA_Calculator.this, "Error", Toast.LENGTH_SHORT).show()
;
84.             }
85.         });
86.         View caculatecgpa = findViewById(R.id.CalculateCGPA);
87.         caculatecgpa.setOnClickListener(new View.OnClickListener() {@
88.             Override public void onClick(View v) {
89.                 if (!semstergpa.getText().toString().equals("") && !semsterc.getText().toStrin
g().equals("") && !totalc.getText().toString().equals("") && !cgpatext.getText().toString().eq
uals("")) {
90.                     int sc = Integer.parseInt(semsterc.getText().toString());
91.                     double sgpa = Double.parseDouble(semstergpa.getText().toString());
92.                     int tc = Integer.parseInt(totalc.getText().toString());
93.                     double cgpa = Double.parseDouble(cgpatext.getText().toString());
94.                     if (sgpa > 4 || cgpa > 4) {
95.                         Toast.makeText(GPA_Calculator.this, "Gpa and CGPA should be less than
4", Toast.LENGTH_LONG).show();
96.                         return;
97.                     }
98.                     if (sc == 0) {
99.                         Toast.makeText(GPA_Calculator.this, "semester credits can't be 0", Toa
st.LENGTH_LONG).show();
100.                        return;
101.                    }
102.                    if (sc > 30) {
103.                        Toast.makeText(GPA_Calculator.this, "semester credits are too l
arge", Toast.LENGTH_LONG).show();
104.                        return;

```



```

105.                }
106.                double ncgpa = (((sgpa / 4.0) * (double) sc) + ((cgpa / 4.0) * (double) tc)) / (sc + tc)) * 4;
107.                DecimalFormat df = new DecimalFormat("0.00");
108.                newCGPA.setText(df.format(ncgpa));
109.            } else {
110.                Toast.makeText(GPA_Calculator.this, "Enter All Required Details", Toast.LENGTH_SHORT).show();
111.            }
112.        }
113.    });
114.    }
115.    private void addgrade(int x, int y) {
116.        double mygrade = 0;
117.        switch (y) {
118.            case 1:
119.                mygrade = 4;
120.                break;
121.            case 2:
122.                mygrade = 3.7;
123.                break;
124.            case 3:
125.                mygrade = 3.3;
126.                break;
127.            case 4:
128.                mygrade = 3;
129.                break;
130.            case 5:
131.                mygrade = 2.7;
132.                break;
133.            case 6:
134.                mygrade = 2.3;
135.                break;
136.            case 7:
137.                mygrade = 2;
138.                break;
139.            case 8:
140.                mygrade = 1.7;
141.                break;
142.            case 9:
143.                mygrade = 1.3;
144.                break;
145.            case 10:
146.                mygrade = 1;
147.                break;
148.            case 11:
149.                mygrade = 0.7;
150.                break;
151.            case 12:
152.                mygrade = 0;
153.                break;
154.        }
155.        allgrades.add(new grades(x, mygrade));
156.    }
157.    private void setallspinners() {
158.        ArrayAdapter < CharSequence > items = ArrayAdapter.createFromResource(this, R.array.Credits, android.R.layout.simple_spinner_item);
159.        items.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
160.        s11.setAdapter(items);
161.        s12.setAdapter(items);
162.        s13.setAdapter(items);

```

```

163.         s14.setAdapter(items);
164.         s15.setAdapter(items);
165.         s16.setAdapter(items);
166.         s17.setAdapter(items);
167.     }
168.     private void setall12spinners() {
169.         ArrayAdapter < CharSequence > items = ArrayAdapter.createFromResource(this, R.array.Grades, android.R.layout.simple_spinner_item);
170.         items.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
171.         s21.setAdapter(items);
172.         s22.setAdapter(items);
173.         s23.setAdapter(items);
174.         s24.setAdapter(items);
175.         s25.setAdapter(items);
176.         s26.setAdapter(items);
177.         s27.setAdapter(items);
178.     }
179.     private class grades {
180.         private int credit;
181.         private double grade;
182.         public grades(int credit, double grade) {
183.             this.credit = credit;
184.             this.grade = grade;
185.         }
186.         public int getCredit() {
187.             return credit;
188.         }
189.         public void setCredit(int credit) {
190.             this.credit = credit;
191.         }
192.         public double getGrade() {
193.             return grade;
194.         }
195.         public void setGrade(double grade) {
196.             this.grade = grade;
197.         }
198.     }
199. }

```

Settings_activity.java

```

1. package com.example.baltu.myapplication.Settings;
2. import android.app.Activity;
3. import android.content.Context;
4. import android.content.Intent;
5. import android.net.ConnectivityManager;
6. import android.net.NetworkInfo;
7. import android.os.Bundle;
8. import android.view.View;
9. import android.widget.Toast;
10. import com.example.baltu.myapplication.R;
11. import com.example.baltu.myapplication.Updater.Main_Updater2;
12. public class Settings_activity extends Activity {@
13.     Override protected void onCreate(Bundle savedInstanceState) {
14.         super.onCreate(savedInstanceState);
15.         setContentView(R.layout.settings);
16.         findViewById(R.id.button4).setOnClickListener(new View.OnClickListener() {@
17.             Override public void onClick(View v) {
18.                 if (isNetworkAvailable()) {
19.                     new Main_Updater2(Settings_activity.this);
20.                 } else Toast.makeText(Settings_activity.this, "Connect To the Internet", Toast.LENGTH_SHORT).show();

```

```

21.     }
22.     });
23. }
24. public void pass(View view) {
25.     Intent calenderintents2 = new Intent(Settings_activity.this, Password_Settings.class);
26.     startActivity(calenderintents2);
27. }
28. private boolean isNetworkAvailable() {
29.     ConnectivityManager connectivityManager = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
30.     NetworkInfo activeNetworkInfo = connectivityManager.getActiveNetworkInfo();
31.     return activeNetworkInfo != null && activeNetworkInfo.isConnected();
32. }
33. }

```

Password_Settings.java

```

1. package com.example.baltu.myapplication.Settings;
2. import android.app.Activity;
3. import android.app.AlertDialog;
4. import android.content.DialogInterface;
5. import android.content.SharedPreferences;
6. import android.os.Bundle;
7. import android.support.v7.widget.SwitchCompat;
8. import android.text.InputType;
9. import android.util.Log;
10. import android.widget.CompoundButton;
11. import android.widget.EditText;
12. import com.example.baltu.myapplication.R;
13. import com.scottyab.aescrypt.AESCrypt;
14. public class Password_Settings extends Activity {
15.     private String m_Text = "";
16.     Override protected void onCreate(Bundle savedInstanceState) {
17.         super.onCreate(savedInstanceState);
18.         setContentView(R.layout.activity_password__settings);
19.         SharedPreferences prefs = getSharedPreferences(getString(R.string.preference_file_key), MODE_PRIVATE);
20.         boolean key = prefs.getBoolean("pass_on", false);
21.         final SwitchCompat witch = (SwitchCompat) findViewById(R.id.switchpass);
22.         witch.setChecked(key);
23.         witch.setOnCheckedChangeListener(new SwitchCompat.OnCheckedChangeListener() {
24.             Override public void onCheckedChanged(CompoundButton switchCompat, boolean isChecked) {
25.                 SharedPreferences prefs = getSharedPreferences(getString(R.string.preference_file_key), MODE_PRIVATE);
26.                 boolean key = prefs.getBoolean("pass_on", false);
27.                 final SharedPreferences.Editor prefsE = prefs.edit();
28.                 if (!isChecked) {
29.                     prefsE.putBoolean("pass_on", false);
30.                     prefsE.commit();
31.                     prefsE.putString("Password_String", "");
32.                     Log.d("Pass_On", "null");
33.                 } else {
34.                     AlertDialog.Builder builder = new AlertDialog.Builder(Password_Settings.this);
35.                     builder.setTitle("Enter New Password"); // Set up the input
36.                     final EditText input = new EditText(Password_Settings.this); // Specify the type of input expected; this, for example, sets the input as a password, and will mask the text
37.                     input.setInputType(InputType.TYPE_CLASS_NUMBER | InputType.TYPE_NUMBER_VARIATION_PASSWORD);

```

```

38.         builder.setView(input); // Set up the buttons
39.         builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {@
40.             Override public void onClick(DialogInterface dialog, int which) {
41.                 m_Text = input.getText().toString();
42.                 if (m_Text.isEmpty()) {
43.                     witch.setChecked(false);
44.                     return;
45.                 } else {
46.                     try {
47.                         prefsE.putBoolean("pass_on", true);
48.                         prefsE.putString("Password_String", AESCrypt.encrypt("Can't
heWorldUnderstand?", "TuNiSbYnIgHt" + m_Text));
49.                         prefsE.commit();
50.                     } catch (Exception e) {
51.                         e.printStackTrace();
52.                     }
53.                 }
54.             }
55.         });
56.         builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener()
{@
57.             Override public void onClick(DialogInterface dialog, int which) {
58.                 dialog.cancel();
59.                 witch.setChecked(false);
60.             }
61.         });
62.         builder.show(); //while(m_Text == "****"){
63.     }
64. }
65. });
66. }
67. }

```

Bus_Data.java

```

1. package com.example.baltu.myapplication.DataTypes;
2. import android.content.ContentValues;
3. import com.example.baltu.myapplication.DB_Constructor.Local_DB.Buss_Table;
4. import com.example.baltu.myapplication.DB_Constructor.User_DB.TasksTable;
5. import java.util.UUID;
6. /** * Created by Baltu on 2017-04-24. */
7. public class Bus_Data {@
8.     com.google.gson.annotations.SerializedName("id") private String ID;
9.     private Integer Line_Number;
10.    private Integer Weekday;
11.    private Integer To_from;@
12.    com.google.gson.annotations.SerializedName("time") private String Time;
13.    public Bus_Data(String ID, Integer line_Number, Integer weekday, Integer to_from, String t
ime) {
14.        if (ID == null) ID = UUID.randomUUID().toString();
15.        this.ID = ID;
16.        Line_Number = line_Number;
17.        Weekday = weekday;
18.        To_from = to_from;
19.        Time = time;
20.    }
21.    public Bus_Data() {
22.        if (ID == null) ID = UUID.randomUUID().toString();
23.    }
24.    public void setTime(String time) {
25.        Time = time;
26.    }

```

```

27.     public String getID() {
28.         return ID;
29.     }
30.     public void setID(String ID) {
31.         this.ID = ID;
32.     }
33.     public Integer getLine_Number() {
34.         return Line_Number;
35.     }
36.     public void setLine_Number(Integer line_Number) {
37.         Line_Number = line_Number;
38.     }
39.     public Integer getWeekday() {
40.         return Weekday;
41.     }
42.     public void setWeekday(Integer weekday) {
43.         Weekday = weekday;
44.     }
45.     public Integer getTo_from() {
46.         return To_from;
47.     }
48.     public void setTo_from(Integer to_from) {
49.         To_from = to_from;
50.     }
51.     public String getTime() {
52.         return Time;
53.     }
54.     public ContentValues toValues() {
55.         ContentValues values = new ContentValues(5);
56.         values.put(Buss_Table.COLUMN_ID, ID);
57.         values.put(Buss_Table.COLUMN_LINE_NUMBER, Line_Number);
58.         values.put(Buss_Table.COLUMN_WEEKDAY, Weekday);
59.         values.put(Buss_Table.COLUMN_TO_FROM, To_from);
60.         values.put(Buss_Table.COLUMN_TIME, Time);
61.         return values;
62.     }
63. }

```

Calender_Data.java

```

1. package com.example.baltu.myapplication.DataTypes;
2. import android.content.ContentValues;
3. import com.example.baltu.myapplication.DB_Constructor.Local_DB.Calender_Table;
4. import java.util.Calendar;
5. import java.util.Date;
6. /** * Created by Baltu on 2017-04-24. */
7. public class Calender_Data {
8.     private String ID;
9.     private String Event_Name;
10.    private Long StartDate;
11.    private Long EndDate;
12.    public Calender_Data() {}
13.    public Calender_Data(String ID, String event_Name, Long startDate, Long endDate) {
14.        this.ID = ID;
15.        Event_Name = event_Name;
16.        StartDate = startDate;
17.        EndDate = endDate;
18.    }
19.    public String getID() {
20.        return ID;

```

```

21.     }
22.     public void setID(String ID) {
23.         this.ID = ID;
24.     }
25.     public String getEvent_Name() {
26.         return Event_Name;
27.     }
28.     public void setEvent_Name(String event_Name) {
29.         Event_Name = event_Name;
30.     }
31.     public Long getStartDate() {
32.         return StartDate;
33.     }
34.     public void setStartDate(Long startDate) {
35.         StartDate = startDate;
36.     }
37.     public Long getEndDate() {
38.         return EndDate;
39.     }
40.     public void setEndDate(Long endDate) {
41.         EndDate = endDate;
42.     }
43.     public ContentValues toValues() {
44.         ContentValues values = new ContentValues(4);
45.         values.put(Calender_Table.COLUMN_ID, ID);
46.         values.put(Calender_Table.COLUMN_Event_Name, Event_Name);
47.         values.put(Calender_Table.COLUMN_SDATE, StartDate);
48.         values.put(Calender_Table.COLUMN_EDATE, EndDate);
49.         return values;
50.     }
51. }

```

Exam_class.java

```

1. package com.example.baltu.myapplication.DataTypes;
2. import android.content.ContentValues;
3. import com.example.baltu.myapplication.DB_Constructor.User_DB.ExamsTable;
4. import java.util.UUID;
5. /** * Created by Baltu on 2017-04-24. */
6. public class Exam_class {
7.     private String ID;
8.     private String Course;
9.     private String discription;
10.    private long Exam_Date;
11.    private String Notes;
12.    public String getID() {
13.        return ID;
14.    }
15.    public void setID(String ID) {
16.        this.ID = ID;
17.    }
18.    public String getCourse() {
19.        return Course;
20.    }
21.    public void setCourse(String course) {
22.        Course = course;
23.    }
24.    public String getDiscription() {
25.        return discription;
26.    }
27.    public void setDiscription(String discription) {
28.        this.discription = discription;

```

```

29.     }
30.     public long getExam_Date() {
31.         return Exam_Date;
32.     }
33.     public void setExam_Date(long deadline_Date) {
34.         Exam_Date = deadline_Date;
35.     }
36.     public String getNotes() {
37.         return Notes;
38.     }
39.     public void setNotes(String notes) {
40.         Notes = notes;
41.     }
42.     public Exam_class(String ID, String course, String discription, long deadline_Date, String
notes) {
43.         if (ID == null) {
44.             ID = UUID.randomUUID().toString();
45.         }
46.         this.ID = ID;
47.         Course = course;
48.         this.discription = discription;
49.         Exam_Date = deadline_Date;
50.         Notes = notes;
51.     }
52.     public Exam_class() {
53.         ID = UUID.randomUUID().toString();
54.     }
55.     public ContentValues toValues() {
56.         ContentValues values = new ContentValues(5);
57.         values.put(ExamsTable.COLUMN_ID, ID);
58.         values.put(ExamsTable.COLUMN_CourseNAME, Course);
59.         values.put(ExamsTable.COLUMN_Discription, discription);
60.         values.put(ExamsTable.COLUMN_EXAMDATE, Exam_Date);
61.         values.put(ExamsTable.COLUMN_NOTES, Notes);
62.         return values;
63.     }
64. }

```

InfoContent.java

```

1. package com.example.baltu.myapplication.DataTypes;
2. import java.util.ArrayList;
3. import java.util.HashMap;
4. import java.util.List;
5. import java.util.Map;
6. /** * Helper class for providing sample content for user interfaces created by * Android templ
ate wizards. * <p> * TODO: Replace all uses of this class before publishing your app. */
7. public class InfoContent {
8.     /** * An array of sample (dummy) items. */
9.     public static final List < InfoItem > ITEMS = new ArrayList < InfoItem > ();
10.    /** * A map of sample (dummy) items, by ID. */
11.    public static final Map < String, InfoItem > ITEM_MAP = new HashMap < String, InfoItem > (
);
12.    private static final int COUNT = 15;
13.    static { // Add some sample items.
14.        for (int i = 1; i <= COUNT; i++) {
15.            addItem(createDummyItem(i));
16.        }
17.    }
18.    private static void addItem(InfoItem item) {
19.        ITEMS.add(item);
20.        ITEM_MAP.put(item.id, item);

```



```

21.     }
22.     private static InfoItem createDummyItem(int position) {
23.         return new InfoItem(String.valueOf(position), "Info For Students No" + position, makeD
etails(position));
24.     }
25.     private static String makeDetails(int position) {
26.         StringBuilder builder = new StringBuilder();
27.         builder.append("Details about Item: ").append(position);
28.         builder.append("\nTimeTables or Phone Number here");
29.         return builder.toString();
30.     }
31.     /**      * A dummy item representing a piece of content.      */
32.     public static class InfoItem {
33.         public final String id;
34.         public final String content;
35.         public final String details;
36.         public InfoItem(String id, String content, String details) {
37.             this.id = id;
38.             this.content = content;
39.             this.details = details;
40.         }
41.         @Override public String toString() {
42.             return content;
43.         }
44.     }
45. }

```

Information.java

```

1. package com.example.baltu.myapplication.DataTypes;
2. /** * Created by Baltu on 2017-05-20. */
3. public class Information {
4.     String ID;
5.     String Title;
6.     String Content;
7.     String Image;
8.     public Information(String ID, String title, String content, String image) {
9.         this.ID = ID;
10.        Title = title;
11.        Content = content;
12.        Image = image;
13.    }
14.    public String getID() {
15.        return ID;
16.    }
17.    public void setID(String ID) {
18.        this.ID = ID;
19.    }
20.    public String getTitle() {
21.        return Title;
22.    }
23.    public void setTitle(String title) {
24.        Title = title;
25.    }
26.    public String getContent() {
27.        return Content;
28.    }
29.    public void setContent(String content) {
30.        Content = content;
31.    }
32.    public String getImage() {
33.        return Image;

```

```

34.     }
35.     public void setImage(String image) {
36.         Image = image;
37.     }
38. }

```

News_Class.java

```

1. package com.example.baltu.myapplication.DataTypes;
2. import java.util.Date;
3. /** * Created by Baltu on 2017-04-24. */
4. public class News_Class {
5.     private String Title;
6.     private String Link;
7.     private String Description;
8.     private Date mDate;
9.     public News_Class(String title, String link, String description, java.util.Date date) {
10.         Title = title;
11.         Link = link;
12.         Description = description;
13.         mDate = date;
14.     }
15.     public String getTitle() {
16.         return Title;
17.     }
18.     public void setTitle(String title) {
19.         Title = title;
20.     }
21.     public String getLink() {
22.         return Link;
23.     }
24.     public void setLink(String link) {
25.         Link = link;
26.     }
27.     public String getDescription() {
28.         return Description;
29.     }
30.     public void setDescription(String description) {
31.         Description = description;
32.     }
33.     public java.util.Date getDate() {
34.         return mDate;
35.     }
36.     public void setDate(java.util.Date date) {
37.         mDate = date;
38.     }
39. }

```

Tasks_Data.java

```

1. package com.example.baltu.myapplication.DataTypes;
2. import android.content.ContentValues;
3. import com.example.baltu.myapplication.DB_Constructor.User_DB.TasksTable;
4. import java.util.UUID;
5. /** * Created by Baltu on 2017-04-24. */
6. public class Tasks_Data {
7.     private String ID;
8.     private String Course;
9.     private String discription;
10.    private long Deadline_Date;
11.    private String Notes;
12.    public String getID() {
13.        return ID;

```

```

14.     }
15.     public void setID(String ID) {
16.         this.ID = ID;
17.     }
18.     public String getCourse() {
19.         return Course;
20.     }
21.     public void setCourse(String course) {
22.         Course = course;
23.     }
24.     public String getDiscription() {
25.         return discription;
26.     }
27.     public void setDiscription(String discription) {
28.         this.discription = discription;
29.     }
30.     public long getDeadline_Date() {
31.         return Deadline_Date;
32.     }
33.     public void setDeadline_Date(long deadline_Date) {
34.         Deadline_Date = deadline_Date;
35.     }
36.     public String getNotes() {
37.         return Notes;
38.     }
39.     public void setNotes(String notes) {
40.         Notes = notes;
41.     }
42.     public Tasks_Data(String ID, String course, String discription, long deadline_Date, String
notes) {
43.         if (ID == null) {
44.             ID = UUID.randomUUID().toString();
45.         }
46.         this.ID = ID;
47.         Course = course;
48.         this.discription = discription;
49.         Deadline_Date = deadline_Date;
50.         Notes = notes;
51.     }
52.     public Tasks_Data() {
53.         ID = UUID.randomUUID().toString();
54.     }
55.     public ContentValues toValues() {
56.         ContentValues values = new ContentValues(5);
57.         values.put(TasksTable.COLUMN_ID, ID);
58.         values.put(TasksTable.COLUMN_CourseNAME, Course);
59.         values.put(TasksTable.COLUMN_Discription, discription);
60.         values.put(TasksTable.COLUMN_ENDDATE, Deadline_Date);
61.         values.put(TasksTable.COLUMN_NOTES, Notes);
62.         return values;
63.     }
64. }

```

TimeTable_Classes.java

```

1. package com.example.baltu.myapplication.DataTypes;
2. import android.content.ContentValues;
3. import com.example.baltu.myapplication.DB_Constructor.User_DB.Courses_Table;
4. import java.util.UUID;
5. /** * Created by Baltu on 2017-04-04. */
6. public class TimeTable_Classes {
7.     private String ID;

```

```

8.     private String Course;
9.     private String starting_time;
10.    private String finishing_time;
11.    private int Day_of_week;
12.    private String Notes;
13.    public TimeTable_Classes() {
14.        if (ID == null) {
15.            ID = UUID.randomUUID().toString();
16.        }
17.    }
18.    public TimeTable_Classes(String ID, String course, String starting_time, String finishing_
time, int day_of_week, String notes) {
19.        if (ID == null) {
20.            ID = UUID.randomUUID().toString();
21.        }
22.        this.ID = ID;
23.        Course = course;
24.        this.starting_time = starting_time;
25.        this.finishing_time = finishing_time;
26.        Day_of_week = day_of_week;
27.        Notes = notes;
28.    }
29.    public String getID() {
30.        return ID;
31.    }
32.    public String getCourse() {
33.        return Course;
34.    }
35.    public String getStarting_time() {
36.        return starting_time;
37.    }
38.    public String getFinishing_time() {
39.        return finishing_time;
40.    }
41.    public int getDay_of_week() {
42.        return Day_of_week;
43.    }
44.    public String getNotes() {
45.        return Notes;
46.    }
47.    public void setID(String ID) {
48.        this.ID = ID;
49.    }
50.    public void setCourse(String course) {
51.        Course = course;
52.    }
53.    public void setStarting_time(String starting_time) {
54.        this.starting_time = starting_time;
55.    }
56.    public void setFinishing_time(String finishing_time) {
57.        this.finishing_time = finishing_time;
58.    }
59.    public void setDay_of_week(int day_of_week) {
60.        Day_of_week = day_of_week;
61.    }
62.    public void setNotes(String notes) {
63.        Notes = notes;
64.    }
65.    public ContentValues toValues() {
66.        ContentValues values = new ContentValues(6);
67.        values.put(Courses_Table.COLUMN_ID, ID);

```

```

68.     values.put(Courses_Table.COLUMN_DAY, Day_of_week);
69.     values.put(Courses_Table.COLUMN_NAME, Course);
70.     values.put(Courses_Table.COLUMN_NOTES, Notes);
71.     values.put(Courses_Table.COLUMN_STARTTIME, starting_time);
72.     values.put(Courses_Table.COLUMN_ENDTIME, finishing_time);
73.     return values;
74. }
75. }

```

University_Locations.java

```

1. package com.example.baltu.myapplication.DataTypes;
2. import android.content.ContentValues;
3. import android.content.pm.LabeledIntent;
4. import com.example.baltu.myapplication.DB_Constructor.Local_DB.MapLocations_Table;
5. /** * Created by Baltu on 2017-05-23. */
6. public class University_Locations {
7.     private String ID;
8.     private String ShortName;
9.     private String LongName;
10.    private String Information;
11.    private Double Latitude;
12.    private Double Longitude;@
13.    com.google.gson.annotations.SerializedName("ImageUrl") private String Image_Url;
14.    public String getID() {
15.        return ID;
16.    }
17.    public void setID(String ID) {
18.        this.ID = ID;
19.    }
20.    public String getShortName() {
21.        return ShortName;
22.    }
23.    public void setShortName(String shortName) {
24.        ShortName = shortName;
25.    }
26.    public String getLongName() {
27.        return LongName;
28.    }
29.    public void setLongName(String longName) {
30.        LongName = longName;
31.    }
32.    public String getInformation() {
33.        return Information;
34.    }
35.    public void setInformation(String information) {
36.        Information = information;
37.    }
38.    public Double getLatitude() {
39.        return Latitude;
40.    }
41.    public void setLatitude(Double latitude) {
42.        Latitude = latitude;
43.    }
44.    public Double getLongitude() {
45.        return Longitude;
46.    }
47.    public void setLongitude(Double longitude) {
48.        Longitude = longitude;
49.    }
50.    public String getImage_Url() {
51.        return Image_Url;

```

```

52.     }
53.     public void setImage_Url(String image_Url) {
54.         Image_Url = image_Url;
55.     }
56.     public University_Locations(String ID, String shortName, String longName, String informati
on, Double latitude, Double longitude, String image_Url) {
57.         this.ID = ID;
58.         ShortName = shortName;
59.         LongName = longName;
60.         Information = information;
61.         Latitude = latitude;
62.         Longitude = longitude;
63.         Image_Url = image_Url;
64.     }
65.     public University_Locations() {}
66.     public ContentValues ToValues() {
67.         ContentValues Values = new ContentValues(7);
68.         Values.put(MapLocations_Table.COLUMN_ID, ID);
69.         Values.put(MapLocations_Table.COLUMN_SHORTNAME, ShortName);
70.         Values.put(MapLocations_Table.COLUMN_LONGNAME, LongName);
71.         Values.put(MapLocations_Table.COLUMN_INFO, Information);
72.         Values.put(MapLocations_Table.COLUMN_LATITUDE, Latitude);
73.         Values.put(MapLocations_Table.COLUMN_LONGITUDE, Longitude);
74.         Values.put(MapLocations_Table.COLUMN_IMAGE_URL, Image_Url);
75.         return Values;
76.     }
77. }

```

Local_DB_Helper.java

```

1. package com.example.baltu.myapplication.DB_Constructor;
2. import android.content.Context;
3. import android.database.sqlite.SQLiteDatabase;
4. import android.database.sqlite.SQLiteOpenHelper;
5. import android.widget.Toast;
6. import com.example.baltu.myapplication.DB_Constructor.*;
7. import com.example.baltu.myapplication.DB_Constructor.Local_DB.Buss_Table;
8. import com.example.baltu.myapplication.DB_Constructor.Local_DB.Calender_Table;
9. import com.example.baltu.myapplication.DB_Constructor.Local_DB.Information_Table;
10. import com.example.baltu.myapplication.DB_Constructor.Local_DB.MapLocations_Table;
11. import com.example.baltu.myapplication.DB_Constructor.User_DB.Courses_Table;
12. import com.example.baltu.myapplication.DataTypes.Bus_Data;
13. import java.io.FileOutputStream;
14. import java.io.IOException;
15. import java.io.InputStream;
16. import java.io.OutputStream;
17. import java.util.Calendar;
18. import java.util.List;
19. /** * Created by Baltu on 2017-05-06. */
20. public class Local_DB_Helper extends SQLiteOpenHelper {
21.     static String DB_PATH = "/data/data/com.example.baltu.myapplication/databases/";
22.     public static final String DB_FILENAME = "LocalDb.db";
23.     public static final int DB_Version = 1;
24.     Context mcontext;
25.     public Local_DB_Helper(Context context) {
26.         super(context, DB_FILENAME, null, DB_Version);
27.         mcontext = context;
28.     }
29.     @Override public void onCreate(SQLiteDatabase db) {
30.         db.execSQL(Buss_Table.SQL_CREATE);
31.         db.execSQL(Calender_Table.SQL_CREATE);
32.         db.execSQL(Information_Table.SQL_CREATE);

```

```

33.         db.execSQL(MapLocations_Table.SQL_CREATE);
34.     }@
35.     Override public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {}@
36.     Override public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
37.         super.onDowngrade(db, oldVersion, newVersion);
38.     }
39. }

```

User_DB_Helper.java

```

1. package com.example.baltu.myapplication.DB_Constructor;
2. import android.content.Context;
3. import android.database.sqlite.SQLiteDatabase;
4. import android.database.sqlite.SQLiteOpenHelper;
5. import com.example.baltu.myapplication.DB_Constructor.User_DB.Courses_Table;
6. import com.example.baltu.myapplication.DB_Constructor.User_DB.ExamsTable;
7. import com.example.baltu.myapplication.DB_Constructor.User_DB.Settings;
8. import com.example.baltu.myapplication.DB_Constructor.User_DB.TasksTable;
9. /** * Created by Baltu on 2017-05-06. */
10. public class User_DB_Helper extends SQLiteOpenHelper {
11.     public static final String DB_FILENAME = "UserDb.db";
12.     public static final int DB_Version = 1;
13.     public User_DB_Helper(Context context) {
14.         super(context, DB_FILENAME, null, DB_Version);
15.     }@
16.     Override public void onCreate(SQLiteDatabase db) {
17.         db.execSQL(Courses_Table.SQL_CREATE);
18.         db.execSQL(TasksTable.SQL_CREATE);
19.         db.execSQL(ExamsTable.SQL_CREATE);
20.     }@
21.     Override public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {}
22. }

```

Buss_Table.java

```

1. package com.example.baltu.myapplication.DB_Constructor.Local_DB;
2. /** * Created by Baltu on 2017-05-11. */
3. public class Buss_Table {
4.     public static final String TABLE_NAME = "buss";
5.     public static final String COLUMN_ID = "id";
6.     public static final String COLUMN_LINE_NUMBER = "Line_Number";
7.     public static final String COLUMN_WEEKDAY = "Weekday";
8.     public static final String COLUMN_TO_FROM = "To_from";
9.     public static final String COLUMN_TIME = "time";
10.    public static final String[] allColoumns = {
11.        COLUMN_ID, COLUMN_LINE_NUMBER, COLUMN_WEEKDAY, COLUMN_TO_FROM, COLUMN_TIME
12.    };
13.    public static final String SQL_CREATE = "CREATE TABLE " + TABLE_NAME + "(" + COLUMN_ID + " TEXT PRIMARY KEY," + COLUMN_LINE_NUMBER + " INTEGER," + COLUMN_WEEKDAY + " INTEGER," + COLUMN_TO_FROM + " INTEGER," + COLUMN_TIME + " TEXT" + ");";
14.    public static final String SQL_DELETE = "DROP TABLE " + TABLE_NAME;
15. }

```

Calender_Table.java

```

1. package com.example.baltu.myapplication.DB_Constructor.Local_DB;
2. /** * Created by Baltu on 2017-05-11. */
3. public class Calender_Table {
4.     public static final String TABLE_NAME = "calender";
5.     public static final String COLUMN_ID = "ID";
6.     public static final String COLUMN_Event_Name = "event_name";
7.     public static final String COLUMN_SDATE = "start_date";
8.     public static final String COLUMN_EDATE = "end_date";
9.     public static final String[] allColoumns = {

```



```

10.         COLUMN_ID, COLUMN_Event_Name, COLUMN_SDATE, COLUMN_EDATE
11.     };
12.     public static final String SQL_CREATE = "CREATE TABLE " + TABLE_NAME + "(" + COLUMN_ID + "
TEXT PRIMARY KEY," + COLUMN_Event_Name + " TEXT," + COLUMN_SDATE + " LONG," + COLUMN_EDATE +
" LONG" + ");";
13.     public static final String SQL_DELETE = "DROP TABLE " + TABLE_NAME;
14. }

```

Information_Table.java

```

1. package com.example.baltu.myapplication.DB_Constructor.Local_DB;
2. /** * Created by Baltu on 2017-05-11. */
3. public class Information_Table {
4.     public static final String TABLE_NAME = "info";
5.     public static final String COLUMN_ID = "ID";
6.     public static final String COLUMN_TITLE = "Title";
7.     public static final String COLUMN_CONTENT = "Content";
8.     public static final String COLUMN_Image = "Image";
9.     public static final String[] allColumns = {
10.         COLUMN_ID, COLUMN_TITLE, COLUMN_CONTENT, COLUMN_Image
11.     };
12.     public static final String SQL_CREATE = "CREATE TABLE " + TABLE_NAME + "(" + COLUMN_ID + "
TEXT PRIMARY KEY," + COLUMN_TITLE + " TEXT," + COLUMN_CONTENT + " TEXT," + COLUMN_Image + " T
EXT" + ");";
13.     public static final String SQL_DELETE = "DROP TABLE " + TABLE_NAME;
14. }

```

MapLocations_Table.java

```

1. package com.example.baltu.myapplication.DB_Constructor.Local_DB;
2. /** * Created by Baltu on 2017-05-11. */
3. public class MapLocations_Table {
4.     public static final String TABLE_NAME = "MapLocations";
5.     public static final String COLUMN_ID = "ID";
6.     public static final String COLUMN_SHORTNAME = "ShortName";
7.     public static final String COLUMN_LONGNAME = "LongName";
8.     public static final String COLUMN_INFO = "Information";
9.     public static final String COLUMN_LATITUDE = "Latitude";
10.    public static final String COLUMN_LONGITUDE = "Longitude";
11.    public static final String COLUMN_IMAGE_URL = "ImageUrl";
12.    public static final String[] allColumns = {
13.        COLUMN_ID, COLUMN_SHORTNAME, COLUMN_LONGNAME, COLUMN_INFO, COLUMN_LATITUDE, COLUMN_LON
GITUDE, COLUMN_IMAGE_URL
14.    };
15.    public static final String SQL_CREATE = "CREATE TABLE " + TABLE_NAME + "(" + COLUMN_ID + "
TEXT PRIMARY KEY," + COLUMN_SHORTNAME + " TEXT," + COLUMN_LONGNAME + " TEXT," + COLUMN_INFO +
" TEXT," + COLUMN_LATITUDE + " REAL," + COLUMN_LONGITUDE + " REAL," + COLUMN_IMAGE_URL + " TE
XT" + ");";
16.    public static final String SQL_DELETE = "DROP TABLE " + TABLE_NAME;
17. }

```

Courses_Table.java

```

1. package com.example.baltu.myapplication.DB_Constructor.User_DB;
2. public class Courses_Table {
3.     public static final String TABLE_ITEMS = "lectures";
4.     public static final String COLUMN_ID = "itemId";
5.     public static final String COLUMN_NAME = "itemName";
6.     public static final String COLUMN_NOTES = "notes";
7.     public static final String COLUMN_STARTTIME = "starttime";
8.     public static final String COLUMN_ENDTIME = "wndtime";
9.     public static final String COLUMN_DAY = "day";
10.    public static final String[] allColumns = {
11.        COLUMN_ID, COLUMN_NAME, COLUMN_NOTES, COLUMN_STARTTIME, COLUMN_ENDTIME, COLUMN_DAY

```

```

12.     };
13.     public static final String SQL_CREATE = "CREATE TABLE " + TABLE_ITEMS + "(" + COLUMN_ID +
" TEXT PRIMARY KEY," + COLUMN_NAME + " TEXT," + COLUMN_NOTES + " TEXT," + COLUMN_STARTTIME + "
TEXT," + COLUMN_ENDTIME + " TEXT," + COLUMN_DAY + " TEXT" + ");";
14.     public static final String SQL_DELETE = "DROP TABLE " + TABLE_ITEMS;
15. }

```

ExamsTable.java

```

1. package com.example.baltu.myapplication.DB_Constructor.User_DB;
2. public class ExamsTable {
3.     public static final String TABLE_NAME = "exams";
4.     public static final String COLUMN_ID = "id";
5.     public static final String COLUMN_CourseNAME = "course";
6.     public static final String COLUMN_Discription = "discription";
7.     public static final String COLUMN_EXAMDATE = "exam_date";
8.     public static final String COLUMN_NOTES = "notes";
9.     public static final String[] allColoumns = {
10.         COLUMN_ID, COLUMN_CourseNAME, COLUMN_Discription, COLUMN_NOTES, COLUMN_EXAMDATE, COLUM
N_NOTES
11.     };
12.     public static final String SQL_CREATE = "CREATE TABLE " + TABLE_NAME + "(" + COLUMN_ID + "
TEXT PRIMARY KEY," + COLUMN_CourseNAME + " TEXT," + COLUMN_Discription + " TEXT," + COLUMN_EX
AMDATE + " INTEGER ," + COLUMN_NOTES + " TEXT" + ");";
13.     public static final String SQL_DELETE = "DROP TABLE " + TABLE_NAME;
14. }

```

Settings.java

```

1. package com.example.baltu.myapplication.DB_Constructor.User_DB;
2. public class Settings {
3.     public static final String TABLE_NAME = "settings";
4.     public static final String COLUMN_ID = "ID";
5.     public static final String COLUMN_OPTIONNAME = "option";
6.     public static final String COLUMN_Discription = "discription";
7.     public static final String COLUMN_CONTENT = "content";
8.     public static final String[] allColoumns = {
9.         COLUMN_ID, COLUMN_OPTIONNAME, COLUMN_Discription, COLUMN_CONTENT
10.     };
11.     public static final String SQL_CREATE = "CREATE TABLE " + TABLE_NAME + "(" + COLUMN_ID + "
TEXT PRIMARY KEY," + COLUMN_OPTIONNAME + " TEXT," + COLUMN_Discription + " TEXT," + COLUMN_CO
NTENT + " TEXT " + ");";
12.     public static final String SQL_DELETE = "DROP TABLE " + TABLE_NAME;
13. }

```

TasksTable.java

```

1. package com.example.baltu.myapplication.DB_Constructor.User_DB;
2. public class TasksTable {
3.     public static final String TABLE_NAME = "tasks";
4.     public static final String COLUMN_ID = "ID";
5.     public static final String COLUMN_CourseNAME = "course";
6.     public static final String COLUMN_Discription = "discription";
7.     public static final String COLUMN_ENDDATE = "deadline_date";
8.     public static final String COLUMN_NOTES = "notes";
9.     public static final String[] allColoumns = {
10.         COLUMN_ID, COLUMN_CourseNAME, COLUMN_Discription, COLUMN_NOTES, COLUMN_ENDDATE, COLUMN
_NOTES
11.     };
12.     public static final String SQL_CREATE = "CREATE TABLE " + TABLE_NAME + "(" + COLUMN_ID + "
TEXT PRIMARY KEY," + COLUMN_CourseNAME + " TEXT," + COLUMN_Discription + " TEXT," + COLUMN_EN
DDATE + " INTEGER ," + COLUMN_NOTES + " TEXT" + ");";
13.     public static final String SQL_DELETE = "DROP TABLE " + TABLE_NAME;
14. }

```

UserDB_Provider.java

```
1. package com.example.baltu.myapplication.Data_Provider;
2. import android.content.ContentValues;
3. import android.content.Context;
4. import android.database.Cursor;
5. import android.database.DatabaseUtils;
6. import android.database.sqlite.SQLiteDatabase;
7. import android.database.sqlite.SQLiteException;
8. import android.database.sqlite.SQLiteOpenHelper;
9. import com.example.baltu.myapplication.DB_Constructor.User_DB.Courses_Table;
10. import com.example.baltu.myapplication.DB_Constructor.User_DB.ExamsTable;
11. import com.example.baltu.myapplication.DB_Constructor.User_DB.Settings;
12. import com.example.baltu.myapplication.DB_Constructor.User_DB.TasksTable;
13. import com.example.baltu.myapplication.DB_Constructor.User_DB_Helper;
14. import com.example.baltu.myapplication.DataTypes.Exam_class;
15. import com.example.baltu.myapplication.DataTypes.Tasks_Data;
16. import com.example.baltu.myapplication.DataTypes.TimeTable_Classes;
17. import com.scottyab.aesencrypt.AESCrypt;
18. import java.security.GeneralSecurityException;
19. import java.util.ArrayList;
20. import java.util.List;
21. public class UserDB_Provider {
22.     private Context mContext;
23.     private final static String passid = "15968987559";
24.     private SQLiteDatabase UserDb;
25.     SQLiteOpenHelper mUserDBhelper;
26.     public UserDB_Provider(Context context) {
27.         this.mContext = context;
28.         mUserDBhelper = new User_DB_Helper(mContext);
29.         UserDb = mUserDBhelper.getWritableDatabase();
30.     }
31.     public void open() {
32.         UserDb = mUserDBhelper.getWritableDatabase();
33.     }
34.     public void close() {
35.         mUserDBhelper.close();
36.     } //----- //-----
37.     public TimeTable_Classes addnewcourse(TimeTable_Classes ttc) {
38.         UserDb.insert(Courses_Table.TABLE_ITEMS, null, ttc.toValues());
39.         return ttc;
40.     }
41.     public long numberofcourses(TimeTable_Classes ttc) {
42.         return DatabaseUtils.queryNumEntries(UserDb, Courses_Table.TABLE_ITEMS, "itemId=?", ne
w String[] {
43.             ttc.getID()
44.         });
45.     }
46.     public TimeTable_Classes updatecourse(TimeTable_Classes ttc) {
47.         UserDb.update(Courses_Table.TABLE_ITEMS, ttc.toValues(), "itemId=?", new String[] {
48.             ttc.getID()
49.         });
50.         return ttc;
51.     }
52.     public boolean deletecourse(String id) {
53.         UserDb.delete(Courses_Table.TABLE_ITEMS, "itemId=?", new String[] {
54.             id
55.         });
56.         return true;
57.     }
```

```

58.     public List < TimeTable_Classes > getcourses(int day) {
59.         List < TimeTable_Classes > nelist = new ArrayList < > ();
60.         String[] arg = {
61.             Integer.toString(day)
62.         };
63.         Cursor cur = UserDb.query(Courses_Table.TABLE_ITEMS, Courses_Table.allColumns, "day=?", arg, null, null, "starttime");
64.         while (cur.moveToNext()) {
65.             TimeTable_Classes ttc = new TimeTable_Classes();
66.             ttc.setID(cur.getString(cur.getColumnIndex(Courses_Table.COLUMN_ID)));
67.             ttc.setCourse(cur.getString(cur.getColumnIndex(Courses_Table.COLUMN_NAME)));
68.             ttc.setDay_of_week(cur.getInt(cur.getColumnIndex(Courses_Table.COLUMN_DAY)));
69.             ttc.setStarting_time(cur.getString(cur.getColumnIndex(Courses_Table.COLUMN_STARTTIME)));
70.             ttc.setFinishing_time(cur.getString(cur.getColumnIndex(Courses_Table.COLUMN_ENDTIME)));
71.             ttc.setNotes(cur.getString(cur.getColumnIndex(Courses_Table.COLUMN_NOTES)));
72.             nelist.add(ttc);
73.         }
74.         return nelist;
75.     } //=====
=====
76.     public Tasks_Data addnewTask(Tasks_Data ttc) {
77.         UserDb.insert(TasksTable.TABLE_NAME, null, ttc.toValues());
78.         return ttc;
79.     }
80.     public Tasks_Data updateTask(Tasks_Data ttc) {
81.         UserDb.update(TasksTable.TABLE_NAME, ttc.toValues(), "ID=?", new String[] {
82.             ttc.getID()
83.         });
84.         return ttc;
85.     }
86.     public int DeleteTask(String id) {
87.         String[] arg = {
88.             id
89.         };
90.         return UserDb.delete(TasksTable.TABLE_NAME, "ID=?", arg);
91.     }
92.     public List < Tasks_Data > GetTasks(int x) {
93.         List < Tasks_Data > nelist = new ArrayList < > ();
94.         String[] arg = {
95.             Long.toString((new java.util.Date()).getTime())
96.         };
97.         Cursor cur;
98.         if (x == 0) {
99.             cur = UserDb.query(TasksTable.TABLE_NAME, TasksTable.allColumns, "deadline_date>?", arg, null, null, TasksTable.COLUMN_ENDDATE);
100.        } else {
101.            cur = UserDb.query(TasksTable.TABLE_NAME, TasksTable.allColumns, "deadline_date<=?", arg, null, null, TasksTable.COLUMN_ENDDATE);
102.        }
103.        while (cur.moveToNext()) {
104.            Tasks_Data ttc = new Tasks_Data();
105.            ttc.setID(cur.getString(cur.getColumnIndex(TasksTable.COLUMN_ID)));
106.            ttc.setCourse(cur.getString(cur.getColumnIndex(TasksTable.COLUMN_COURSE_NAME)));
107.            ttc.setDeadline_Date(cur.getLong(cur.getColumnIndex(TasksTable.COLUMN_ENDDATE)));
108.            ttc.setDiscription(cur.getString(cur.getColumnIndex(TasksTable.COLUMN_DESCRIPTION)));

```

```

109.         ttc.setNotes(cur.getString(cur.getColumnIndex(ExamsTable.COLUMN_NOTES))
110.     );
111.         nelist.add(ttc);
112.     }
113.     return nelist;
114. } //=====
=====
114.     public Exam_class addnewExam(Exam_class ttc) {
115.         UserDb.insert(ExamsTable.TABLE_NAME, null, ttc.toValues());
116.         return ttc;
117.     }
118.     public Exam_class updateExam(Exam_class ttc) {
119.         UserDb.update(ExamsTable.TABLE_NAME, ttc.toValues(), "id=?", new String[] {
120.             ttc.getID()
121.         });
122.         return ttc;
123.     }
124.     public int DeleteExam(String id) {
125.         String[] arg = {
126.             id
127.         };
128.         return UserDb.delete(ExamsTable.TABLE_NAME, "id=?", arg);
129.     }
130.     public List < Exam_class > GetExams(int x) {
131.         List < Exam_class > nelist = new ArrayList < > ();
132.         String[] arg = {
133.             Long.toString((new java.util.Date()).getTime())
134.         };
135.         Cursor cur;
136.         if (x == 0) {
137.             cur = UserDb.query(ExamsTable.TABLE_NAME, ExamsTable.allColumns, "exam
138. _date>?", arg, null, null, ExamsTable.COLUMN_EXAMDATE);
139.         } else {
140.             cur = UserDb.query(ExamsTable.TABLE_NAME, ExamsTable.allColumns, "exam
141. _date<=?", arg, null, null, ExamsTable.COLUMN_EXAMDATE);
142.         }
143.         while (cur.moveToNext()) {
144.             Exam_class ttc = new Exam_class();
145.             ttc.setID(cur.getString(cur.getColumnIndex(ExamsTable.COLUMN_ID)));
146.             ttc.setCourse(cur.getString(cur.getColumnIndex(ExamsTable.COLUMN_Course
147. NAME)));
148.             ttc.setExam_Date(cur.getLong(cur.getColumnIndex(ExamsTable.COLUMN_EXAMD
149. ATE)));
150.             ttc.setDiscription(cur.getString(cur.getColumnIndex(ExamsTable.COLUMN_D
151. iscription)));
152.             ttc.setNotes(cur.getString(cur.getColumnIndex(ExamsTable.COLUMN_NOTES))
153.         );
154.             nelist.add(ttc);
155.         }
156.         return nelist;
157.     } //=====
===== //Settings
152.     public int isPasswordOn() {
153.         Cursor cur;
154.         String[] arg = {
155.             passid
156.         };
157.         cur = UserDb.query(Settings.TABLE_NAME, Settings.allColumns, Settings.COLUMN_I
158. D + "=?", arg, null, null, null);
159.         if (cur.getCount() == 1) {

```

```

159.         if (cur.getString(cur.getColumnIndex(Settings.COLUMN_Discription)).equals("
on")) {
160.             return 1;
161.         } else if (cur.getString(cur.getColumnIndex(Settings.COLUMN_Discription)).e
quals("off")) {
162.             return 0;
163.         } else return 2;
164.     } else return 2;
165. }
166. public String GetPasswordon() {
167.     Cursor cur;
168.     String[] arg = {
169.         passid
170.     };
171.     cur = UserDb.query(Settings.TABLE_NAME, Settings.allColoumns, Settings.COLUMN_I
D + "=? ", arg, null, null, null);
172.     return cur.getString(cur.getColumnIndex(Settings.COLUMN_CONTENT)) + passid;
173. }
174. public boolean SetPassword(String mypass) throws GeneralSecurityException {
175.     ContentValues values = new ContentValues(4);
176.     values.put(Settings.COLUMN_ID, passid);
177.     values.put(Settings.COLUMN_Discription, "on");
178.     values.put(Settings.COLUMN_OPTIONNAME, "Password");
179.     values.put(Settings.COLUMN_CONTENT, AESCrypt.encrypt(passid, mypass));
180.     String[] arg = {
181.         passid
182.     };
183.     try {
184.         UserDb.delete(Settings.TABLE_NAME, Settings.COLUMN_ID + "=?", arg);
185.         UserDb.insert(Settings.TABLE_NAME, null, values);
186.     } catch (SQLException e) {
187.         e.printStackTrace();
188.         return false;
189.     } finally {
190.         return true;
191.     }
192. }
193. public boolean SetPasswordoff() throws GeneralSecurityException {
194.     ContentValues values = new ContentValues(4);
195.     values.put(Settings.COLUMN_ID, passid);
196.     values.put(Settings.COLUMN_Discription, "off");
197.     values.put(Settings.COLUMN_OPTIONNAME, "Password");
198.     values.put(Settings.COLUMN_CONTENT, "");
199.     String[] arg = {
200.         passid
201.     };
202.     try {
203.         UserDb.delete(Settings.TABLE_NAME, Settings.COLUMN_ID + "=?", arg);
204.         UserDb.insert(Settings.TABLE_NAME, null, values);
205.     } catch (SQLException e) {
206.         e.printStackTrace();
207.         return false;
208.     } finally {
209.         return true;
210.     }
211. }
212. }

```

Local_DB_Main_Provider.java

```

1. package com.example.baltu.myapplication.Data_Provider;
2. import android.content.Context;

```

```

3. import android.database.Cursor;
4. import android.database.sqlite.SQLiteDatabase;
5. import android.database.sqlite.SQLiteOpenHelper;
6. import com.example.baltu.myapplication.DB_Constructor.Local_DB.Buss_Table;
7. import com.example.baltu.myapplication.DB_Constructor.Local_DB.Calender_Table;
8. import com.example.baltu.myapplication.DB_Constructor.Local_DB.MapLocations_Table;
9. import com.example.baltu.myapplication.DB_Constructor.Local_DB_Helper;
10. import com.example.baltu.myapplication.DataTypes.Bus_Data;
11. import com.example.baltu.myapplication.DataTypes.Calender_Data;
12. import com.example.baltu.myapplication.DataTypes.University_Locations;
13. import java.util.ArrayList;
14. import java.util.List;
15. public class Local_DB_Main_Provider {
16.     private Context mContext;
17.     private SQLiteDatabase LocalDB;
18.     SQLiteOpenHelper mLocalDBhelper;
19.     public Local_DB_Main_Provider(Context context) {
20.         this.mContext = context;
21.         mLocalDBhelper = new Local_DB_Helper(mContext);
22.         LocalDB = mLocalDBhelper.getWritableDatabase();
23.     }
24.     public void open() {
25.         LocalDB = mLocalDBhelper.getWritableDatabase();
26.     }
27.     public void close() {
28.         mLocalDBhelper.close();
29.     } //-----
-----
30.     public List < Bus_Data > GetAllBus(int x, int y) {
31.         List < Bus_Data > nelist = new ArrayList < > ();
32.         String[] arg = {
33.             Long.toString((new java.util.Date()).getTime())
34.         };
35.         Cursor cur;
36.         cur = LocalDB.query(Buss_Table.TABLE_NAME, Buss_Table.allColoumns, " To_from=? AND We
ekday=?", new String[] {
37.             String.valueOf(x), Integer.toString(y)
38.         }, null, null, Buss_Table.COLUMN_TIME);
39.         while (cur.moveToNext()) {
40.             Bus_Data ttc = new Bus_Data();
41.             ttc.setID(cur.getString(cur.getColumnIndex(Buss_Table.COLUMN_ID)));
42.             ttc.setLine_Number(cur.getInt(cur.getColumnIndex(Buss_Table.COLUMN_LINE_NUMBER)));
43.             ttc.setTime(cur.getString(cur.getColumnIndex(Buss_Table.COLUMN_TIME)));
44.             ttc.setTo_from(cur.getInt(cur.getColumnIndex(Buss_Table.COLUMN_TO_FROM)));
45.             ttc.setWeekday(cur.getInt(cur.getColumnIndex(Buss_Table.COLUMN_WEEKDAY)));
46.             nelist.add(ttc);
47.         }
48.         return nelist;
49.     }
50.     public List < Bus_Data > GetllineBus(int x, int y, int z) {
51.         List < Bus_Data > nelist = new ArrayList < > ();
52.         String[] arg = {
53.             Long.toString((new java.util.Date()).getTime())
54.         };
55.         Cursor cur;
56.         cur = LocalDB.query(Buss_Table.TABLE_NAME, Buss_Table.allColoumns, "Line_Number=? AND
To_from=? AND Weekday=?", new String[] {
57.             String.valueOf(x), Integer.toString(y), Integer.toString(z)
58.         }, null, null, Buss_Table.COLUMN_TIME);
59.         while (cur.moveToNext()) {

```



```

60.         Bus_Data ttc = new Bus_Data();
61.         ttc.setID(cur.getString(cur.getColumnIndex(Buss_Table.COLUMN_ID)));
62.         ttc.setLine_Number(cur.getInt(cur.getColumnIndex(Buss_Table.COLUMN_LINE_NUMBER)));

63.         ttc.setTime(cur.getString(cur.getColumnIndex(Buss_Table.COLUMN_TIME)));
64.         ttc.setTo_from(cur.getInt(cur.getColumnIndex(Buss_Table.COLUMN_TO_FROM)));
65.         ttc.setWeekday(cur.getInt(cur.getColumnIndex(Buss_Table.COLUMN_WEEKDAY)));
66.         nelist.add(ttc);
67.     }
68.     return nelist;
69. }
70. public List < Calender_Data > GetCalendarItems() {
71.     List < Calender_Data > nelist = new ArrayList < > ();
72.     Cursor cur;
73.     cur = LocalDB.query(Calender_Table.TABLE_NAME, Calender_Table.allColoumns, null, null,
null, null, Calender_Table.COLUMN_SDATE);
74.     while (cur.moveToNext()) {
75.         Calender_Data ttc = new Calender_Data();
76.         ttc.setID(cur.getString(cur.getColumnIndex(Calender_Table.COLUMN_ID)));
77.         ttc.setEvent_Name(cur.getString(cur.getColumnIndex(Calender_Table.COLUMN_Event_Nam
e)));
78.         ttc.setStartDate(cur.getLong(cur.getColumnIndex(Calender_Table.COLUMN_SDATE)));
79.         ttc.setEndDate(cur.getLong(cur.getColumnIndex(Calender_Table.COLUMN_EDATE)));
80.         nelist.add(ttc);
81.     }
82.     return nelist;
83. }
84. public List < University_Locations > GetMapLocations() {
85.     List < University_Locations > Locationslist = new ArrayList < > ();
86.     Cursor cur;
87.     cur = LocalDB.query(MapLocations_Table.TABLE_NAME, MapLocations_Table.allColoumns, nul
l, null, null, null, MapLocations_Table.COLUMN_LONGNAME);
88.     while (cur.moveToNext()) {
89.         University_Locations location = new University_Locations();
90.         location.setID(cur.getString(cur.getColumnIndex(MapLocations_Table.COLUMN_ID)));
91.         location.setShortName(cur.getString(cur.getColumnIndex(MapLocations_Table.COLUMN_S
HORTNAME)));
92.         location.setLongName(cur.getString(cur.getColumnIndex(MapLocations_Table.COLUMN_LO
NGNAME)));
93.         location.setInformation(cur.getString(cur.getColumnIndex(MapLocations_Table.COLUMN
_INFO)));
94.         location.setLatitude(cur.getDouble(cur.getColumnIndex(MapLocations_Table.COLUMN_LA
TITUDE)));
95.         location.setLongitude(cur.getDouble(cur.getColumnIndex(MapLocations_Table.COLUMN_L
ONGITUDE)));
96.         location.setImage_Url(cur.getString(cur.getColumnIndex(MapLocations_Table.COLUMN_I
MAGE_URL)));
97.         Locationslist.add(location);
98.     }
99.     return Locationslist;
100. }
101. }

```

BusItemsAdapter.java

```

1. package com.example.baltu.myapplication.Data_Provider.Adapters;
2. import android.content.Intent;
3. import android.support.v7.widget.RecyclerView;
4. import android.view.LayoutInflater;
5. import android.view.View;
6. import android.view.ViewGroup;
7. import android.widget.TextView;

```

```

8. import com.example.baltu.myapplication.DataTypes.Bus_Data;
9. import com.example.baltu.myapplication.R;
10. import com.example.baltu.myapplication.Map.University_Map;
11. import java.text.DateFormat;
12. import java.text.ParseException;
13. import java.text.SimpleDateFormat;
14. import java.util.Date;
15. import java.util.List;
16. public class BusItemsAdapter extends RecyclerView.Adapter < BusItemsAdapter.ViewHolder > { //
    The items to display in your RecyclerView
17.     private List < Bus_Data > items; // Provide a suitable constructor (depends on the kind of
        dataset)
18.     public BusItemsAdapter(List < Bus_Data > items) {
19.         this.items = items;
20.     } // Return the size of your dataset (invoked by the layout manager)
21.     @
22.     Override public int getItemCount() {
23.         return this.items.size();
24.     }@
25.     Override public int getItemViewType(int position) {
26.         return items.get(position).getLine_Number();
27.     }@
28.     Override public BusItemsAdapter.ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
29.         LayoutInflater inflater = LayoutInflater.from(viewGroup.getContext());
30.         final ViewHolder viewHolder;
31.         View v1;
32.         switch (viewType) {
33.             case 1:
34.                 v1 = inflater.inflate(R.layout.bus1_layout, viewGroup, false);
35.                 viewHolder = new ViewHolder(v1);
36.                 break;
37.             case 2:
38.                 v1 = inflater.inflate(R.layout.bus2_layout, viewGroup, false);
39.                 viewHolder = new ViewHolder(v1);
40.                 break;
41.             case 3:
42.                 v1 = inflater.inflate(R.layout.bus3_layout, viewGroup, false);
43.                 viewHolder = new ViewHolder(v1);
44.                 break;
45.             case 4:
46.                 v1 = inflater.inflate(R.layout.bus4_layout, viewGroup, false);
47.                 viewHolder = new ViewHolder(v1);
48.                 break;
49.             case 5:
50.                 v1 = inflater.inflate(R.layout.bus5_layout, viewGroup, false);
51.                 viewHolder = new ViewHolder(v1);
52.                 break;
53.             default:
54.                 v1 = inflater.inflate(R.layout.bus1_layout, viewGroup, false);
55.                 viewHolder = new ViewHolder(v1);
56.                 break;
57.         }
58.         return viewHolder;
59.     }@
60.     Override public void onBindViewHolder(final BusItemsAdapter.ViewHolder viewHolder, final int position) {
61.         final Bus_Data item = items.get(position);
62.         DateFormat df = new SimpleDateFormat("hh:mm aaa");
63.         DateFormat tlf = new SimpleDateFormat("HH:mm");
64.         Date now = new Date();

```

```

65.         String day_of_week = (new SimpleDateFormat("E")).format(now);
66.         String time = tlf.format(now);
67.         Date bustime = new Date();
68.         try {
69.             bustime = tlf.parse(item.getTime());
70.             now = tlf.parse(time);
71.         } catch (ParseException e) {
72.             e.printStackTrace();
73.         }
74.         viewHolder.Timet.setText(df.format(bustime));
75.         Date left = new Date(0);
76.         if (day_of_week.equals("Sun") || day_of_week.equals("Mon") || day_of_week.equals("Tue")
77. ) || day_of_week.equals("Wed") || day_of_week.equals("Thu")) {
78.             if (bustime.after(now)) {
79.                 left = new Date(0, 0, 0, bustime.getHours() - now.getHours(), bustime.getMinut
80. es() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
81.             } else {
82.                 left = new Date(0, 0, 0, 23 - (now.getHours() - bustime.getHours()), 60 - (now
83. .getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
84.             }
85.         } else if (day_of_week.equals("Fri")) {
86.             if (bustime.after(now)) {
87.                 left = new Date(0, 0, 0, bustime.getHours() - now.getHours(), bustime.getMinut
88. es() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
89.             } else {
90.                 left = new Date(0, 0, 2, 23 - (now.getHours() - bustime.getHours()), 60 - (now
91. .getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
92.             }
93.         } else if (day_of_week.equals("Sat")) {
94.             if (bustime.after(now)) {
95.                 left = new Date(0, 0, 2, bustime.getHours() - now.getHours(), bustime.getMinut
96. es() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
97.             } else {
98.                 left = new Date(0, 0, 1, 23 - (now.getHours() - bustime.getHours()), 60 - (now
99. .getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
100.             }
101.         }
102.         viewHolder.Timeleft.setText(Integer.toString(left.getDay()) + "d:" + Integer.toString(
103. left.getHours()) + "h:" + Integer.toString(left.getMinutes()) + "m");
104.         viewHolder.mView.setOnClickListener(new View.OnClickListener() {@
105.             Override public void onClick(View v) {
106.                 Intent intf = new Intent(viewHolder.mView.getContext(), University_Map.class);
107.
108.                 intf.putExtra("type", 1);
109.                 intf.putExtra("line", items.get(position).getLine_Number());
110.                 viewHolder.mView.getContext().startActivity(intf);
111.             }
112.         });
113.     }
114.     public static class ViewHolder extends RecyclerView.ViewHolder {
115.         public TextView Timet;
116.         public TextView Timeleft;
117.         public View mView;
118.         public Long Timer;
119.         public ViewHolder(View itemView) {
120.             super(itemView);
121.             Timet = (TextView) itemView.findViewById(R.id.bustime);
122.             Timeleft = (TextView) itemView.findViewById(R.id.timelefttime);
123.             mView = itemView;
124.         }
125.     }

```

117. }

BusItemsAdapterweekend.java

```
1. package com.example.baltu.myapplication.Data_Provider.Adapters;
2. import android.content.Intent;
3. import android.support.v7.widget.RecyclerView;
4. import android.view.LayoutInflater;
5. import android.view.View;
6. import android.view.ViewGroup;
7. import android.widget.TextView;
8. import com.example.baltu.myapplication.DataTypes.Bus_Data;
9. import com.example.baltu.myapplication.R;
10. import com.example.baltu.myapplication.Map.University_Map;
11. import java.text.DateFormat;
12. import java.text.ParseException;
13. import java.text.SimpleDateFormat;
14. import java.util.Date;
15. import java.util.List;
16. public class BusItemsAdapterweekend extends RecyclerView.Adapter < BusItemsAdapterweekend.View
    Holder > { // The items to display in your RecyclerView
17.     private List < Bus_Data > items; // Provide a suitable constructor (depends on the kind of
        dataset)
18.     public BusItemsAdapterweekend(List < Bus_Data > items) {
19.         this.items = items;
20.     } // Return the size of your dataset (invoked by the layout manager)
21.     @
22.     Override public int getItemCount() {
23.         return this.items.size();
24.     }@
25.     Override public int getItemViewType(int position) {
26.         return items.get(position).getLine_Number();
27.     }@
28.     Override public BusItemsAdapterweekend.ViewHolder onCreateViewHolder(ViewGroup viewGroup,
        int viewType) {
29.         LayoutInflater inflater = LayoutInflater.from(viewGroup.getContext());
30.         final ViewHolder viewHolder;
31.         View v1;
32.         switch (viewType) {
33.             case 1:
34.                 v1 = inflater.inflate(R.layout.bus1_layout, viewGroup, false);
35.                 viewHolder = new ViewHolder(v1);
36.                 break;
37.             case 2:
38.                 v1 = inflater.inflate(R.layout.bus2_layout, viewGroup, false);
39.                 viewHolder = new ViewHolder(v1);
40.                 break;
41.             case 3:
42.                 v1 = inflater.inflate(R.layout.bus3_layout, viewGroup, false);
43.                 viewHolder = new ViewHolder(v1);
44.                 break;
45.             case 4:
46.                 v1 = inflater.inflate(R.layout.bus4_layout, viewGroup, false);
47.                 viewHolder = new ViewHolder(v1);
48.                 break;
49.             case 5:
50.                 v1 = inflater.inflate(R.layout.bus5_layout, viewGroup, false);
51.                 viewHolder = new ViewHolder(v1);
52.                 break;
53.             default:
54.                 v1 = inflater.inflate(R.layout.bus1_layout, viewGroup, false);
55.                 viewHolder = new ViewHolder(v1);
```

```

56.         break;
57.     }
58.     return viewHolder;
59. }@
60. Override public void onBindViewHolder(final BusItemsAdapterweekend.ViewHolder viewHolder,
    final int position) {
61.     final Bus_Data item = items.get(position);
62.     DateFormat df = new SimpleDateFormat("hh:mm aaa");
63.     DateFormat tlf = new SimpleDateFormat("HH:mm");
64.     Date now = new Date();
65.     DateFormat day_of_week_fomat = new SimpleDateFormat("E");
66.     String day_of_week = day_of_week_fomat.format(now);
67.     String time = tlf.format(now);
68.     Date bustime = new Date();
69.     try {
70.         bustime = tlf.parse(item.getTime());
71.         now = tlf.parse(time);
72.     } catch (ParseException e) {
73.         e.printStackTrace();
74.     }
75.     viewHolder.Timet.setText(df.format(bustime));
76.     Date left = new Date(0);
77.     if (day_of_week.equals("Sat")) {
78.         if (bustime.after(now)) {
79.             left = new Date(0, 0, 0, bustime.getHours() - now.getHours(), bustime.getMinute
es() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
80.         } else {
81.             left = new Date(0, 0, 0, 23 - (now.getHours() - bustime.getHours()), 60 - (now
.getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
82.         }
83.     } else if (day_of_week.equals("Fri")) {
84.         if (bustime.after(now)) {
85.             left = new Date(0, 0, 1, bustime.getHours() - now.getHours(), bustime.getMinute
es() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
86.         } else {
87.             left = new Date(0, 0, 0, 23 - (now.getHours() - bustime.getHours()), 60 - (now
.getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
88.         }
89.     } else if (day_of_week.equals("Sun")) {
90.         if (bustime.after(now)) {
91.             left = new Date(0, 0, 6, bustime.getHours() - now.getHours(), bustime.getMinute
es() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
92.         } else {
93.             left = new Date(0, 0, 5, 23 - (now.getHours() - bustime.getHours()), 60 - (now
.getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
94.         }
95.     } else if (day_of_week.equals("Mon")) {
96.         if (bustime.after(now)) {
97.             left = new Date(0, 0, 5, bustime.getHours() - now.getHours(), bustime.getMinute
es() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
98.         } else {
99.             left = new Date(0, 0, 4, 23 - (now.getHours() - bustime.getHours()), 60 - (now
.getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
100.        }
101.    } else if (day_of_week.equals("Tue")) {
102.        if (bustime.after(now)) {
103.            left = new Date(0, 0, 4, bustime.getHours() - now.getHours(), bustime.g
etMinutes() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
104.        } else {

```

```

105.         left = new Date(0, 0, 3, 23 - (now.getHours() - bustime.getHours()), 60
- (now.getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
106.     }
107.     } else if (day_of_week.equals("Wed")) {
108.         if (bustime.after(now)) {
109.             left = new Date(0, 0, 3, bustime.getHours() - now.getHours(), bustime.g
etMinutes() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
110.         } else {
111.             left = new Date(0, 0, 2, 23 - (now.getHours() - bustime.getHours()), 60
- (now.getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
112.         }
113.     } else if (day_of_week.equals("Thu")) {
114.         if (bustime.after(now)) {
115.             left = new Date(0, 0, 2, bustime.getHours() - now.getHours(), bustime.g
etMinutes() - now.getMinutes(), bustime.getSeconds() - now.getSeconds());
116.         } else {
117.             left = new Date(0, 0, 1, 23 - (now.getHours() - bustime.getHours()), 60
- (now.getMinutes() - bustime.getMinutes()), 59 - (now.getSeconds() - bustime.getSeconds()));
118.         }
119.     }
120.     viewHolder.Timeleft.setText(Integer.toString(left.getDay()) + "d:" + Integer.to
String(left.getHours()) + "h:" + Integer.toString(left.getMinutes()) + "m");
121.     viewHolder.mView.setOnClickListener(new View.OnClickListener() {@
122.         Override public void onClick(View v) {
123.             Intent intf = new Intent(viewHolder.mView.getContext(), University_Map.
class);
124.             intf.putExtra("type", 1);
125.             intf.putExtra("line", items.get(position).getLine_Number());
126.             viewHolder.mView.getContext().startActivity(intf);
127.         }
128.     });
129. }
130. public static class ViewHolder extends RecyclerView.ViewHolder {
131.     public TextView Timet;
132.     public TextView Timeleft;
133.     public View mView;
134.     public Long Timer;
135.     public ViewHolder(View itemView) {
136.         super(itemView);
137.         Timet = (TextView) itemView.findViewById(R.id.bustime);
138.         Timeleft = (TextView) itemView.findViewById(R.id.timelefttime);
139.         mView = itemView;
140.     }
141. }
142. }

```

CalendarItemsAdapter.java

```

1. package com.example.baltu.myapplication.Data_Provider.Adapters;
2. import android.content.Context;
3. import android.support.v7.widget.RecyclerView;
4. import android.view.LayoutInflater;
5. import android.view.View;
6. import android.view.ViewGroup;
7. import android.widget.TextView;
8. import com.example.baltu.myapplication.DataTypes.Calender_Data;
9. import com.example.baltu.myapplication.R;
10. import java.text.DateFormat;
11. import java.text.SimpleDateFormat;

```

```

12. import java.util.Date;
13. import java.util.List;
14. public class CalendarItemsAdapter extends RecyclerView.Adapter < CalendarItemsAdapter.ViewHolder > {
15.     private List < Calender_Data > mItems;
16.     private Context mContext;
17.     public CalendarItemsAdapter(Context context, List < Calender_Data > items) {
18.         this.mContext = context;
19.         this.mItems = items;
20.     }
21.     Override public CalendarItemsAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
22.         LayoutInflater inflater = LayoutInflater.from(mContext);
23.         View itemView = inflater.inflate(R.layout.calendar_item_layout, parent, false);
24.         ViewHolder viewHolder = new ViewHolder(itemView);
25.         return viewHolder;
26.     }
27.     Override public void onBindViewHolder(final CalendarItemsAdapter.ViewHolder holder, int position) {
28.         final Calender_Data item = mItems.get(position);
29.         DateFormat df = new SimpleDateFormat("dd / MM / yyyy");
30.         if (item.getEvent_Name().contains("$")) { //Months will contain $
31.             holder.TITLE.setText(item.getEvent_Name().substring(2));
32.             holder.TITLE.setTextSize(24);
33.             holder.EDateView.setVisibility(View.GONE);
34.             holder.SDate_View.setVisibility(View.GONE);
35.             holder.TITLE.setTextColor(mContext.getResources().getColor(android.R.color.white));
36.             holder.mView.setBackgroundColor(mContext.getResources().getColor(android.R.color.holo_blue_dark));
37.         } else {
38.             if (item.getEvent_Name().contains("##")) { //Years will contain ##
39.                 holder.TITLE.setText(item.getEvent_Name().substring(2));
40.                 holder.TITLE.setTextSize(24);
41.                 holder.TITLE.setTextColor(mContext.getResources().getColor(android.R.color.white));
42.                 holder.EDateView.setVisibility(View.GONE);
43.                 holder.SDate_View.setVisibility(View.GONE);
44.                 holder.mView.setBackgroundColor(mContext.getResources().getColor(android.R.color.holo_green_dark));
45.             } else {
46.                 holder.TITLE.setText(item.getEvent_Name());
47.                 if (item.getStartDate().equals(item.getEndDate())) {
48.                     holder.SDateText.setText(df.format(new Date(item.getStartDate())));
49.                     holder.EDateView.setVisibility(View.GONE);
50.                 } else {
51.                     holder.SDateText.setText(df.format(new Date(item.getStartDate())));
52.                     holder.SDate.setText("Start Date:");
53.                     holder.EDateText.setText(df.format(new Date(item.getEndDate())));
54.                 }
55.             }
56.         }
57.     }
58.     Override public int getItemCount() {
59.         return mItems.size();
60.     }
61.     public static class ViewHolder extends RecyclerView.ViewHolder {
62.         public TextView TITLE;
63.         public View SDate_View;
64.         public TextView SDate;
65.         public TextView SDateText;

```



```

66.     public TextView EDateText;
67.     public View EDateView;
68.     public View mView;
69.     public ViewHolder(View itemView) {
70.         super(itemView);
71.         TITLE = (TextView) itemView.findViewById(R.id.calendar_title);
72.         SDateText = (TextView) itemView.findViewById(R.id.calendar_date_text);
73.         SDate = (TextView) itemView.findViewById(R.id.calendar_date);
74.         EDateText = (TextView) itemView.findViewById(R.id.calendar_Enddate_text);
75.         EDateView = itemView.findViewById(R.id.calendar_end_view);
76.         SDate_View = itemView.findViewById(R.id.Calendar_StartDate_View);
77.         mView = itemView;
78.     }
79. }
80. }

```

CoursesItemAdapter.java

```

1. package com.example.baltu.myapplication.Data_Provider.Adapters;
2. import android.content.Context;
3. import android.content.Intent;
4. import android.support.v7.widget.RecyclerView;
5. import android.view.LayoutInflater;
6. import android.view.View;
7. import android.view.ViewGroup;
8. import android.widget.TextView;
9. import com.example.baltu.myapplication.DataTypes.TimeTable_Classes;
10. import com.example.baltu.myapplication.R;
11. import com.example.baltu.myapplication.TimeTables.Courses_Viewer_Editor_Activity;
12. import java.text.DateFormat;
13. import java.text.SimpleDateFormat;
14. import java.util.List;
15. public class CoursesItemAdapter extends RecyclerView.Adapter < CoursesItemAdapter.ViewHolder >
    {
16.     private static DateFormat timeformat = new SimpleDateFormat("HH:mm");
17.     private List < TimeTable_Classes > mItems;
18.     private Context mContext;
19.     public CoursesItemAdapter(Context context, List < TimeTable_Classes > items) {
20.         this.mContext = context;
21.         this.mItems = items;
22.     }
23.     @Override public CoursesItemAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
24.         LayoutInflater inflater = LayoutInflater.from(mContext);
25.         View itemView = inflater.inflate(R.layout.lecture_layout, parent, false);
26.         ViewHolder viewHolder = new ViewHolder(itemView);
27.         return viewHolder;
28.     }
29.     @Override public void onBindViewHolder(final CoursesItemAdapter.ViewHolder holder, int position) {
30.         final TimeTable_Classes item = mItems.get(position);
31.         holder.cName.setText(item.getCourse());
32.         holder.starttime.setText(item.getStarting_time());
33.         holder.endtime.setText(item.getFinishing_time());
34.         holder.courseid.setText(item.getID());
35.         holder.mView.setOnClickListener(new View.OnClickListener() {
36.             @Override public void onClick(View v) {
37.                 Intent ce = new Intent(mContext, Courses_Viewer_Editor_Activity.class);
38.                 ce.putExtra("name", item.getCourse());
39.                 ce.putExtra("day", item.getDay_of_week());
40.                 ce.putExtra("start", item.getStarting_time());
41.                 ce.putExtra("end", item.getFinishing_time());

```

```

42.         ce.putExtra("note", item.getNotes());
43.         ce.putExtra("id", item.getID());
44.         mContext.startActivity(ce);
45.     }
46. });
47. }@
48. Override public int getItemCount() {
49.     return mItems.size();
50. }
51. public static class ViewHolder extends RecyclerView.ViewHolder {
52.     public TextView cName;
53.     public TextView starttime;
54.     public TextView endtime;
55.     public TextView courseid;
56.     public View mView;
57.     public ViewHolder(View itemView) {
58.         super(itemView);
59.         cName = (TextView) itemView.findViewById(R.id.Task_course_name);
60.         starttime = (TextView) itemView.findViewById(R.id.Startt);
61.         endtime = (TextView) itemView.findViewById(R.id.Endt);
62.         courseid = (TextView) itemView.findViewById(R.id.course_id);
63.         mView = itemView;
64.     }
65. }
66. }

```

ExamsItemsAdapter.java

```

1. package com.example.baltu.myapplication.Data_Provider.Adapters;
2. import android.content.Context;
3. import android.content.Intent;
4. import android.support.v7.widget.RecyclerView;
5. import android.view.LayoutInflater;
6. import android.view.View;
7. import android.view.ViewGroup;
8. import android.widget.TextView;
9. import com.example.baltu.myapplication.DataTypes.Exam_class;
10. import com.example.baltu.myapplication.R;
11. import com.example.baltu.myapplication.TimeTables.Exams_Viewer_Editor_Activity;
12. import java.text.DateFormat;
13. import java.text.SimpleDateFormat;
14. import java.util.Date;
15. import java.util.List;
16. public class ExamsItemsAdapter extends RecyclerView.Adapter < ExamsItemsAdapter.ViewHolder > {
17.     private List < Exam_class > mItems;
18.     private Context mContext;
19.     int f;
20.     public ExamsItemsAdapter(Context context, List < Exam_class > items) {
21.         this.mContext = context;
22.         this.mItems = items;
23.     }@
24. Override public ExamsItemsAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
25.     LayoutInflater inflater = LayoutInflater.from(mContext);
26.     View itemView = inflater.inflate(R.layout.exam_layout, parent, false);
27.     ViewHolder viewHolder = new ViewHolder(itemView);
28.     return viewHolder;
29. }@
30. Override public void onBindViewHolder(final ExamsItemsAdapter.ViewHolder holder, int position) {
31.     final Exam_class item = mItems.get(position);

```

```

32.     holder.TName.setText(item.getCourse());
33.     holder.TDescription.setText(item.getDiscription());
34.     Date d = new Date(item.getExam_Date()); //String ddate=d.toString();
35.     DateFormat dtd = new SimpleDateFormat("EEE dd MMM yyyy");
36.     DateFormat dtt = new SimpleDateFormat("hh:mm aaa");
37.     holder.DeadLineT.setText(dtt.format(d));
38.     holder.DeadLineD.setText(dtd.format(d));
39.     holder.Taskid.setText(item.getID());
40.     holder.mView.setOnClickListener(new View.OnClickListener() {@
41.         Override public void onClick(View v) {
42.             Intent In = new Intent(mContext, Exams_Viewer_Editor_Activity.class);
43.             In.putExtra("id", item.getID());
44.             In.putExtra("name", item.getCourse());
45.             In.putExtra("desc", item.getDiscription());
46.             In.putExtra("date", item.getExam_Date());
47.             In.putExtra("notes", item.getNotes());
48.             mContext.startActivity(In);
49.         }
50.     });
51. }@
52. Override public int getItemCount() {
53.     return mItems.size();
54. }
55. public static class ViewHolder extends RecyclerView.ViewHolder {
56.     public TextView TName;
57.     public TextView TDescription;
58.     public TextView DeadLineT;
59.     public TextView DeadLineD;
60.     public TextView Taskid;
61.     public View mView;
62.     public ViewHolder(View itemView) {
63.         super(itemView);
64.         TName = (TextView) itemView.findViewById(R.id.Task_course_name);
65.         TDescription = (TextView) itemView.findViewById(R.id.Task_Description);
66.         DeadLineT = (TextView) itemView.findViewById(R.id.DeadlineTime);
67.         DeadLineD = (TextView) itemView.findViewById(R.id.DeadlineDate);
68.         Taskid = (TextView) itemView.findViewById(R.id.TaskID);
69.         mView = itemView;
70.     }
71. }
72. }

```

TasksItemsAdapter.java

```

1. package com.example.baltu.myapplication.Data_Provider.Adapters;
2. import android.content.Context;
3. import android.content.Intent;
4. import android.support.v7.widget.RecyclerView;
5. import android.view.LayoutInflater;
6. import android.view.View;
7. import android.view.ViewGroup;
8. import android.widget.TextView;
9. import com.example.baltu.myapplication.DataTypes.Tasks_Data;
10. import com.example.baltu.myapplication.R;
11. import com.example.baltu.myapplication.TimeTables.Tasks_Viewer_Editor_Activity;
12. import java.text.DateFormat;
13. import java.text.SimpleDateFormat;
14. import java.util.Date;
15. import java.util.List;

```

```

16. public class TasksItemsAdapter extends RecyclerView.Adapter < TasksItemsAdapter.ViewHolder > {
17.     private List < Tasks_Data > mItems;
18.     private Context mContext;
19.     public TasksItemsAdapter(Context context, List < Tasks_Data > items) {
20.         this.mContext = context;
21.         this.mItems = items;
22.     }
23.     Override public TasksItemsAdapter.ViewHolder onCreateViewHolder(ViewGroup parent, int view
Type) {
24.         LayoutInflater inflater = LayoutInflater.from(mContext);
25.         View itemView = inflater.inflate(R.layout.task_layout, parent, false);
26.         ViewHolder viewHolder = new ViewHolder(itemView);
27.         return viewHolder;
28.     }
29.     Override public void onBindViewHolder(final TasksItemsAdapter.ViewHolder holder, int posit
ion) {
30.         final Tasks_Data item = mItems.get(position);
31.         holder.TName.setText(item.getCourse());
32.         holder.TDescription.setText(item.getDiscription());
33.         Date d = new Date(item.getDeadline_Date()); //String ddate=d.toString();
34.         DateFormat dtd = new SimpleDateFormat("EEE dd MMM yyyy");
35.         DateFormat dtt = new SimpleDateFormat("hh:mm aaa");
36.         holder.DeadLineT.setText(dtt.format(d));
37.         holder.DeadLineD.setText(dtd.format(d));
38.         holder.Taskid.setText(item.getID());
39.         holder.mView.setOnClickListener(new View.OnClickListener() {@
40.             Override public void onClick(View v) {
41.                 Intent In = new Intent(mContext, Tasks_Viewer_Editor_Activity.class);
42.                 In.putExtra("id", item.getID());
43.                 In.putExtra("name", item.getCourse());
44.                 In.putExtra("desc", item.getDiscription());
45.                 In.putExtra("date", item.getDeadline_Date());
46.                 In.putExtra("notes", item.getNotes());
47.                 mContext.startActivity(In);
48.             }
49.         });
50.     }
51.     Override public int getItemCount() {
52.         return mItems.size();
53.     }
54.     public static class ViewHolder extends RecyclerView.ViewHolder {
55.         public TextView TName;
56.         public TextView TDescription;
57.         public TextView DeadLineT;
58.         public TextView DeadLineD;
59.         public TextView Taskid;
60.         public View mView;
61.         public ViewHolder(View itemView) {
62.             super(itemView);
63.             TName = (TextView) itemView.findViewById(R.id.Task_course_name);
64.             TDescription = (TextView) itemView.findViewById(R.id.Task_Description);
65.             DeadLineT = (TextView) itemView.findViewById(R.id.DeadlineTime);
66.             DeadLineD = (TextView) itemView.findViewById(R.id.DeadlineDate);
67.             Taskid = (TextView) itemView.findViewById(R.id.TaskID);
68.             mView = itemView;
69.         }
70.     }
71. }

```