软件设计综合实验

实现项目：课程表 Block
小组成员及联系方式：
14331130 李伽泽 2439166979@qq.com
14331249 佟欣玥 1041783598@qq.com
14331125 蓝艺元 435558621@qq.com
14331337 杨真如 896793325@qq.com

软件设计综合实验

软件设计文档

实验环境

工具：Android Studio

SDK：Android 4.4（KitKat） API Level 19

JDK：Java 1.8

应用功能

实现一个课程表，具体功能如下：

1. 程序的启动界面为欢迎界面，3s 后进入课程表界面。
2. 点击最上方周数，可调整当前周数。
3. 点击左边一列的任意上课节数，即可添加以这一节开始的课程。
4. 添加的内容可以有：课程名称、老师、Email 或 ftp（课程资料）、星期、开始课节、结束课节以及教室。
5. 点击添加完成的课程，即可查看课程名称、课程信息、课程重要程度、课程作业以及作业的截止时间。
6. 以上内容可以编辑修改，也可以删除课程。
7. 设置作业截止时间后，在作业截止时间前一天会弹出通知并震动，以提醒作业提交即将截止。
8. 可以在桌面上添加课程表的 widget，显示作业截止的详细信息，点击 widget 右上角方向键可调整查看每一门课程作业。

技术、方法和工具

1. 界面框架设计

用户友好，控件布局合理，美观。

首先说明修改 Android app 图标。

res/drawable 放置 logo.jpg，修改 AndroidManifest.xml：

```
android:icon="@drawable/logo"
```

其次，加入了欢迎界面。

```
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    setContentView(R.layout.welcome);

    Handler handler = new Handler();
    //当计时结束，跳转至主界面
    handler.postDelayed(() → {
            Intent intent = new Intent(WelcomeActivity.this, MainActivity.class);
            startActivity(intent);
            WelcomeActivity.this.finish();
    }, 3000);
}
```

最后，之前实验中没有使用过的控件 Picker（DatePicker、TimePicker）在此处详细说明。

DatePicker 是一个日期选择控件，它继承自 FrameLayout 类，用来实现的主要功能是使用户可以方便选择日期。如果要捕获用户修改 DataPicker 控件中的数据改变事件，需要为 DatePicker 添加 OnDateChangedListener 监听器。

TimePicker 是一个时间选择控件，也继承自 FrameLayout 类。时间选择控件向用户显示一天中的时间（可以为 24 小时，也可以为 AM/PM 制），并允许用户进行选择。如果要捕获用户修改时间数据的事件，便需要为 TimePicker 添加 OnTimeChangedListener 监听器。

界面布局文件中的 DatePicker 和 TimePicker 控件：

```
<DatePicker
    android:id="@+id/datepicker"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:calendarViewShown="false" />

<TimePicker
    android:id="@+id/timepicker"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

为 DatePicker 添加的 OnDateChangedListener 监听器：

```
public void onDateChanged(DatePicker view, int year, int monthOfYear,
                          int dayOfMonth) {
    // 获得日历实例
    Calendar calendar = Calendar.getInstance();

    calendar.set(datePicker.getYear(), datePicker.getMonth(),
            datePicker.getDayOfMonth(), timePicker.getCurrentHour(),
            timePicker.getCurrentMinute());
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy年MM月dd日 HH:mm");

    dateTime = sdf.format(calendar.getTime());
    ad.setTitle(dateTime);
}
```

为 TimePicker 添加的 OnTimeChangedListener 监听器：

```
public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
    onDateChanged(null, 0, 0, 0);
}
```

2. 事件处理的实现

点击事件、activity 跳转。

在主界面中，点击课程节数会跳转到添加课程界面。点击课程会跳转到显示课程详情界面。

在添加课程界面，可以设置要添加课程的星期数、开始课节与结束课节。可添加的内容有课程名称、老师、Email 或者 ftp（课程资料）、教室。点击提交按钮，又会跳转回主界面。

在显示课程详情界面点击开启定时功能按钮，以开启提醒作业截止功能；若已开启定时功能，点击关闭定时功能按钮，以关闭提醒作业截止功能；点击删除按钮，删除此课程并跳转回主界面；点击确定按钮，跳转回主界面；点击编辑按钮，跳转到编辑课程详情界面；点击删除按钮，删除此课程并跳转回主界面。

在编辑课程详情界面，点击作业截止时间后的输入框，弹出日历，以选择作业截止时间；点击设置截止时间输入框后面的清空时间按钮，设置截止时间输入框内容被清空；点击提交按钮，课程详情即被修改，并跳转回程序主界面。

点击弹出的提醒作业即将截止的通知，开启程序至主界面。

3. SQLite 数据存储方式

课程详情等此程序需要储存的所有信息都是由 SQLite 数据存储方式储存的。

创建数据库 Course：

包含课程信息以及课程截止时间。

```
public void onCreate(SQLiteDatabase sqLiteDatabase) {
    String CREATE_TABLE = "CREATE TABLE if not exists " + TABLE_NAME
            + " (name TEXT, teacher TEXT, email_ftp TEXT, week int, start int, end int, address TEXT, ddl TEXT, ddl_time TEXT, lamp int)";
    sqLiteDatabase.execSQL(CREATE_TABLE);
}
```

重写查询方法，通过获取周数以及开始课节返回该课程的数据。

```
public Course query(int week, int start) {
    SQLiteDatabase db = getReadableDatabase();
    Course course = new Course();
    Cursor cursor = db.rawQuery("select * from " + TABLE_NAME + " where week = " + week + "" + " and start = " + start + "", null);
    if (cursor.getCount() != 0) {
        while (cursor.moveToNext()) {
            course.setName(cursor.getString(cursor.getColumnIndex("name")));
            course.setTeacher(cursor.getString(cursor.getColumnIndex("teacher")));
            course.setAddress(cursor.getString(cursor.getColumnIndex("address")));
            course.setEmail_ftp(cursor.getString(cursor.getColumnIndex("email_ftp")));
            course.setDdl(cursor.getString(cursor.getColumnIndex("ddl")));
            course.setStart(cursor.getInt(cursor.getColumnIndex("start")));
            course.setEnd(cursor.getInt(cursor.getColumnIndex("end")));
            course.setWeek(cursor.getInt(cursor.getColumnIndex("week")));
            course.setDdl_time(cursor.getString(cursor.getColumnIndex("ddl_time")));
            course.setLamp(cursor.getInt(cursor.getColumnIndex("lamp")));
        }
    }
    db.close();
    return course;
}
```

重写删除，通过周数以及开始课节可以确定删除的课程。

```
public void delete(int week, int start) {
    SQLiteDatabase db = getWritableDatabase();
    String whereClause = "week = ? and start = ?";
    String[] whereArgs = {week + "", start + ""};
    db.delete(TABLE_NAME, whereClause, whereArgs);
    db.close();
}
```

更新，利用 update 更新数据库中的数据。

```
public void update(Course course) {
    SQLiteDatabase db = getWritableDatabase();
    ContentValues cv = new ContentValues();
    cv.put("name", course.getName());
    cv.put("teacher", course.getTeacher());
    cv.put("email_ftp", course.getEmail_ftp());
    cv.put("week", course.getWeek());
    cv.put("start", course.getStart());
    cv.put("end", course.getEnd());
    cv.put("address", course.getAddress());
    cv.put("ddl", course.getDdl());
    cv.put("ddl_time", course.getDdl_time());
    cv.put("lamp", course.getLamp());
    String whereClause = "week = ? and start = ?";
    String[] whereArgs = {course.getWeek() + "", course.getStart() + ""};
    db.update(TABLE_NAME, cv, whereClause, whereArgs);
    db.close();
}
```

查重，由于不能在同一时间内添加两门课程，如果数据库中有冲突数据（利用开始时间与结束时间判断），需要查重。

```
public Boolean query_same(int week, int start, int end) {
    Boolean has = false;
    SQLiteDatabase db = getReadableDatabase();
    String[] column = {"name"};
    String select = "week = ? and ((start >= ? and start <= ?) or (end >= ? and end <= ?))";
    String[] args = {week + "", start + "", end + "", start + "", end + ""};
    Cursor cursor = db.query(TABLE_NAME, column, select, args, null, null, null);
    if (cursor.getCount() != 0) {
        has = true;
    }
    db.close();
    return has;
}
```

返回所有数据。

```
public Cursor query_all() {
    SQLiteDatabase db = getReadableDatabase();
    Cursor cursor = db.rawQuery("select * from Course", null);
    return cursor;
}
```

4. Notification、Widget 显示方式

当出现操作错误（例如添加课程的位置已添加了课程）时，会弹出 Toast 消息提醒。

下面以添加课程时，判断添加的内容是否合法为例：

```
if (week.getText().toString().equals("") || start.getText().toString().equals("") || end.getText().toString().equals("")) {
    Toast.makeText(AddActivity.this, "请输入周数开始课节和结束课节", Toast.LENGTH_SHORT).show();
} else {
    _week = Integer.parseInt(week.getText().toString());
    _start = Integer.parseInt(start.getText().toString());
    _end = Integer.parseInt(end.getText().toString());
    if (_end < _start) {
        Toast.makeText(AddActivity.this, "结束课节应该大于开始课节", Toast.LENGTH_SHORT).show();
    } else if (_week < 1 || _week > 7) {
        Toast.makeText(AddActivity.this, "周数应该在1-7之间", Toast.LENGTH_SHORT).show();
    } else if (_start < 1 || _start > 16 || _end > 16 || _end < 1) {
        Toast.makeText(AddActivity.this, "课节数应该在1-16之间", Toast.LENGTH_SHORT).show();
    } else if (dataBase.query_same(_week, _start, _end)) {
        Toast.makeText(AddActivity.this, "该时间段有重复的课程", Toast.LENGTH_SHORT).show();
```

在上面一段代码处理了输入的开始课节或结束课节为空、输入的结束课节小于开始课节、周数不在 1-7 之间以及重复添加课程这五种非法输入。通过弹出 Toast 的方式提醒用户，处理错误。

在桌面上添加课程表的 widget，显示作业截止的详细信息。

```
@Override
public void onReceive(Context context, Intent intent) {
    super.onReceive(context, intent);

    switch (intent.getAction()) {
        case PREV_ONCLICK:
            index = index - 1 < 0 ? 0 : index - 1;
            break;

        case NEXT_ONCLICK:
            index++;
            break;
    }

    update(context);
}
```

```
private void update(Context context) {
    AppWidgetManager manager = AppWidgetManager.getInstance(context);

    Intent clickInt = new Intent(context, MainActivity.class);
    Intent clickPrev = new Intent(PREV_ONCLICK);
    Intent clickNext = new Intent(NEXT_ONCLICK);

    PendingIntent pi = PendingIntent.getActivity(context, 0, clickInt, 0);
    PendingIntent piPrev = PendingIntent.getBroadcast(context, 1, clickPrev, PendingIntent.FLAG_UPDATE_CURRENT);
    PendingIntent piNext = PendingIntent.getBroadcast(context, 1, clickNext, PendingIntent.FLAG_UPDATE_CURRENT);

    RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.widget_demo);
    ComponentName cn = new ComponentName(context, WidgetDemo.class);

    views.setOnClickPendingIntent(R.id.content_layout, pi);
    views.setOnClickPendingIntent(R.id.prev, piPrev);
    views.setOnClickPendingIntent(R.id.next, piNext);

    String ddlContent = "Content";
    String ddlClass = "Class";
    String ddlTime = "Time";

    if (!getCourseDdlContent(index).equals(""))
    {
        ddlContent = getCourseDdlContent(index);
        ddlClass = getCourseDdlClass(index);
        ddlTime = getCourseDdlTime(index);
    }

    views.setTextViewText(R.id.ddl_content, ddlContent);
    views.setTextViewText(R.id.ddl_class, ddlClass);
    views.setTextViewText(R.id.ddl_time, ddlTime);

    if (manager != null) {
        manager.updateAppWidget(manager.getAppWidgetIds(cn), views);
    }
}
```

此部分需注意 widget 的更新。在 widget 的 onReceive 方法里进行更新，当编辑页面提交时和详情界面删除时发送一个广播，这个广播会使 widget 的 onReceive 接收被更新页面，更新页面从数据库里进行数据获取。

5. Broadcast

设置作业截止时间后，在作业截止时间前的每一天都会发送广播，前一天弹出通知，以提醒作业提交即将截止。

```
public class DayReceiver  extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals("DAY")) {
            Bundle bundle = intent.getExtras();
            String day = bundle.getString("day");
            String course = bundle.getString("course");
            String content = bundle.getString("content");
            String detail_content;
            if (Integer.parseInt(day) == 0) {
                MainActivity.vibrator.vibrate(1000);
                detail_content = course + "\n" + "内容: " + content + "\n还有1天";
                int largeIcon = R.mipmap.ic_launcher;
                Notification.Builder builder = new Notification.Builder(context);
                builder.setContentTitle("DDL")
                        .setContentText(detail_content)
                        .setTicker("又来了一个DDL")
                        .setLargeIcon(BitmapFactory.decodeResource(context.getResources(), largeIcon))
                        .setSmallIcon(largeIcon)
                        .setAutoCancel(true);
                NotificationManager manager = (NotificationManager)context.getSystemService(Context.NOTIFICATION_SERVICE);

                Intent back = new Intent(context, MainActivity.class);
                PendingIntent goBack = PendingIntent.getActivity(context, 0, back, 0);
                builder.setContentIntent(goBack);

                Notification notify = builder.build();
                manager.notify(0, notify);
            }
        }
    }
}
```

在此部分为了测试方便设为每 10s 发送一次广播，这样在提前一天的时间内，每 10s 就会提醒并震动一次。

6. Service

设置定时服务，以实现在作业截止时间前一天会弹出通知。

通过 Binder 来保持 Activity 和 Service 的通信。

```
private final IBinder binder = new MyBinder();
public class MyBinder extends Binder {
    TimerService getService() { return TimerService.this; }
}
```

重写 onStartCommand、onDestroy。

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    return Service.START_STICKY;
}
@Override
public void onDestroy() { super.onDestroy(); }
```

定时器：

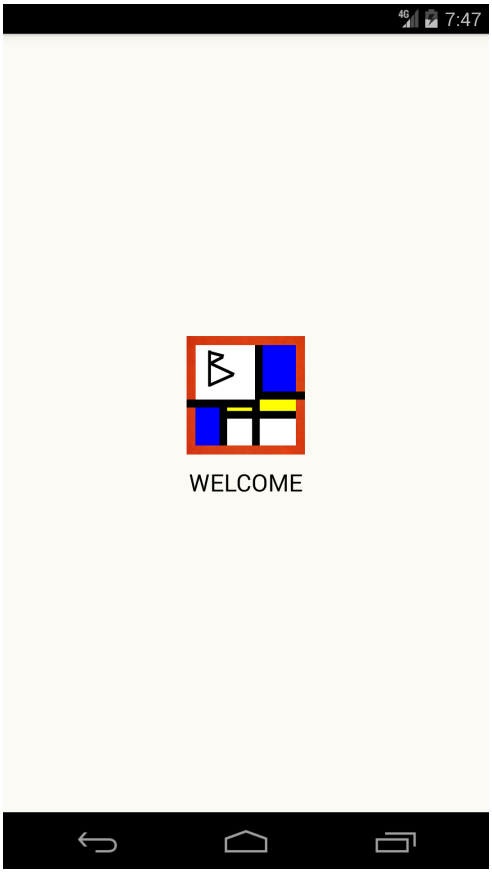
```
private Timer _timer;
public void timer(String end, final String ddl, final String name) {
    final SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy年MM月dd日HH:mm");
    try {
        final Date endDate = simpleDateFormat.parse(end);
        _timer = new Timer();
        TimerTask task = () → {
                //格式化当前时间
                Date currDate = new Date(System.currentTimeMillis());
                long day = (endDate.getTime() - currDate.getTime()) / (24 * 60 *60 * 1000);
                    Intent intent = new Intent("DAY");
                    Bundle bundle = new Bundle();
                    bundle.putString("day", day + "");
                    bundle.putString("content", ddl);
                    bundle.putString("course", name);
                    intent.putExtras(bundle);
                    sendBroadcast(intent);
                Log.v("notification", day + "");
        };
        _timer.schedule(task, 0, 10*1000);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
public void cancel_Timer() {
    if (_timer != null) {
        _timer.cancel();
        _timer = null;
    }

}
```

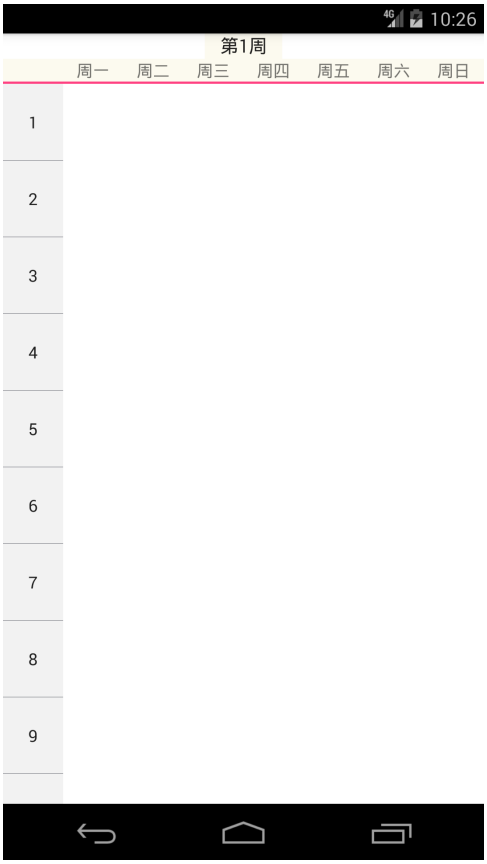
程序运行

程序 app 图标。

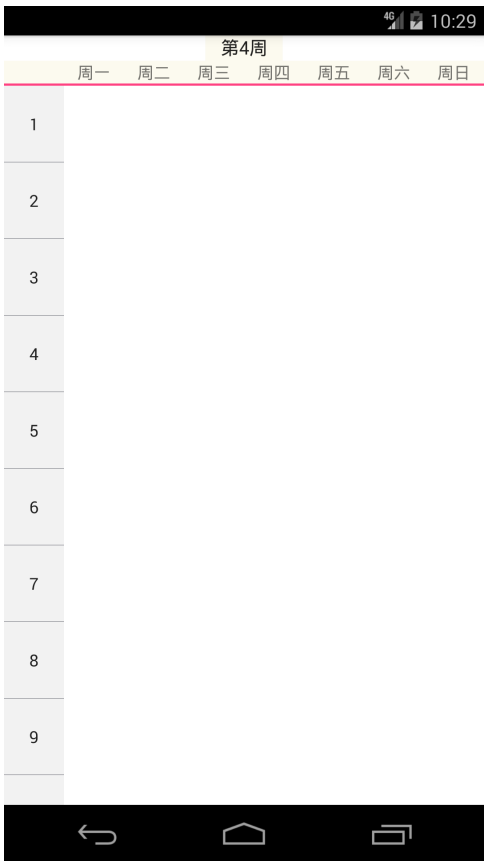
程序运行，显示欢迎界面。



3 秒后显示程序初始主界面。



设置周数



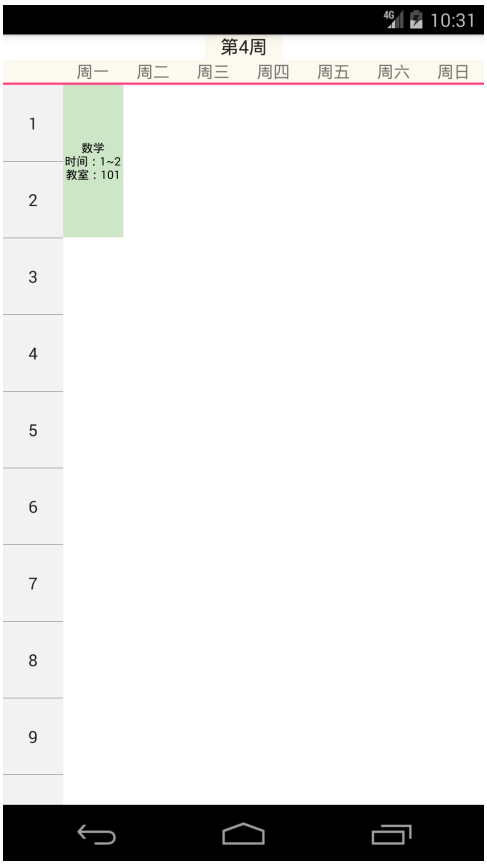
点击 1 按钮，跳转到添加课程界面。


6:43
TimeTable
课程名称
老师
Email或者ftp
(课程资料)
星期
请输入1~7的数字
开始课节
1
结束课节
请输入1~16的数字且大于开始课节
的数字
教室
提交

添加课程信息如下图：

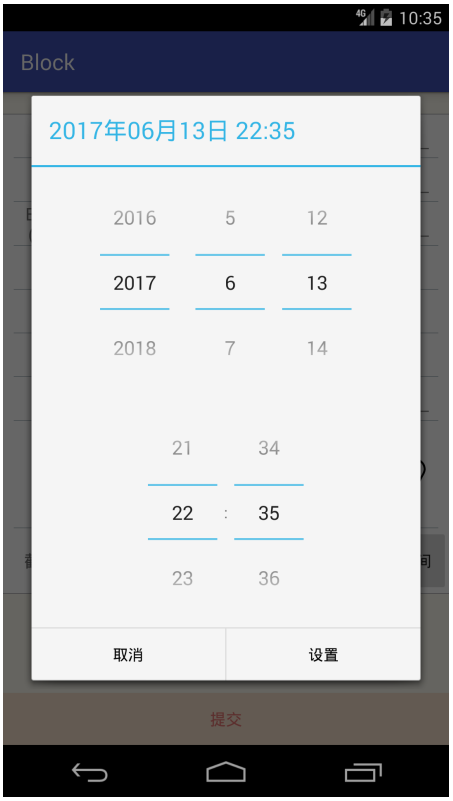

6:48
TimeTable
课程名称
数学
老师
李
Email或者ftp
(课程资料)
123
星期
1
开始课节
1
结束课节
2
教室
101
提交

点击提交按钮，跳转到主界面，刚刚添加的课程已显示在课程表的指定位置上。

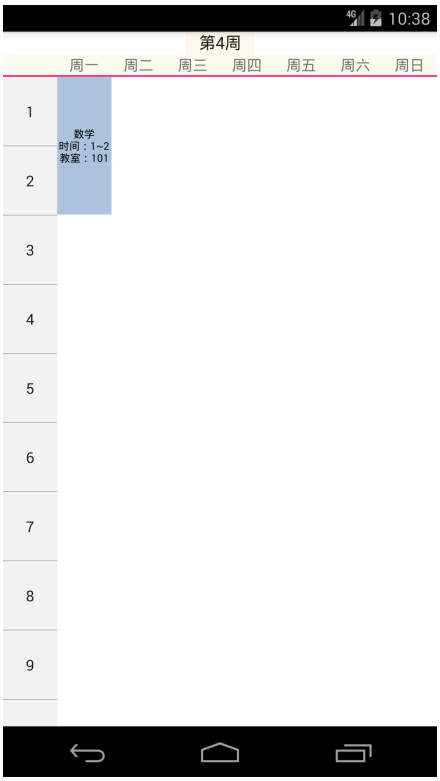


点击刚刚添加好的课程，跳转到显示课程详情界面。

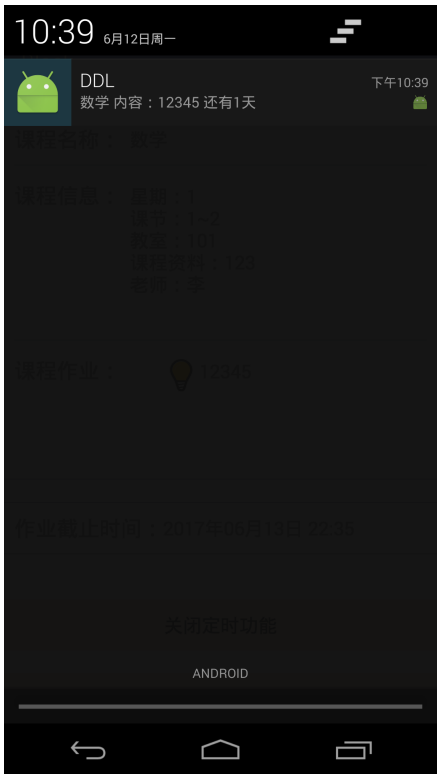
点击编辑按钮，跳转到编辑课程详情界面。点击截止时间后的编辑框，选择截止时间。编辑课程详情如下图：



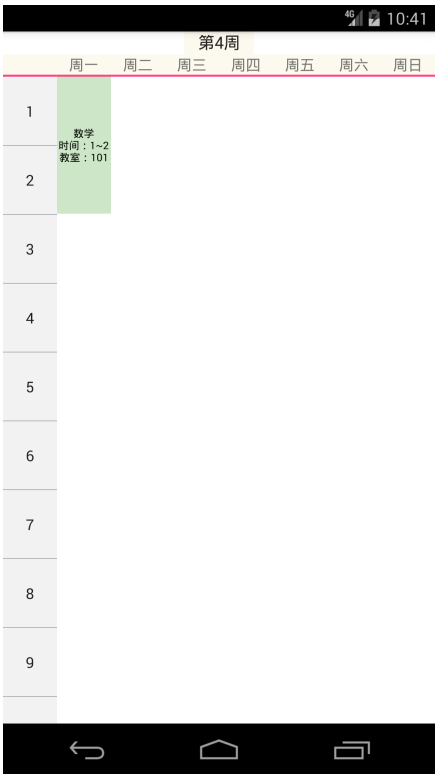
点击提交，跳转回主界面。点击课程，进入显示课程详情界面，刚刚添加的信息已显示在此界面。



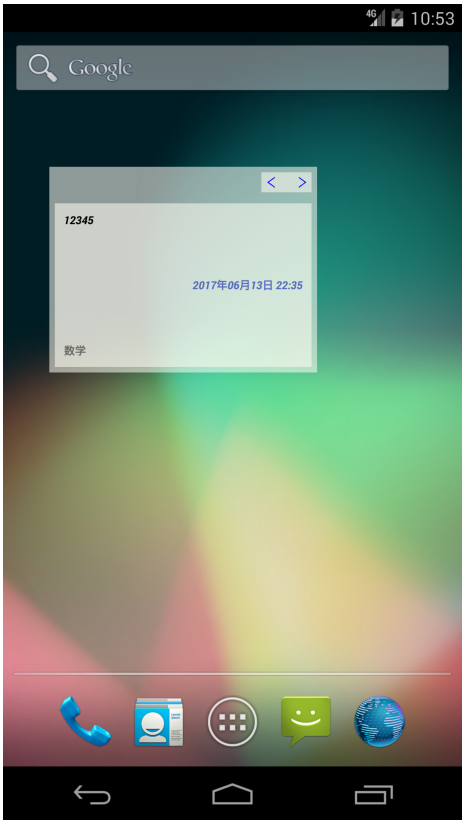
点击开启定时功能按钮，为了测试方便已设为每 10s 发送一次广播，这样在提前一天的时间内，每 10s 就会提醒并震动一次。根据我设置的时间，确实有通知。

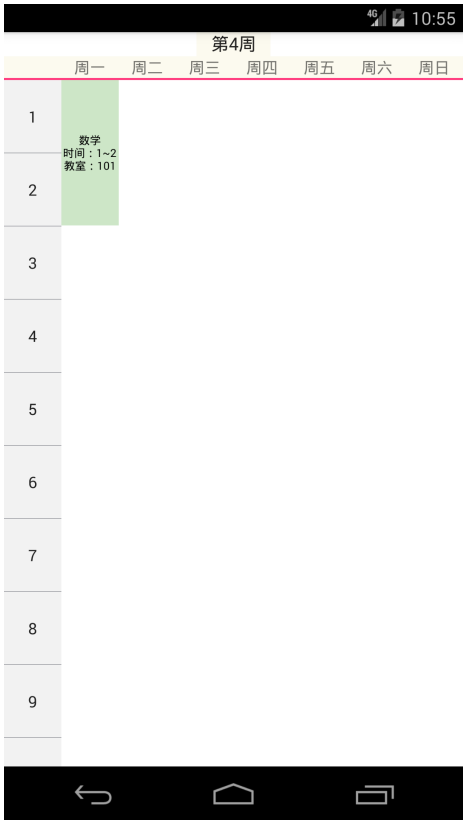


点击通知，跳转到程序主界面。点击课程进入显示课程详情界面，点击关闭定时功能，不再发送通知。

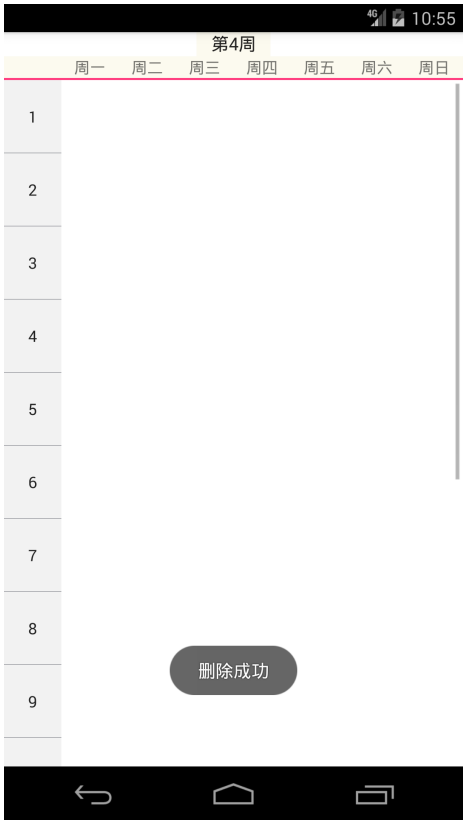


在桌面上添加 widget，widget 上显示了课程、作业以及截止时间，点击右上角方向键，可以调整查看所有作业。点击 widget 跳转回主界面。





点击课程进入显示课程详情界面，点击删除按钮，Toast 信息显示删除成功，程序返回主界面，该课程已被删除。



遇到的问题及解决方法

1. 页面布局的问题。

   一开始考虑使用 gridlayout 进行布局，因为有良好的特性即可以拉伸控件，但是实际做的过程中发现这个控件对于整体的宽度的把握不是很好，里面的 button 的宽度不受限制，整体页面超出屏幕页面。

   因此，后来改为使用 relativelayout 的方式，在 MainActivity 里设置控件的长宽及位置：

2. 第一次运行 app 时获取不到一周七天每一列的 relativelayout 的宽度。

   一开始意识到这个问题是由于第一次运行程序页面上没有课程信息，而可以从数据库获得数据。后来想到应该是在添加到页面这个过程出了问题，而无论在 onStart，onResume 还是在 onCreate 都获取不到控件的宽和高。

   后来通过在网上查询发现，这三个过程其实都没有绘制具体的页面，没绘制页面是获取不到宽和高，也就是宽和高都为 0。所以参照网上博客的方法，使用对一个 layout 的监听函数。

   这个函数是当这个 layout 被绘制的时候调用的，这样在这个函数流获取宽高信息，并在获取之后更新页面。

3. 警告信息：“跳过 31 帧 .....”。

   在 onCreate 写了太多与页面更新有关的问题，结果警告信息：“跳过 31 帧 .....”。

   这主要是因为在 UI 线程里做了太多更页面更新有关的操作。因此，后来将添加 View 的操作都放到了 handler 里去做，这样就解决了这个问题。

4. 截止时间的填写的问题。

   在截止时间的填写那里由于是点击出现一个 picker 来操作的，所以在一开始如果对点击不进行调整就出现了问题。在获得焦点时我们出现 picker 的同时应该将这个 EditText 禁用，防止用户自行更改格式，在失去焦点后再重新使这个 EditText 可以使用，以方便用户更改时间。

5. Widget 更新问题。

   一开始没有考虑 widget 及时更新的问题，只是在新建 widget 的时候更新了数据。经过修改，在 widget 的 onReceive 方法里又进行了更新，当编辑页面提交时和详情界面删除时发送一个广播，这个广播会使 widget 的 onReceive 接收被更新页面，更新页面从数据库里进行数据获取。

6. 多个 timer 计时器问题。

由于有多个课程需要定时器，而且每个定时器需要一个保存状态的信息，所以我在 service 服务里加上了一个类来将 timer 和课程信息合在一起，课程信息是课程的名称与 ddl 和课程作业合起来的字符串（中间有空格），在 service 保存这个类的一个数组存储计时器及其对应的课程信息，每次进入详情界面都会调用 service 的 query_timers 方法查询是否有存在该课程的计时器，如果有设置计时器 button 的文字为关闭定时功能，否则为开启定时功能。这样就可以将课程和计时器对应起来了。