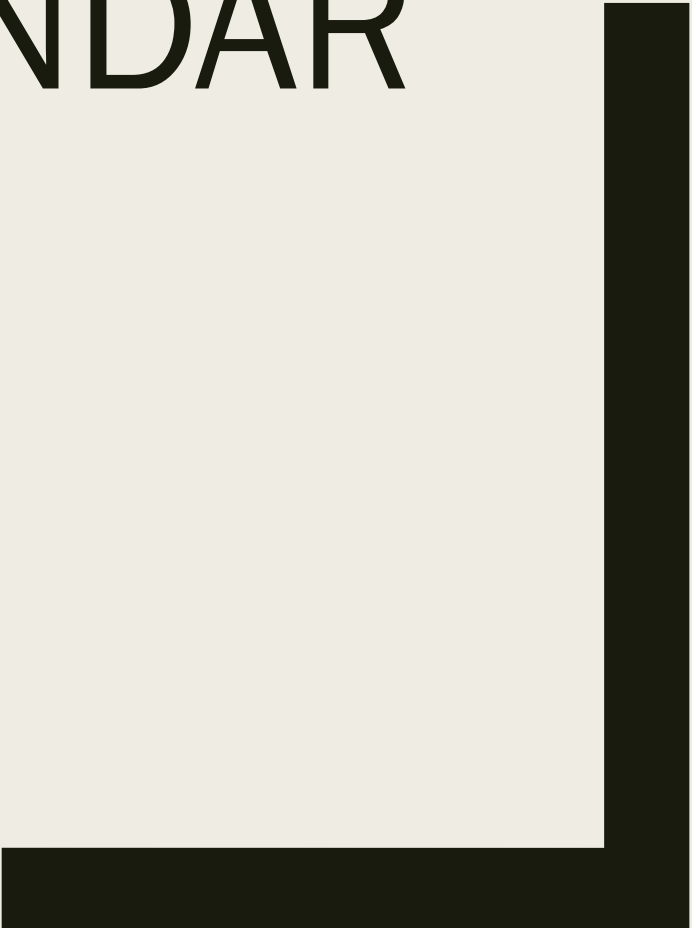




ANDROID CALENDAR

Team-1

Alex Wimer
Connor Mahaffey
Theodore Sosnowski
Chris Wilson



Project Demo

- View events
 - *Monthly*
 - *Daily*
 - *Agenda*
- Add event
 - *Single*
 - *Weekly*
 - *Monthly*
- Edit event
 - *Change date*
 - *Change time*
 - *Change occurrence*
- Delete event

Commits

 97 commits

Commits on Dec 3, 2017



Updated Event View

cwilson11 committed 2 hours ago



a93db20



added buttons to views

t3421 committed 2 hours ago



f31c087



Commits on Dec 2, 2017



Event update conflict check updated

cwilson11 committed 19 hours ago



f2c0658



Can update events

cwilson11 committed 20 hours ago



f5331d7



Adding DayView ...

cmahaffey225 committed 20 hours ago



72f3f86



didnt work

Verified



7b8e6f4



Insight

November 3, 2017 – December 3, 2017

Period: 1 month ▾

Overview

2 Active Pull Requests

0 Active Issues

2

Merged Pull Requests

0

Proposed Pull Requests

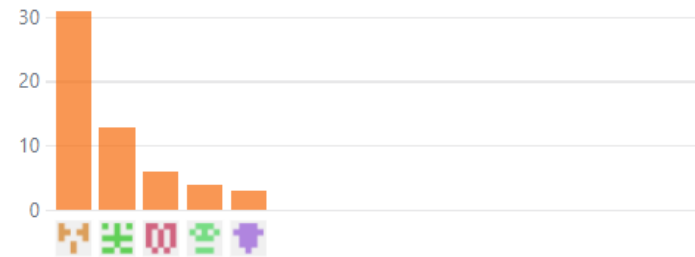
0

Closed Issues

0

New Issues

Excluding merges, **5 authors** have pushed **55 commits** to master and **57 commits** to all branches. On master, **406 files** have changed and there have been **6,901 additions** and **0 deletions**.



2 Pull requests merged by 2 people

Merged

#9 Views 19 days ago

Merged

#7 Add main add code 27 days ago

Team Organization

- Team meetings
 - *Meetings occurred after each class for updates*
 - *Once a month longer meetings pertain to the project*
 - *Communicated by text or in person*
- Task distribution
 - *Appointment of task by group discussion and skill level*

Coding Style and Comments

Link -

<http://www.oracle.com/technetwork/java/javase/documentation/codereviewfaq-136057.html>

- We tried to stick to the code convention linked above for oracle
- Camel case style
- Comments in code are used on a as needed basis for inside methods
 - *Methods should all be commented on for use in Java Docs*
 - *Statements that need clarification or more guidance*
 - *Classes should all be commented on for use in Java Docs*

```
/**
 * Checks for error within the user input such as long comment or title along with missing
 * needed items to add in the event
 *
 * @param startMinute the start minute must be assigned
 * @param startHour the start hour must be assigned
 * @param endMinute the end minute must be assigned
 * @param endHour the end hour must be assigned
 * @param startDay checking the day is not 0
 * @param eventTitle must be checked for for >0 and <64 characters
 * @param eventComments must be checked for <256 characters
 * @return returns true if no errors are found, false is at least one is found
 */
public boolean Noerrors(int startMinute , int startHour, int endMinute, int endHour,
                        int startDay, String eventTitle, String eventComments){

    if(startMinute == 0 && startHour == 0 || endMinute == 0 && endHour == 0 || startDay == 0 ||
        eventTitle.length()==0 || eventTitle.length()>32 || eventComments.length()>256){

        StringBuilder errors =new StringBuilder();
        if ( eventComments.length()>256){errors.append("comment length,");} //Check comments
        if ( eventTitle.length() == 0 || eventTitle.length()>32){errors.append
            ("event title length, ");} //Check blank title
        if ( startMinute == 0 && startHour == 0){errors.append("start time, ");} //Check start time
        if ( endMinute == 0 && endHour == 0){errors.append("end time, ");} //Check end time
        if ( startDay == 0){errors.append("start Date, ");} //Check blank date
        Toast.makeText(getBaseContext(), text: " Please correct the error/s " +
            errors, Toast.LENGTH_LONG).show(); //push message
        return false;
    }
    return true;
}
```

Design Pattern

Singleton Pattern

- Single instance of the database is used by views.

```
public class EventsDb extends SQLiteOpenHelper {
```

```
    //Database Instance
```

```
    private static EventsDb databaseInstance;
```

```
    public static EventsDb getInstance(Context context) {
```

```
        if(databaseInstance == null)
```

```
            databaseInstance = new EventsDb(context);
```

```
        return databaseInstance;
```

```
    }
```

```
    private EventsDb(Context context ){super(context, DATABASE_NAME,null, DATABASE_VERSION);}
```

```
public class EventView extends AppCompatActivity {
```

```
    int startY = 0, startM = 0, startD = 0, startH = 0, startMi = 0, endH = 0, endM = 0, returnToView=0;
```

```
    String occurrenceId, eventId, extraId, color;
```

```
    private int id = 0;
```

```
    private int occurrenceCheck = 0;
```

```
    EventsDb db = EventsDb.getInstance(this);
```

```
    Event event = new Event();
```

EXTRACT METHOD

Refactoring

Before

```

36 startHour = data.getIntExtra( "selectedHour" , 0);
37 startMinute = data.getIntExtra("selectedMinute", 0);
38 - boolean isValid = validate(startHour , startMinute , endHour , endMinute);
39 - if (isValid){
37 + if (validate(startHour , startMinute , endHour , endMinute)){
40 38 Toast.makeText(getBaseContext(),"hour = " + startHour + " minute = " + startMinute , Toast.LENGTH_LONG).show();
41 - modifiedStartHour = startHour;
42 - if ( startHour > 12 ){
43 -     modifiedStartHour = startHour - 12;
44 -     ampm = "PM";
45 - }
46 - else if (startHour == 0){startHour+=1;}
47 - // String modifiedStartMinute = String.format("%02d", startMinute);
48 - ((TextView)findViewById(R.id.start_time_display)).setText(modifiedStartHour + ":" + String.format("%02d" ,startMinute));
39 + ((TextView)findViewById(R.id.start_time_display)).setText(getTimeString(startHour , startMinute));
49 40 }
50 41 else{
51 42     startHour = transH;
@@ -63,14 +54,7 @@ public void onActivityResult(int requestCode, int resultCode, Intent data) {
63 54 boolean isValid = validate(startHour , startMinute , endHour , endMinute);
64 55 if (isValid){
65 56     Toast.makeText(getBaseContext(),"hour = " + endHour + " minute = " + endMinute , Toast.LENGTH_LONG).show();
66 - modifiedEndHour = endHour;
67 - if ( endHour > 12 ){
68 -     modifiedEndHour = endHour - 12;
69 -     ampm = "PM";
70 - }
71 - else if (endHour == 0){endHour+=1;}
72 - // String modifiedEndMinute = String.format("%02d", endMinute);
73 - ((TextView)findViewById(R.id.end_time_display)).setText(modifiedEndHour + ":" + String.format("%02d" , endMinute));
57 + ((TextView)findViewById(R.id.end_time_display)).setText(getTimeString(endHour , endMinute));
74 58 }

```

After

```

/**...*/
public boolean validate(int startH , int startM , int endH , int endM) {
return (((startH == 0 && startM == 0) || (endH == 0 && endM == 0)) || ((startH < endH) ||
((startH == endH) && (startM < endM ))));
}

```


Testing

```
import org.junit.Before;
import org.junit.Test;
import static org.junit.Assert.*;

public class AddEventTest {

    private AddEvent tester;

    @Before
    public void setUp() throws Exception {
        tester = new AddEvent();
    }

    @Test
    public void getDateString() throws Exception {
        assertEquals( expected: "1 January 2017" , tester.getDateString( year: 2017, month: 1, day: 1));
        assertEquals( expected: "31 November 2017" , tester.getDateString( year: 2017, month: 11, day: 31));
        assertEquals( expected: "" , tester.getDateString( year: 0, month: 1, day: 1));
        assertEquals( expected: "" , tester.getDateString( year: 2017, month: 1, day: 0));
    }

    @Test
    public void validate() throws Exception {
        assertEquals( expected: true , tester.validate( startH: 0, startM: 0, endH: 12, endM: 30));
        assertEquals( expected: true , tester.validate( startH: 24, startM: 0, endH: 24, endM: 1));
        assertEquals( expected: true , tester.validate( startH: 12, startM: 30, endH: 0, endM: 0));
        assertEquals( expected: false , tester.validate( startH: 12, startM: 0, endH: 11, endM: 1));
    }

    @Test
    public void getTimeString() throws Exception {
```

```
@Test
public void getTimeString() throws Exception {
    assertEquals( expected: "" , tester.getTimeString( hour: 0, minute: 0));
    assertEquals( expected: "12:00 PM" , tester.getTimeString( hour: 12, minute: 0));
    assertEquals( expected: "12:30 AM" , tester.getTimeString( hour: 0, minute: 30));
    assertEquals( expected: "9:01 PM" , tester.getTimeString( hour: 21, minute: 1));
}

@Test
public void getSpinnerPosition() throws Exception {
    assertEquals( expected: 0 , tester.getSpinnerPosition( color: ""));
    assertEquals( expected: 0 , tester.getSpinnerPosition( color: "Yellow"));
    assertEquals( expected: 1 , tester.getSpinnerPosition( color: "Fuchsia"));
    assertEquals( expected: 2 , tester.getSpinnerPosition( color: "Red"));
    assertEquals( expected: 3 , tester.getSpinnerPosition( color: "Green"));
    assertEquals( expected: 4 , tester.getSpinnerPosition( color: "Purple"));
    assertEquals( expected: 5 , tester.getSpinnerPosition( color: "Blue"));
    assertEquals( expected: 6 , tester.getSpinnerPosition( color: "Maroon"));
    assertEquals( expected: 7 , tester.getSpinnerPosition( color: "Teal"));
}
}
```

Test for add event class