Konstantine Tavadze

Victor Miller

Senior Project

UniPal Android Application

Documentation

Table of Contents

1 Introduction

1A Definition

UniPal is an Android application designed to help students keep track of upcoming assignments and deadlines.

1B Background

UniPal was built by me, Konstantine Tavadze, during Spring and Summer of 2018 for my Senior Project (CMPS 450) at Ramapo College.

In Spring 2018, I took Artificial Intelligence (CMPS 331), where I built my first Android app. That app was called Konane (Hawaiian checkers) and implemented lots of different algorithms, such as best-first search, branch and bound, and minimax. Building Konane was difficult, but it taught me a lot about Android development.

Therefore, I decided to use my newfound knowledge for my senior project, and build another Android app. However, this time I wanted to have user authentication and to hook it up to a database. Fortunately, Google Firebase simplified both of those tasks significantly, and I was able to build out all the features mentioned in the specs, as well as others from the wish list.

2 Installation

1. Connect the submitted USB drive to your PC
2. Open File Explorer on your PC
    a. Navigate to *This PC*
    b. Under device, double click on the USB drive
    c. Navigate to UniPal\app\build\outputs\apk\debug
    d. Copy app-debug.apk
3. Connect an Android device to your PC using a USB cable
4. The Android device will raise a connection notification
    a. Click *Allow*
5. Open File Explorer on your PC
    a. Navigate to *This PC*
    b. Under devices, double click on the Android device
    c. Double click on the device name
    d. Double click on the *Download* folder
    e. Paste app-debug.apk
6. Disconnect the Android device from your PC
7. Open My Files on your Android device
    a. Tap on *Internal Storage*
    b. Tap on the *Download* folder
    c. Tap on app-debug.apk***

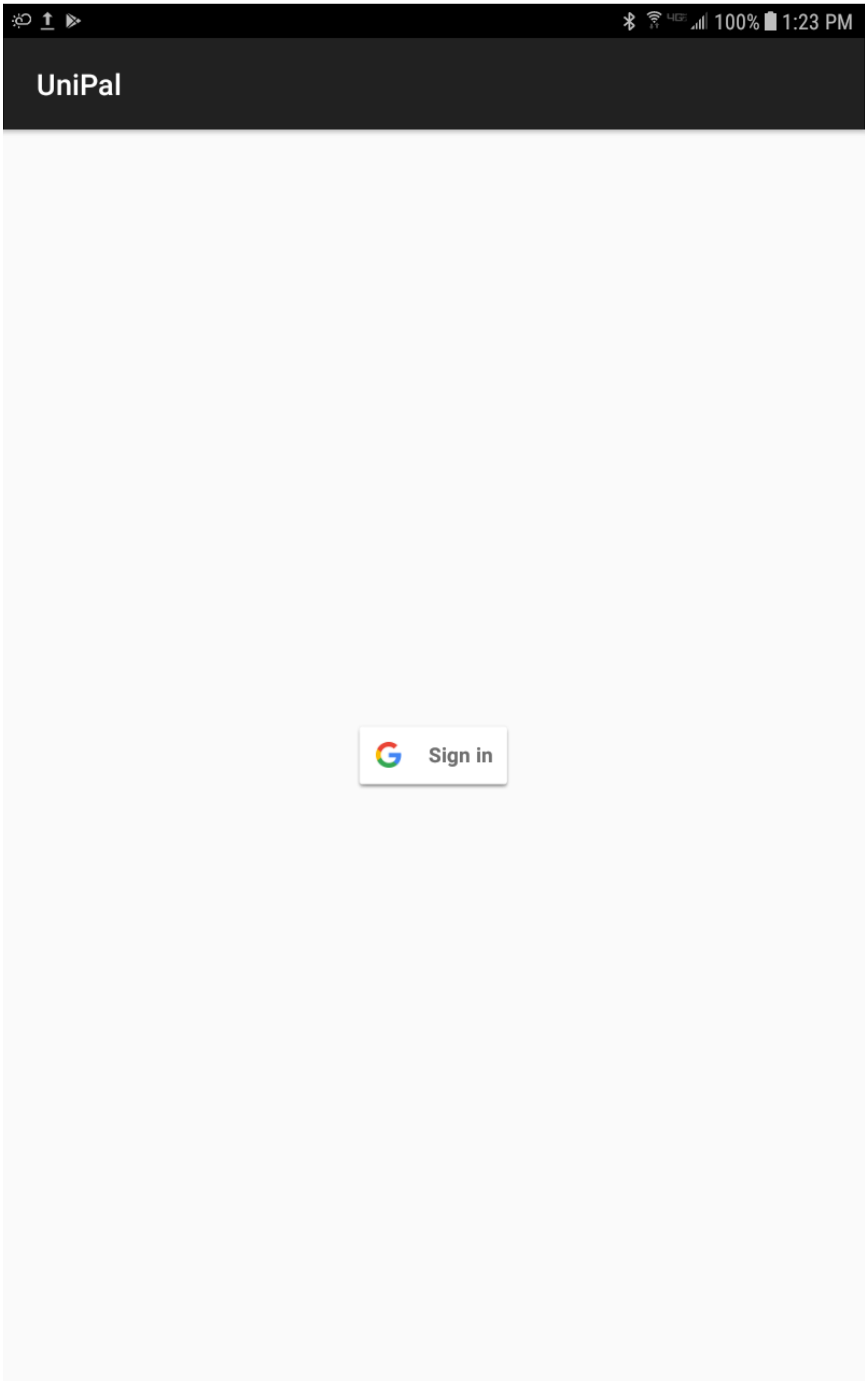
***You may have to modify your device security settings to allow installation of applications from unknown sources

3 Manual

3A Sign In Activity

When the app is opened for the first time, the user will be directed to the Sign In activity. In the middle of the view, there is a Google Sign In button. Once the button is clicked, a list of available Google accounts (if any) will pop up. This interface is widely used and should be familiar to most users. Once a valid account is selected and sign in is successful, the user is redirected to the Main activity. If sign in fails for any reason (e.g. no internet access), a snack bar will appear at the bottom of the view notifying the user.

Signing in is only necessary the first time around (or if the user signs out manually through the user fragment). Otherwise, the credentials persist even if the app is terminated or the device is rebooted, and the user is taken directly to the dashboard fragment in the Main activity.


100% 1:23 PM
UniPal
Sign in

3B Main Activity

Once the user is signed in, she is redirected to the Main activity, which serves as the container for ALL subsequent views. There are 5 main views:

1. Dashboard
2. Calendar
3. Courses
4. Schools
5. User

There are also 4 sub-views:

1. Date
2. Event
3. Course
4. School

All the views are associated with at least one XML and one Java file. Each of these is an Android fragment, which allows us to swap them in and out of the Main activity as needed. The major difference between main views and sub-views is how they are accessed by the user. The user can navigate to any of the 5 main views from anywhere in the app using the bottom navigation bar. In contrast, to access the 4 sub-views the user must go through one of the main views. For example, the user can only access the Date view by clicking one of the dates in the Calendar view. Similarly, the user can only access the Event view by clicking one of the events in the recycler (found on the Dashboard and the Date view).

3C Dashboard

The Dashboard view is the focal point of the application. It is the first of the 5 main views and the default one loaded into the Main activity container when the application is opened/re-opened. It features 4 buttons at the top that look like tabs:

1. All
2. Month
3. Week
4. Day

These allow the user to easily filter her events based on how many days ahead she wants to see. The Day button will only show the events that are due today. The Week button will show the events due over the next 7 days; The Month button will show the events due over the next 31 days. Meanwhile, the All button will show all the events, regardless of their due date or status.

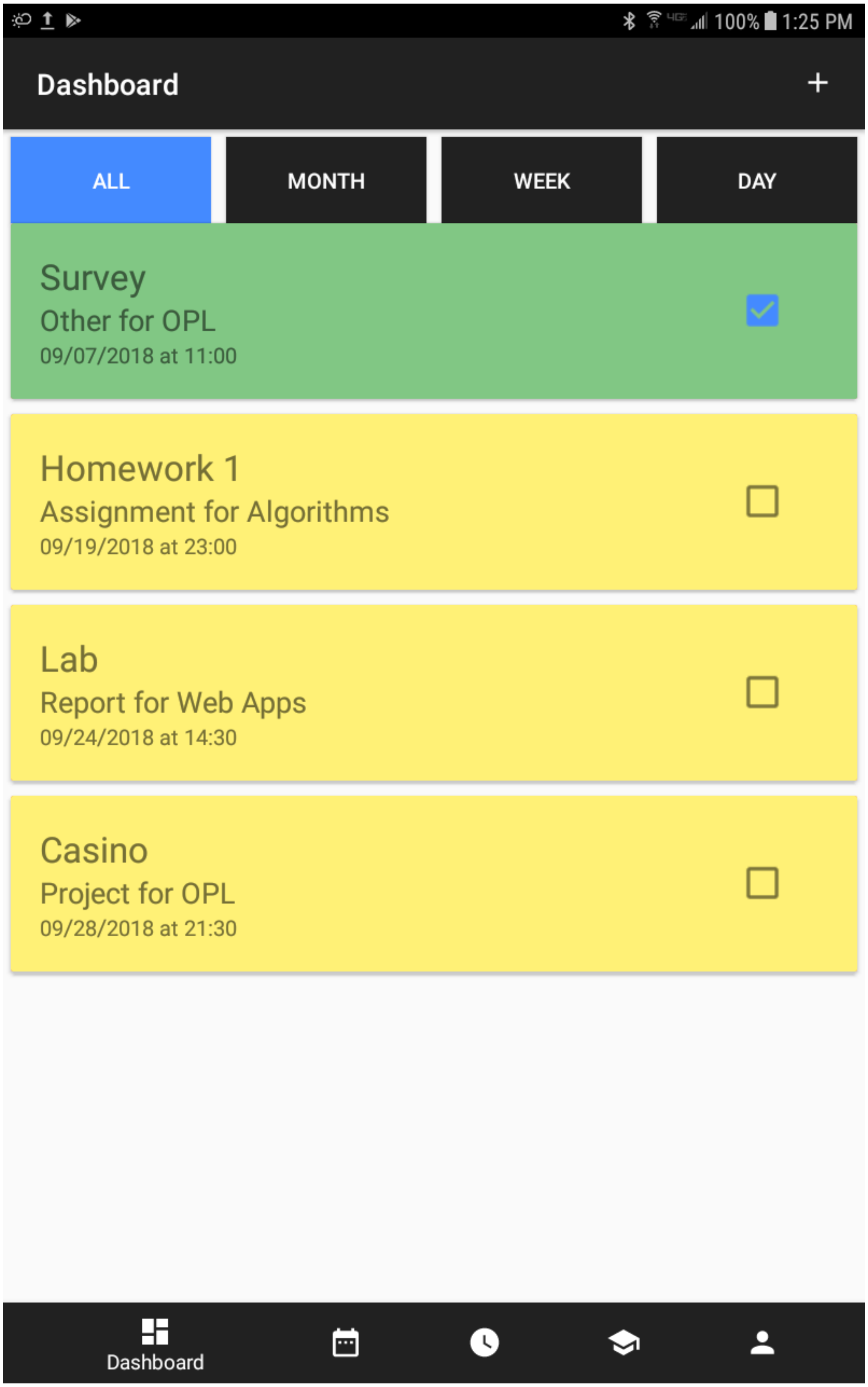
In the top-right of the action bar, there is a plus icon. Clicking this icon will bring up the New Event dialog, which contains the following fields:

1. Name
2. Type
3. Course***
4. Alarm
5. Date
6. Time

None of the fields are required. Fields will be auto-populated as necessary.

***The course spinner will only show up if the user has already added at least one course to the Courses view. The status of all new events will initially be set to incomplete. Please note that the user must click on the date field to access the DatePickerDialog. Same goes for the time field, which uses the military format (24 hours). Once all desired information has been entered, the user must click DONE for the actual event to be created.

The Dashboard view is essentially a recycler that lists the filtered events, but it also allows for additional functionality. For example, the user can toggle event status directly from the recycler by checking/un-checking the box to the right of each event. The user can also swipe-to-delete events from the recycler, which will trigger a temporary snack bar at the bottom offering her the option to undo (restore the event). Clicking one of the events will navigate the user to the corresponding Event view, where she can edit, toggle or delete the event. The Date view offers similar functionality, because it essentially employees the same recycler as the Dashboard view.



Survey
Other for OPL
09/07/2018 at 11:00
[✓]

Homework 1
Assignment for Algorithms
09/19/2018 at 23:00
[ ]

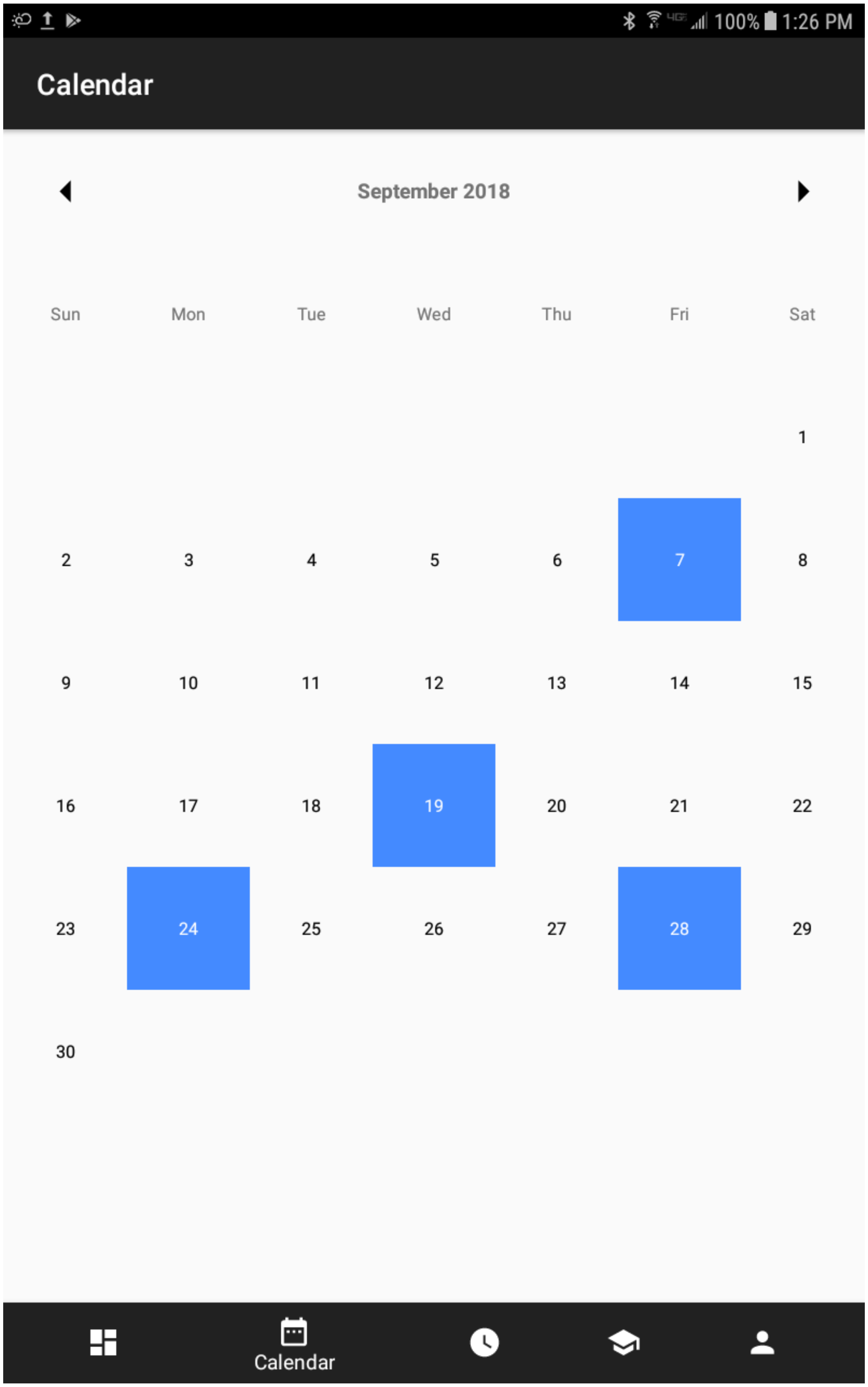
Lab
Report for Web Apps
09/24/2018 at 14:30
[ ]

Casino
Project for OPL
09/28/2018 at 21:30
[ ]

3D Calendar

The Calendar view is the 2nd of the 5 main views. It allows the user to visualize how the events are spread out and when they are due. The user can navigate between months using the left/right arrows at the top. The dates of interest are highlighted in blue. Clicking one of the dates will navigate the user to a corresponding Date view, where all matching events will be listed, if any.

For this view, I used a third-party implementation of the Material CalendarView by Prolific Interactive. Please see references for the GitHub link and other information.



Calendar

September 2018

Sun	Mon	Tue	Wed	Thu	Fri	Sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

3E Courses

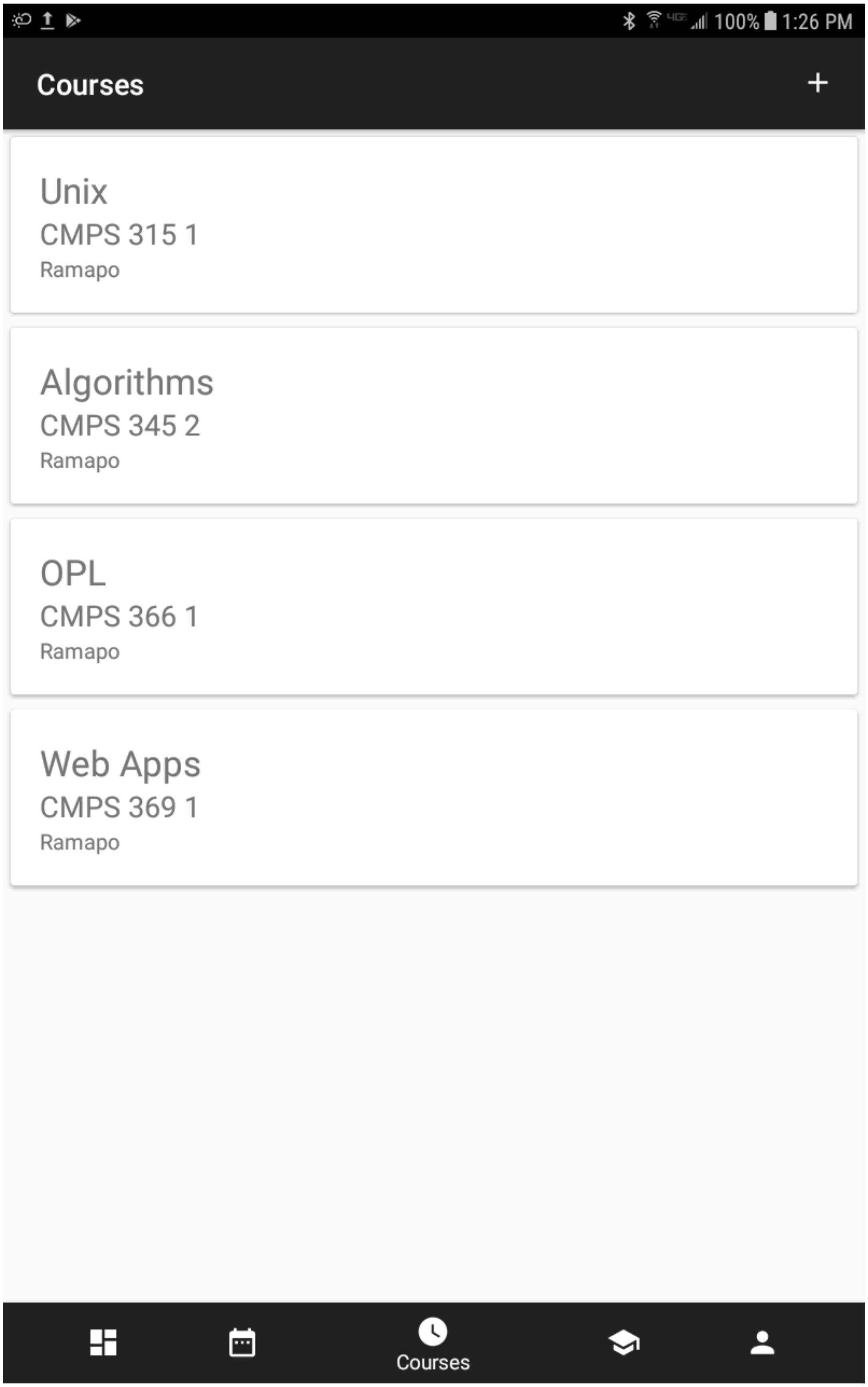
The Courses view is the 3rd of the 5 main views. It allows the user to add new courses or browse existing ones. Clicking one of the courses will navigate the user to the corresponding Course view, where she can edit or delete the course.

In the top-right of the action bar, there is a plus icon. Clicking this icon will bring up the New Course dialog, which contains the following fields:

1. Name
2. Department
3. Number
4. Section
5. School***

None of the fields are required. Fields will be auto-populated as necessary.

***The school spinner will only show up if the user has already added at least one school to the Schools view. Once all desired information has been entered, the user must click DONE for the actual course to be created.



Courses

Unix
CMPS 315 1
Ramapo

Algorithms
CMPS 345 2
Ramapo

OPL
CMPS 366 1
Ramapo

Web Apps
CMPS 369 1
Ramapo

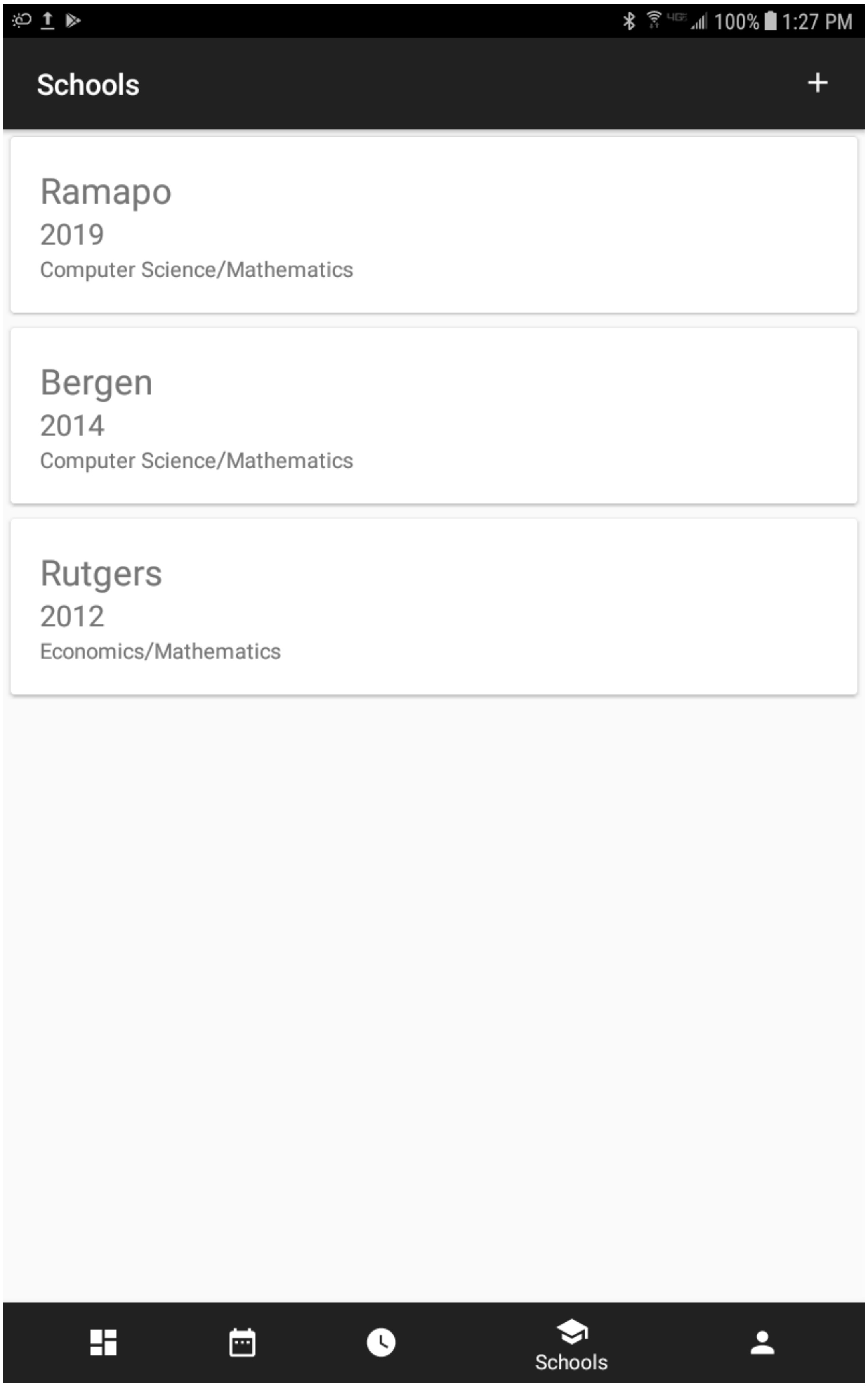
3F Schools

The Schools view is the 4th of the 5 main views. It allows the user to add new schools or browse existing ones. Clicking one of the schools will navigate the user to the corresponding School view, where she can edit or delete the school.

In the top-right of the action bar, there is a plus icon. Clicking this icon will bring up the New School dialog, which contains the following fields:

1. Name
2. Year
3. Major
4. Minor

None of the fields are required. Fields will be auto-populated as necessary. Once all desired information has been entered, the user must click DONE for the actual school to be created.



Schools

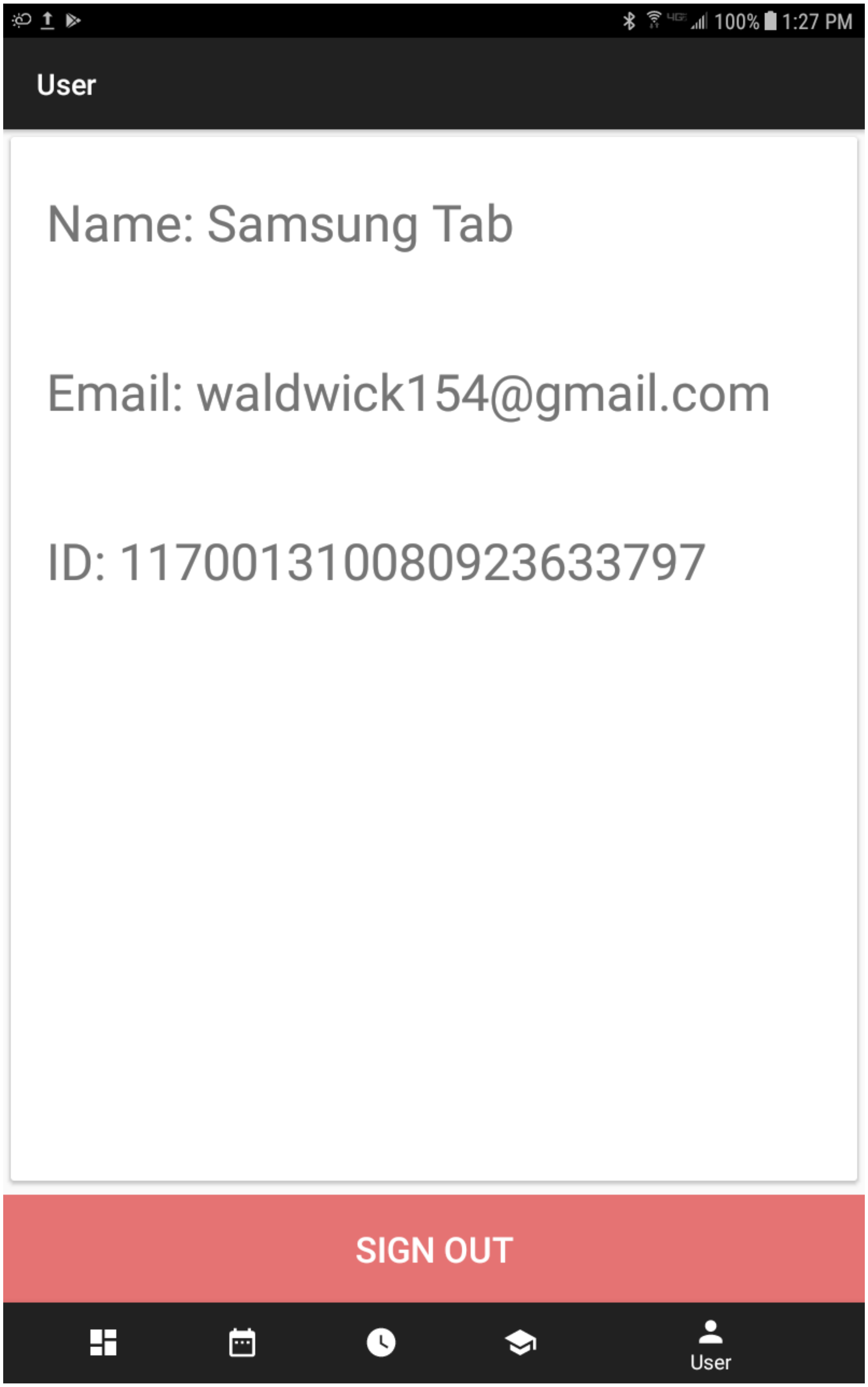
Ramapo
2019
Computer Science/Mathematics

Bergen
2014
Computer Science/Mathematics

Rutgers
2012
Economics/Mathematics

3G User

The User view is the last of the 5 main views. It allows the user to view the name, email and id associated with her Google account. Clicking the big red button at the bottom will sign out the user and redirect her to the Sign In activity.


100% 1:27 PM

User

Name: Samsung Tab

Email: waldwick154@gmail.com

ID: 117001310080923633797

3H Date

The Date view is the first of the 4 sub-views. It is accessible by clicking one of the dates from the Calendar view. It allows the user to view the events matching the selected date, if any.

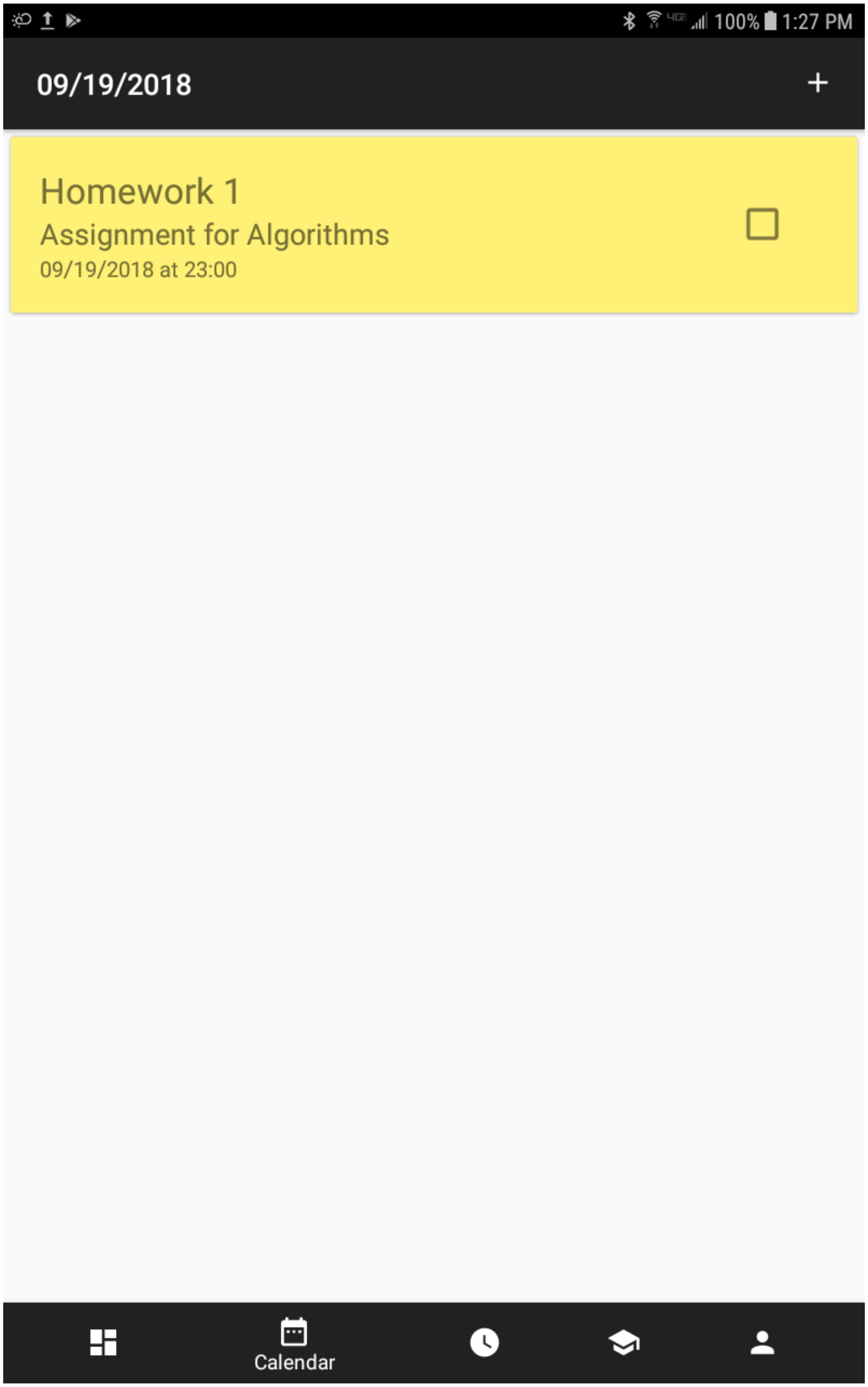
In the top-right of the action bar, there is a plus icon. Clicking this icon will bring up the New Event dialog, which contains the following fields:

1. Name
2. Type
3. Course***
4. Alarm
5. Time

None of the fields are required. Fields will be auto-populated as necessary.

***The course spinner will only show up if the user has already added at least one course to the Courses view. The status of all new events will initially be set to incomplete. The date will be pre-set to match the one being viewed. Please note that the user must click on the time field to access the TimePickerDialog, which uses the military format (24 hours). Once all desired information has been entered, the user must click DONE for the actual event to be created.

The Date view is essentially a recycler that lists matching events, but it also allows for additional functionality. For example, the user can toggle event status directly from the recycler by checking/un-checking the box to the right of each event. The user can also swipe-to-delete events from the recycler, which will trigger a temporary snack bar at the bottom offering her the option to undo (restore the event). Clicking one of the events will navigate the user to the corresponding Event view, where she can edit, toggle or delete the event.



1:27 PM

09/19/2018

Homework 1

Assignment for Algorithms

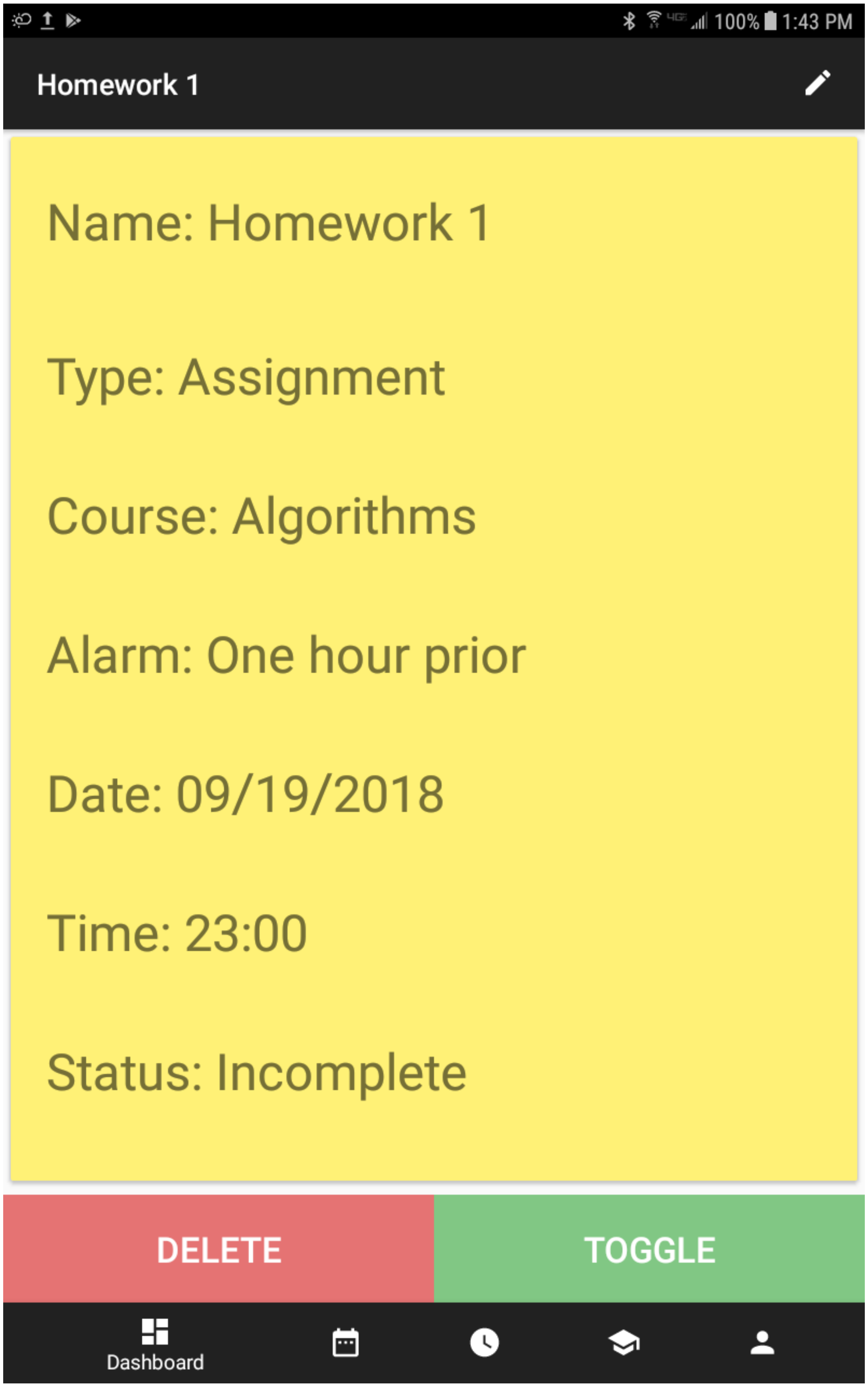
09/19/2018 at 23:00

3I Event

The Event view is the 2nd of the 4 sub-views. It is accessible by clicking one of the events from either the Dashboard or the Date view. It allows the user to view, edit, toggle or delete the selected event.

In the top-right of the action bar, there is a pencil icon. Clicking this icon will allow the user to start editing the fields. Once all desired fields have been updated, the user must click UPDATE to commit the changes.

Clicking the toggle button in the bottom right will toggle the status of the event. Clicking the delete button in the bottom left will delete the event from the database.



100% 1:43 PM

Homework 1

Name: Homework 1

Type: Assignment

Course: Algorithms

Alarm: One hour prior

Date: 09/19/2018

Time: 23:00

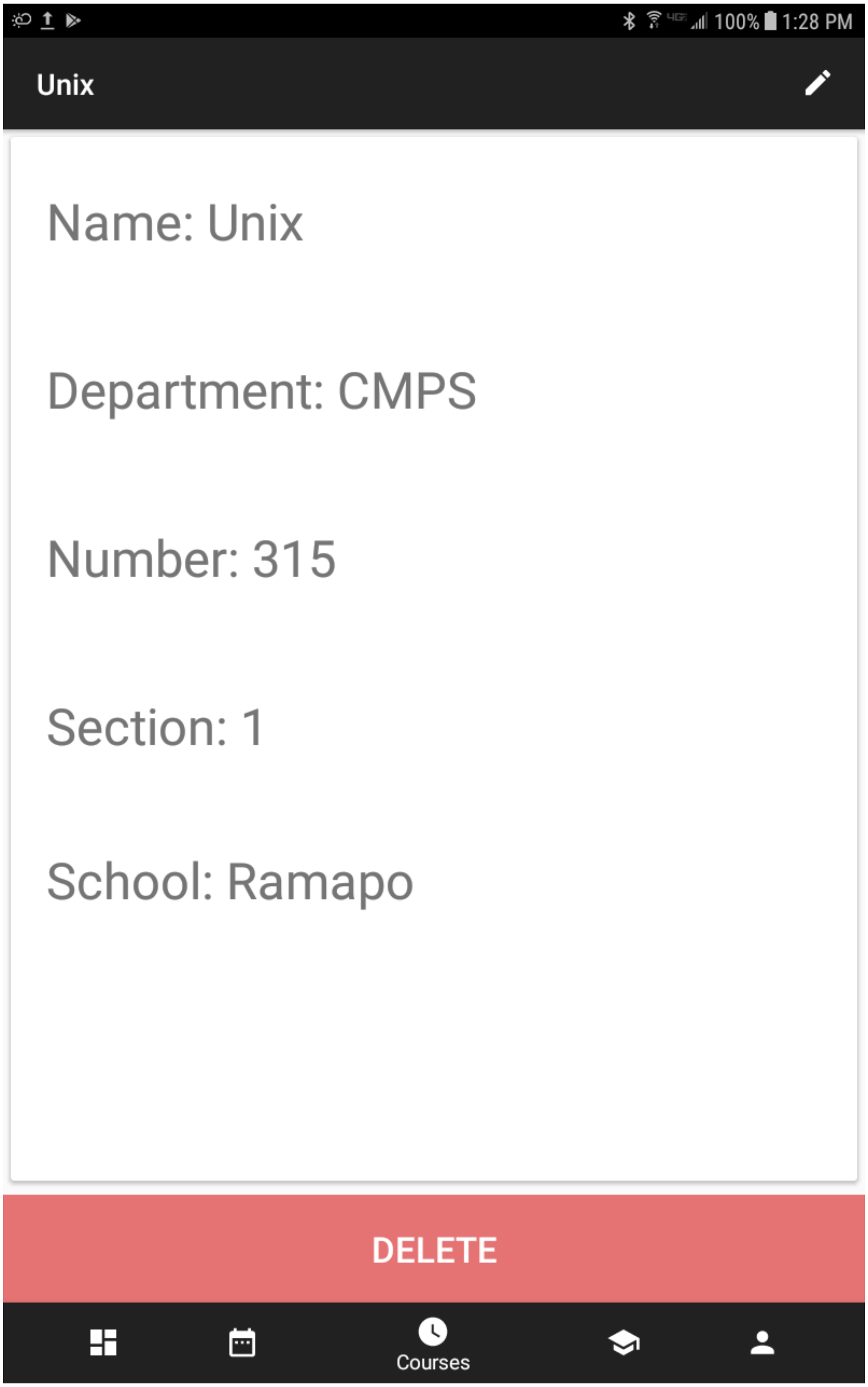
Status: Incomplete

3J Course

The Course view is the 3nd of the 4 sub-views. It is accessible by clicking one of the courses from the Courses view. It allows the user to view, edit or delete the selected course.

In the top-right of the action bar, there is a pencil icon. Clicking this icon will allow the user to start editing the fields. Once all desired fields have been updated, the user must click UPDATE to commit the changes.

Clicking the delete button at the bottom will delete the course from the database.



Unix

Name: Unix

Department: CMPS

Number: 315

Section: 1

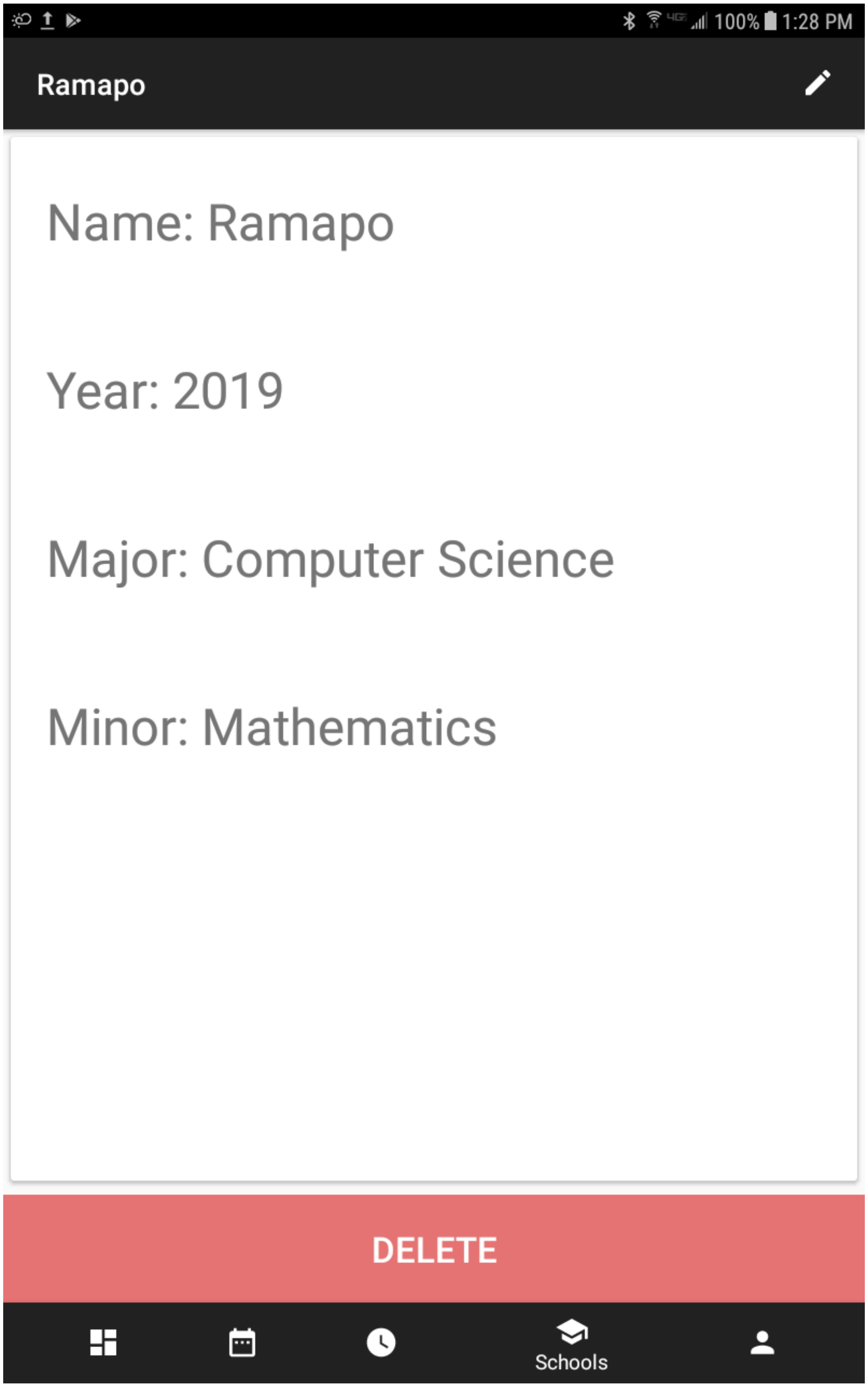
School: Ramapo

3K School

The School view is the last of the 4 sub-views. It is accessible by clicking one of the schools from the Schools view. It allows the user to view, edit or delete the selected school.

In the top-right of the action bar, there is a pencil icon. Clicking this icon will allow the user to start editing the fields. Once all desired fields have been updated, the user must click UPDATE to commit the changes.

Clicking the delete button at the bottom will delete the school from the database.


1:28 PM 100%

Ramapo

Name: Ramapo

Year: 2019

Major: Computer Science

Minor: Mathematics

DELETE

Schools

4 Design

4A Summary

Going into this project, I was reluctant to add authentication and database to the list of core features. However, after doing some research, I realized that it was feasible with the help of Google Firebase. Firebase is an application development platform that can be used for native or mobile apps. In fact, Android Studio has a built-in Firebase tool that makes is simple to add all the features you need.

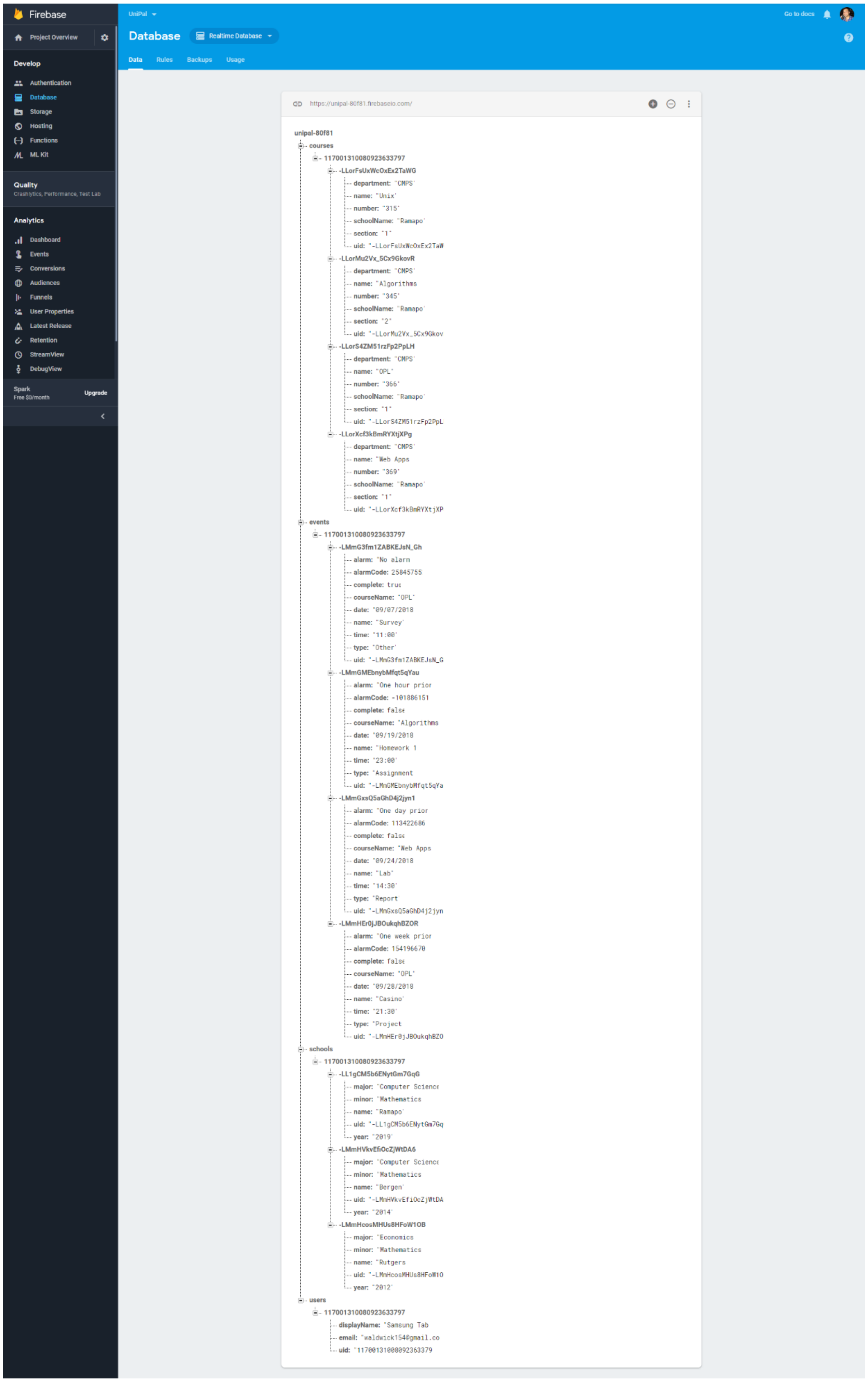
Although Firebase offers a ton of sign in options, I figured Google Sign In would suffice in this case. I was contemplating adding the plain old email/password option as well, but decided against it, since I'm using the Google ID (which is unique for each Google account) to group data in the database.

The database was designed to be as flat as possible, to minimize complexity and maximize usability by matching the classes used in the code. The DB is split up into 4 main groups: events, courses, schools and users. Inside each of these is a list of key-value pairs, where the keys are the users' unique Google ID's. This way, it's super simple to get only the items associated with a specific user. For example, here's the structure of the events in the database:

1. Events
    a. Unique user id
        i. Unique event id
            1. Event data in key-value pairs
        ii. Unique event id
            1. Event data in key-value pairs
    b. Unique user id
        i. Unique event id
            1. Event data in key-value pairs
        ii. Unique event id
            1. Event data in key-value pairs

Similar approach is used for courses and schools. The users group is slightly simpler:

1. Users
    a. Unique user id
        i. Display name
        ii. Email
        iii. Unique user id



```
unipal-80f81
 ⊟-- courses
     ⊟- 117001310080923633797
         ⊟-- -LLorFsUxWcOxEx2TaWG
               |-- department: "CMPS'
               |-- name: "Unix'
               |-- number: "315'
               |-- schoolName: "Ramapo'
               |-- section: "1"
               `-- uid: "-LLorFsUxWcOxEx2TaW
         ⊟-- -LLorMu2Vx_5Cx9GkovR
               |-- department: "CMPS'
               |-- name: "Algorithms
               |-- number: "345'
               |-- schoolName: "Ramapo'
               |-- section: "2"
               `-- uid: "-LLorMu2Vx_5Cx9Gkov
         ⊟-- -LLorS4ZM51rzFp2PpLH
               |-- department: "CMPS'
               |-- name: "OPL"
               |-- number: "366'
               |-- schoolName: "Ramapo'
               |-- section: "1"
               `-- uid: "-LLorS4ZM51rzFp2PpL
         ⊟-- -LLorXcf3kBmRYXtjXPg
               |-- department: "CMPS'
               |-- name: "Web Apps
               |-- number: "369'
               |-- schoolName: "Ramapo'
               |-- section: "1"
               `-- uid: "-LLorXcf3kBmRYXtjXP
 ⊟-- events
     ⊟- 117001310080923633797
         ⊟-- -LMmG3fm1ZABKEJsN_Gh
               |-- alarm: "No alarm
               |-- alarmCode: 25845755
               |-- complete: true
               |-- courseName: "OPL"
               |-- date: "09/07/2018
               |-- name: "Survey'
               |-- time: "11:00'
               |-- type: "Other'
               `-- uid: "-LMmG3fm1ZABKEJsN_G
         ⊟-- -LMmGMEbnybMfqt5qYau
               |-- alarm: "One hour prior
               |-- alarmCode: -101886151
               |-- complete: false
               |-- courseName: "Algorithms
               |-- date: "09/19/2018
               |-- name: "Homework 1
               |-- time: "23:00'
               |-- type: "Assignment
               `-- uid: "-LMmGMEbnybMfqt5qYa
         ⊟-- -LMmGxsQ5aGhD4j2jyn1
               |-- alarm: "One day prior
               |-- alarmCode: 113422686
               |-- complete: false
               |-- courseName: "Web Apps
               |-- date: "09/24/2018
               |-- name: "Lab"
               |-- time: "14:30'
               |-- type: "Report'
               `-- uid: "-LMmGxsQ5aGhD4j2jyn
         ⊟-- -LMmHEr0jJBOukqhBZOR
               |-- alarm: "One week prior
               |-- alarmCode: 154196670
               |-- complete: false
               |-- courseName: "OPL"
               |-- date: "09/28/2018
               |-- name: "Casino'
               |-- time: "21:30'
               |-- type: "Project
               `-- uid: "-LMmHEr0jJBOukqhBZO
 ⊟-- schools
     ⊟- 117001310080923633797
         ⊟-- -LL1gCM5b6ENytGm7GqG
               |-- major: "Computer Science
               |-- minor: "Mathematics
               |-- name: "Ramapo'
               |-- uid: "-LL1gCM5b6ENytGm7Gq
               `-- year: "2019'
         ⊟-- -LMmHVkvEfiOcZjWtDA6
               |-- major: "Computer Science
               |-- minor: "Mathematics
               |-- name: "Bergen'
               |-- uid: "-LMmHVkvEfiOcZjWtDA
               `-- year: "2014'
         ⊟-- -LMmHcosMHUs8HFoW1OB
               |-- major: "Economics
               |-- minor: "Mathematics
               |-- name: "Rutgers
               |-- uid: "-LMmHcosMHUs8HFoW1O
               `-- year: "2012'
 ⊟-- users
     ⊟- 117001310080923633797
         |-- displayName: "Samsung Tab
         |-- email: "waldwick154@gmail.co
         `-- uid: "117001310080923633379
```

Organizing the data in this fashion has a lot of advantages, since there's no unnecessary nesting. It minimizes the amount of data having to be downloaded when a single item, such as an event, is updated. It also allows for the data to be transferred directly into a corresponding class instance in the code, so that it can be easily manipulated.

Notifications were a bit tricky, since there are a lot of different implementations depending on the SDK version, such as AlarmManager, JobScheduler and WorkManager. I decided to go with AlarmManager, since it has been around the longest, but I had to increase the minimum SDK requirement to account for Doze Mode, which was introduced in SDK 23 to improve battery life. Basically, what it does is it prevents apps from waking the device and showing notifications at the exact time that they are scheduled, which is not ideal in this situation. To make sure my notifications appear exactly when they are scheduled, even if the device is in Doze Mode, I had to use the AlarmManager's setExactAndAllowWhileIdle method call, which requires SDK 23.

The rest of the app was relatively straightforward. The only two activities are Sign In and Main. The Main activity serves as the container for all subsequent views, which are fragments. Aside from portability, this design is ideal for persisting the data that has already been downloaded from the database, which minimizes mobile data usage. It allows us to swap fragments in and out of the Main activity with all the data at hand, without having to fetch data individually for each view.

4B Classes

There are several types of classes used throughout the app. We will cover these types individually:

1. Basic classes
    a. Event
    b. Course
    c. School
    d. User
2. Adapter classes
    a. EventsRecyclerAdapter
    b. CoursesRecyclerAdapter
    c. SchoosRecyclerAdapter
3. Utility classes
    a. RecyclerEventTouchHelper
    b. AlarmScheduler
    c. AlarmReceiver
    d. Database

4C Basic Classes

Event, Course, School and User classes consist of relevant properties and their getters/setters. They all contain a uid (unique id) property, the value for which is generated by Firebase as the item is added to the database. This uid is used to override the equals method. So, if two items have the same uid, they are treated as equals, even if the rest of the fields do not match up.

The Event class also contains a method called getCalendarDay, which converts the date from one format to another and is used for decorating the corresponding dates in the Calendar view. The User class contains static properties, because there can only be one user per running application instance.

4D Adapter Classes

The recycler adapter classes facilitate the recycler (list) functionality for Events (Dashboard/Date), Courses and Schools views. They define the fields of recycler items, display the relevant data, and redirect the user to a corresponding sub-view when she clicks one of the items.

The EventsRecyclerAdapter is also tasked with determining the background color (yellow/green) of each recycler item based on the event status (incomplete/complete), as well as handling the checkbox clicks and updating the status in the database.

4E Utility Classes

The utility classes facilitate various features throughout the app and differ considerably from one another. For example, the RecyclerEventTouchHelper class is specifically designed to enable swipe-to-delete functionality for the Events recyclers on the Dashboard and Date views. On the other hand, the Database class is tasked with communicating with the Firebase database and making sure all the data is current using ChildEventListeners.

The AlarmScheduler class oversees scheduling new alarms, cancelling existing alarms and showing notifications. Meanwhile, the AlarmReceiver class captures scheduled alarm broadcasts and calls AlarmScheduler to display the notifications to the user.

5 Conclusion

Building UniPal app has been an amazing learning experience. I feel extremely proud to have implemented all the features listed in the specifications. From authentication to database to notifications, I now feel even more confident that I can build almost anything on Android. I am looking forward to continuing to learn and experiment as I prepare to build yet another Android application for OPL this semester (Casino).

6 References

- https://developer.android.com/reference/
- https://firebase.google.com/docs/guides/
- https://github.com/prolificinteractive/material-calendarview